TÉCNICO LISBOA

An Introductory approach to Audio Signal Processing for Music Transcription

Diana Raquel Piteira Félix

Thesis to obtain the Master of Science Degree in

Electrotechnical and Computers Engineering

Supervisor: Prof. António Luis Campos da Silva Topa

Examination Committee

Chairman: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino
Supervisor: Prof. António Luis Campos da Silva Topa
Member of the Committee: Prof. António José Castelo Branco Rodrigues

November 2018

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

The atom is more like a tune than a table. Since a table is made of atoms, is it not just a more complicated tune? Are we, as Plato's Simmias argued, no more than complicated tunes?

(MARTIN GARDNER, The Whys of a Philosophical Scrivener)

La première fois que je me servis d'un phonoscope, j'examinai un si bémol de moyenne grosseur. Je n'ai, je vous assure, jamais vu chose plus répugnante. J'appelai mon domestique pour le lui faire voir.

(ERIK SATIE, Mémoires d'un amnésique)

Acknowledgements

Esta Dissertação teve como instituição de acolhimento, o Instituto de Telecomunicações (IT), Pólo de Lisboa.

Resumo

Esta tese foca-se nas técnicas de processamento de sinal aplicadas à transcrição de um excerto a três vozes de uma Fuga de J.S.Bach. O objectivo é estudar e aplicar algumas técnicas básicas de processamento de sinal a um sinal acústico e retirar informação musical relevante que irá permitir a conversão do sinal em notação simbólica. A transcrição musical é por si só uma tarefa complexa. De forma a limitar os problemas que este desafio coloca, foi escolhida uma gravação de piano solo de uma fuga, que é uma técnica de composição baseada numa forte independência melódica das vozes. Este modelo introduz várias simplificações como a existência de um único instrumento e descarta a necessidade de conduzir uma análise simultanamente harmónica e melódica. Os desafios colocados por esta tarefa envolvem a computação do espectro do sinal acústico, a detecção de eventos ritmicos, remoção de ruído, detecção do espectro das frequências fundamentais e mapeamento melódico.

Palavras-chave: transcrição musical, processamento de sinal, fuga, STFT, DFT

Abstract

This thesis is focused on signal processing techniques applied to the transcription of an excerpt of a three voice fugue from J.S.Bach. The idea is to study and apply some basic signal processing tools to an audio input signal and retrieve relevant music information that will allow to convert the signal into a symbolic representation. Music transcription is a complex task. So in order to narrow down the problems this challenge naturally imposes, it was chosen a piano solo recording of a fugue, which is a western composition technique based on a strong melodic independence. This introduces several simplifications like no need for instrument source detection or simultaneous harmonic and melodic complex analysis. The challenges faced in this task involve the computation of the spectrum of the input signal, rhythm events detection, background spectrum removal, fundamental frequencies detection and pitch tracking.

Keywords: music transcription, signal processing, fugue, STFT, DFT

Contents

List of Tables

List of Figures

Nomenclature

δ	Spectral resolution
λ_n	Eigenvalues
Ω_M	Main lobe width
ϕ	Phase of a sinusoid
B	Inharmonicity coefficient
c	Speed of sound
f_0	Fundamental frequency
f_n	n-th order harmonic frequency
f_s	Sampling rate
H	Window hop-size
K	Main lobe width in bins
k	Frequency bins
l	Frame number
M	Window length
N	DFT size
p	Pitch
Q	Quality factor
T	Sampling period
t_n	n-th sampling instant
$u(x,t)$	Displacement function
$W(k)$	DFT of the window function at frequency index k

$w(n)$	Window function
w_k	Angular frequency
$x(n)$	Signal amplitude at sample n
$x(t_n)$	Signal amplitude at instant t_n
$X(w_k)$	DFT of x at frequency w_k

Glossary

DFT Discrete Fourier Transform

FFT Fast Fourier Transform

HF Harmonic Function

IDFT Inverse Discrete Fourier Transform

JND Just-noticeable difference

MIR Music information retrieval

ODF Onset detection function

STFT Short-Time Fourier Transform

Chapter 1

Introduction

1.1 Motivation

Music transcription involves the translation of an acoustic signal into a symbolic representation, consisting on musical notes, the respective time events and the classification of the instruments used. In other words, it consists on listening to a piece of music and writing down the musical notation for that piece.

Similarly to natural language, listening, writing and reading music requires education. An untrained listener will mostly tap along the rhythm, hum the melody, recognize musical instruments and even music patterns such as chorus. However, the inner lines and all the rich and subtle harmonic nuances that make music sound special - or not so - are mostly unsconcious. A trained listener though, can recognize different pitch intervals, timing relationships, sub-melodies, all mentally encoded into symbolic music notation. So one learns on how to pay attention and turn conscious the otherwise unconscious phenomenons which do shape and enrich one's perception of the experience. This human hability to understand and process music information - and sound in general - is not only related to its physical auditory system but also to ones own neural coding and physcological mechanisms, which is the field of study of psychoacoustics. Leaving the inherent human perspective aside, an objective complete transcription can still be an extremely complex challenge. It requires the retrieval of simultaneous information on several dimensions, like pitch, timing and instrument to resolve all the sound events so the goal is usually redefined as being either to notate as many of the constituent sounds as possible or to transcribe only some well-defined part of the music signal, for example, the dominant melody, the chords and bass progression or the musical key. This allows to perform automatic tasks like song identification, cover identification, genre/mood classification, score following, know as music information retrieval (MIR). When automatized, music transcription systems can assist musicians and composers to efficiently analyze compositions they only have in the form of acoustic recordings, and provide control on flexible mixing, editing, selective signal coding, sound synthesis or computer-

aided orchestration. However, it also opens the door to the in depth approach of concrete problems on signal processing with a wide range of applications. It provides a comprehensive set of descriptors to populate databases of music metadata, which can then be used for statistical analysis and content-based machine-learning approaches, that can also be applied to more popular fileds such as speach processing or physcoacoustics and perceptual research.

1.2 Overview

Despite the monophonic music transcription being considered a solved issue, poliphonic transcription with several sources, however, still poses a challenge nowadays. Much progress has been made in this area over the last decades. The approaches are varied and can be based on computational models of human auditory system or stochastic processes that explore the random character of sounds, for example. While newer and more sophisticated algorithms perform increasingly better, they also get considerably complex, computationally expensive and still quite depend on the type of input. More recent and efficient approaches employ machine-learning techniques (supervised or unsupervised, where a minimum number of prior assumptions on the signal are made) and statistical models like hidden Markov chains or Bayesian inference. The music transcription task, which starts from the analaysis of a spectrogram, can be approached as an image processing problem, so tools like convolution neural networks, popular for computer vision tasks, can also be applied to signal processing. These techniques allow to overcome the individual treatment and syncronization of the multiple problems posed by poliphonic music transcription. However, they do rely on a sizeable and validated database from which to train the network which can be a challenge by itself.

1.3 Objectives

The purpose of this thesis is to provide a study framework for the developpment of competences on the basics of audio signal processing tools and apply them to a concrete problem which is the transcription of polyphonic music. This thesis aims to provide no new solutions to a widely investigated problem or to improve the already existing methodologies, but instead to understand and explore the potencials of signal processing techniques and evaluate the challenges of polyphonic music transcription.

1.4 Thesis Outline

This thesis is divided into five chapters. In chapter 2, we provide an overview of the Discrete Fourier Transform and its properties, together with the Short-Time Fourier Transform and its ap-

plications to spectral analysis and audio signal processing. In chapter 3, we provide some background on music theory, from the basic physical concepts of the harmonic series to some of the most relevant tuning systems in western music. Chapter 4, contains a detailed description of the algorithms developed for transcribing an excerpt of a Bach's fugue and chapter 5, finally summarizes the methods and findings presented in this work.

Chapter 2

Fourier Transform

2.1 The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) of a signal $x(n) \in \mathbb{C}^N$, with $n \in \mathbb{Z}$ and $n = 0, 1, 2, ..., N-1$, is defined as:

$$X(w_k) = \sum_{n=0}^{N-1} x(t_n) e^{-jw_k t_n} = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \qquad k = 0, 1, 2, ..., N-1 \tag{2.1}$$

where $t_n = nT$ is the n-th sampling instant, $w_k = 2\pi k/Nf_s$ is the k-th frequency sample and $fs = 1/T$ is the sampling rate. To understand the meaning of the expression (2.1) and the effect of the DFT when applied to a discrete signal x, let's consider the *Euler's equality* applied to the exponential term of the transform so that

$$s_k(t_n) = e^{jw_k t_n} = e^{j2\pi kn/N} = \cos(w_k t_n) + j\sin(w_k t_n) \tag{2.2}$$

The term $s_k(t_n)$ defines a complex sampled sinusoid designated as the kernel of the transform. The kernel consists on a sampled sinusoid with N discrete frequencies w_k, equally spaced between 0 and the sampling rate $w_s = 2\pi f_s$. This way, the DFT provides a measure of the magnitude and phase of a complex sinusoid present in the signal x on the frequency w_k and can be interpreted as the dot product of the signals x and s_k, which determines the projection coefficients of x on the complex sinusoid $\cos(w_k t_n) + j\sin(w_k t_n)$. So from the definition of dot product between two vectors x e y

$$\langle x, y \rangle = \sum_{n=0}^{N-1} x(n)\overline{y(n)} \tag{2.3}$$

the DFT defined in (2.1) can also be written as

$$X(w_k) = \langle x, s_k \rangle = \sum_{n=0}^{N-1} x(n)\overline{s_k(n)} = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \qquad k = 0, 1, 2, ..., N-1 \tag{2.4}$$

2.1.1 Inverse Discrete Fourier Transform

Let's consider the signals $x \in \mathbb{C}^N$ and $s_k \in \mathbb{C}^N$ as vectors in a space of dimension N. With the sum of the projections of x on N-vectors s_k, it is possible to recover the original signal when ${s_k}_{k=0}^{N-1}$ is an orthogonal base of $\mathbb{C}^N$, which means that the vectors on the basis of this space are linearly independent. Let's consider x a complex sinusoid

$$x(n) = e^{j2\pi nf_0}, \qquad \text{with } f_0 = f_k T = \frac{f_k}{f_s} \tag{2.5}$$

The dot product of x and the complex sinusoids s_k (2.2) then comes as

$$\langle x, s_k \rangle = \sum_{n=0}^{N-1} x(n)\overline{s_k(n)} = \sum_{n=0}^{N-1} e^{j2\pi nf_0}e^{j2\pi nk/N} = \sum_{n=0}^{N-1} e^{j2\pi n(f_0 - k/N)} \tag{2.6}$$

Applying to (2.6) the formula for the sum of a geometric progression $(1 - z^N)/(1 - z)$, the dot product can be written as

$$\langle x, s_k \rangle = \frac{1 - e^{j2\pi(f_0 - k/N)^N}}{1 - e^{j2\pi(f_0 - k/N)}} \tag{2.7}$$

From (2.7) the dot product $\langle x, s_k \rangle$ will be zero for $f_0 \neq k/N$ and equal to N (2.6) only when $f_0 = k/N$, so that

$$f_k = k\frac{f_s}{N}, \qquad k = 0, 1, 2, ..., N-1 \tag{2.8}$$

This way we conclude that the sinusoids defined in (2.2) are an orthogonal basis of the space for harmonics with frequencies f_k that verify the condition (2.8). These frequencies correspond to whole periods of N samples and are generated by N roots of unity of the complex plane, as represented in Figure 2.1 where each root satisfies the condition

$$z^N = [W_N^k]^N = [e^{jw_k T}]^N = [e^{jk2\pi/N}]^N = e^{jk2\pi} = 1 \tag{2.9}$$

These roots result on the division of the unity circle into N equal parts with one point anchored at $z = 1$. The sampled sinusoids generated by integer powers of the N roots of unity are plotted in Figure 2.2. These are the sampled sinusoids $[W_N^k]^n = e^{j2\pi kn/N} = e^{j\omega_k nT}$ used by the DFT. As the sinusoids s_k are orthogonal and are a basis for $\mathbb{C}^N$, unlike the continuous Fourier Transform, the DFT always has an inverse because the number of samples of the signal is always finite. Considering the projection of x in s_k as

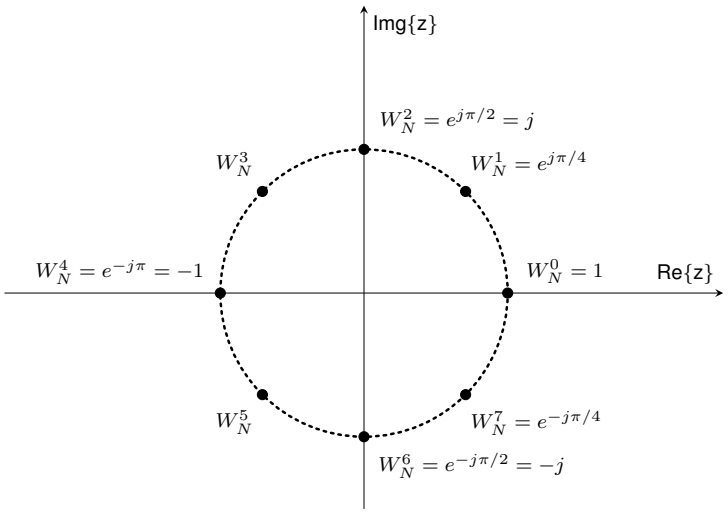

Figure 2.1: N-th roots of unity for N=8.

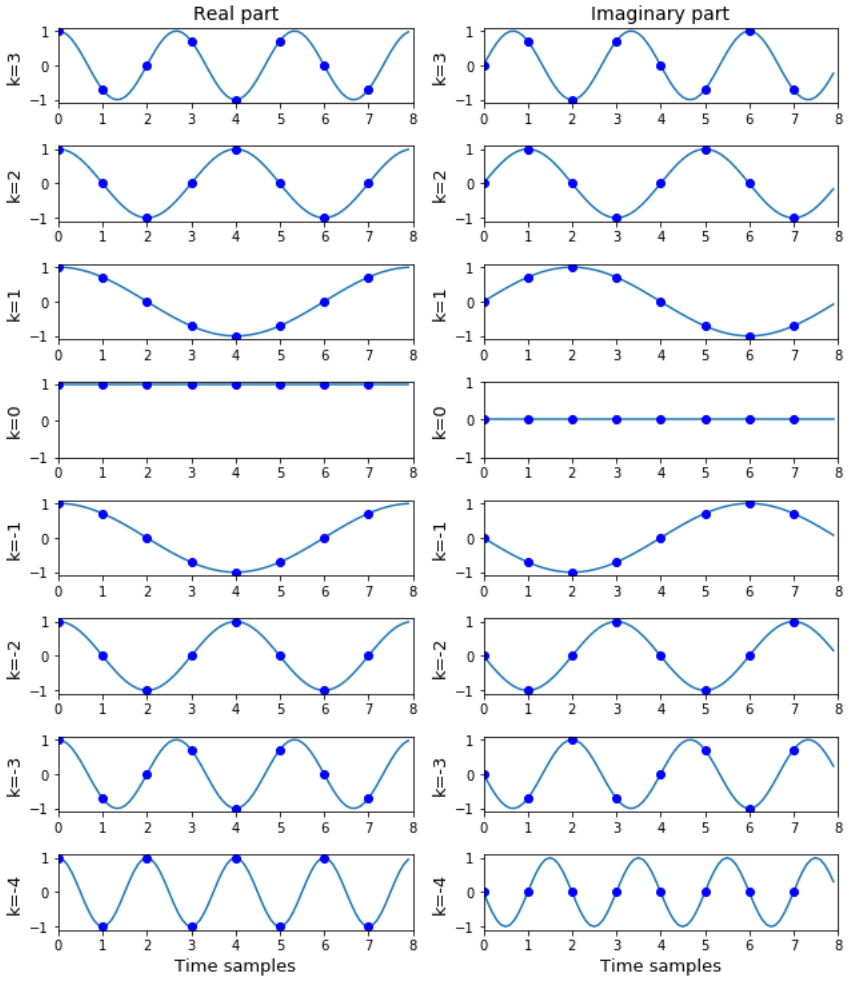

Figure 2.2: Complex simusoids used by the DFT for N=8.

$$P_{s_k}(x) = \frac{\langle x, s_k \rangle}{\| s_k \|^2} s_k = \frac{X(w_k)}{N} s_k \tag{2.10}$$

and that the Inverse Discrete Fourier Transform (IDFT) is computed by the sum of the projections

$$x(n) = \sum_{k=0}^{N-1} \frac{X(k)}{N} s_k(n), \qquad n = 0, 1, 2, ..., N-1 \tag{2.11}$$

where $X(k)/N$ corresponds to the projection coefficients of x in s_k, the IDFT can be defined by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(w_k) e^{jw_k t_n} = \frac{1}{N} \sum_{k=0}^{N-1} X(w_k) e^{j2\pi nk/N}, \qquad n = 0, 1, 2, ..., N-1 \tag{2.12}$$

In conclusion, $X(w_k)$ is called the spectrum of x and can be interpreted as the sum of the projections of x on the orthogonal basis $s_k{}_{k=0}^{N-1}$ so its inverse IDFT is the recovery of the original signal as a superposition of its projections on the N complex sinusoids.

2.1.2 Non-periodic signals

In the particular case of periodic signals $x(t) = x(t + NT)$, with $t \in (-\infty, \infty)$, the DFT can be interpreted as the computation of the coeffcients of the Fourier series of $x(t)$, from a period of the sampled signal $x(nT)$, $n = 0, 1, \ldots, N-1$. In this case the spectrum will be zero except in $\omega = \omega_k$, for $k \in [0, N-1]$, as illustrated in Figure 2.3.

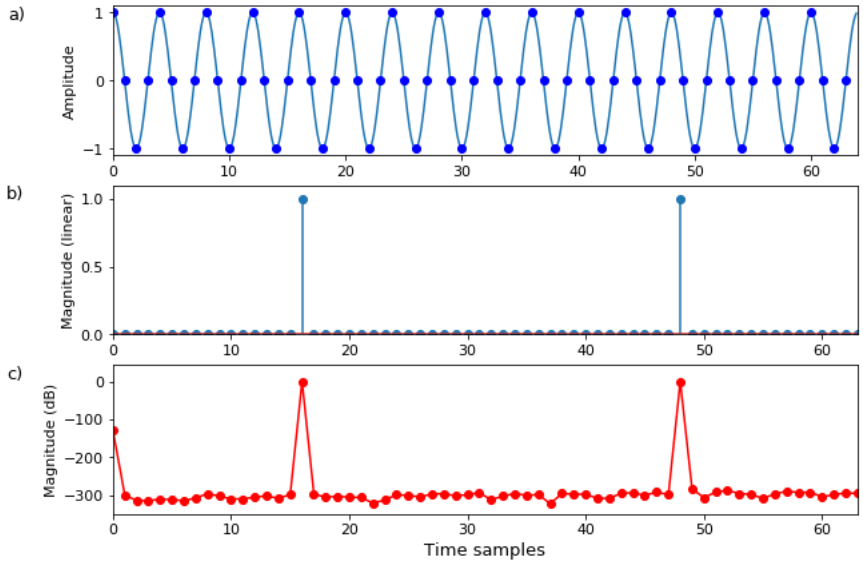

Figure 2.3: Sampled sinusoid at frequency $k = N/4$, $f = f_s/4$ and $N = 64$. a) Time waveform. b) Magnitude spectrum. c) Magnitude spectrum in decibels.

However, when the period of x does not correspond exactly to the roots of w_k - which is the case in most of the analysis of acoustic real signals - the energy of the sinusoid is spread along

all the frequency bins. This effect, designated by *spectral leakage* or *cross-talk*, can be imagined as being equivalent to abruptly truncating the sinusoid on its edges so it can not be reduced by simply increasing N. Only the sinusoids of the DFT don't suffer this effect. All other sinusoids will be distorted and produce side lobes in the spectrum. When a sinusoid with frequency $w \neq w_k$ is not periodic in N samples we have a *glitch* in the spectrum on every N samples. This effect is demonstrated in Figure 2.4 and can be considered as an energy source in the spectrum.

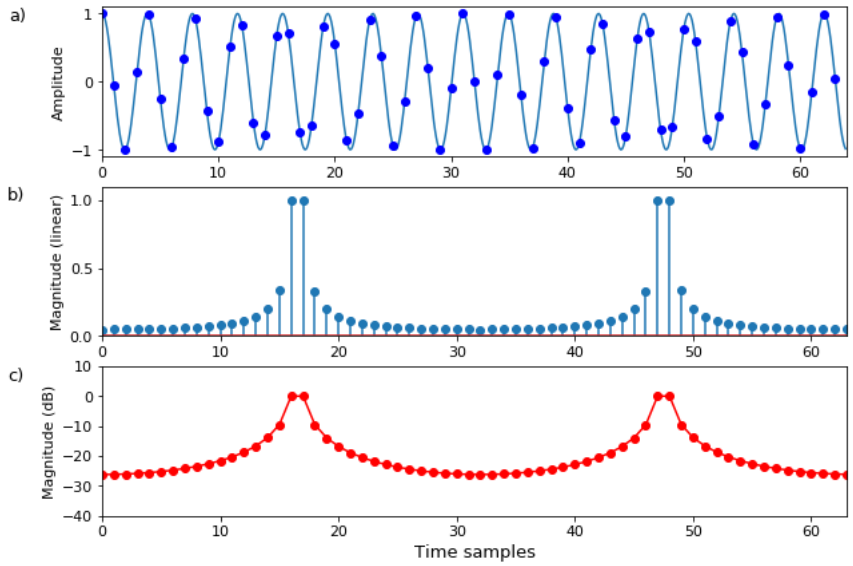

Figure 2.4: Sampled sinusoid at frequency $k = N/4 + 0.5$, $f = f_s/4 + f_s/2N$ and $N = 64$. a) Time waveform. b) Magnitude spectrum. c) Magnitude spectrum in decibels.

The DFT can also be interpreted as an operation of a digital filter on each frequency bin k, with input $x(n)$ and output $X(w_k)$ for the instant $n = N - 1$. Applying the sum of a geometric progression to the formulation (2.4), the frequency magnitude response of this filter is defined by

$$|X(w_k)| = \left| \frac{\sin((w_x - w_k)NT/2)}{\sin((w_x - w_k)T/2)} \right| \tag{2.13}$$

In all the remaining values of k, the frequency response is the same but with a circular shift to the left or right, so that the main lobe is centered in w_k. A technique that allows to reduce the *spectral leakage* in the computation of the spectrum of audio signals is running a segmented analysis of the signal. For this purpose, the input signal is divided into smaller sets of samples and multiplied by a window function which will soften the signal decay to 0 in both edges of the window. As consequence of this computation, the main spectral lobe becomes wider and the magnitude of the side lobes decreases in the DFT frequency response. Not applying a window to the signal is equivalent to the use of a rectangular window of size N, unless the signal is exactly periodic in N samples. In section 2.3 we will elaborate on some of the basic and more common window functions used in signal processing.

2.1.3 Properties of the DFT

The DFT can be efficiently computed with the *Fast Fourier Transform* (FFT) algorithm developed by Cooley-Tukey [4]. This algorithm computes a set of N dot products of length N and delivers a complexity of $O(N^2)$. When N is a power of 2 it becomes more efficient and its complexity is reduced to $O(NlogN)$. For this reason it's frequent to add zeros at the end of the digital signal prior to the computation of the DFT to optimize the length of the FFT. This process is called *zero-padding* and corresponds to a signal interpolation in the frequency domain. In the FFT algorithm the input signal is automatically *zero-padded* when the length of the DFT is larger than the length of the signal M $(N > M)$. It's important to note though that *zero-padding* a signal is not translated into more resolution in the frequency domain. To increase the resolution more samples are required, which is only achieved by increasing the length of DFT and removing the *zero-padding*.

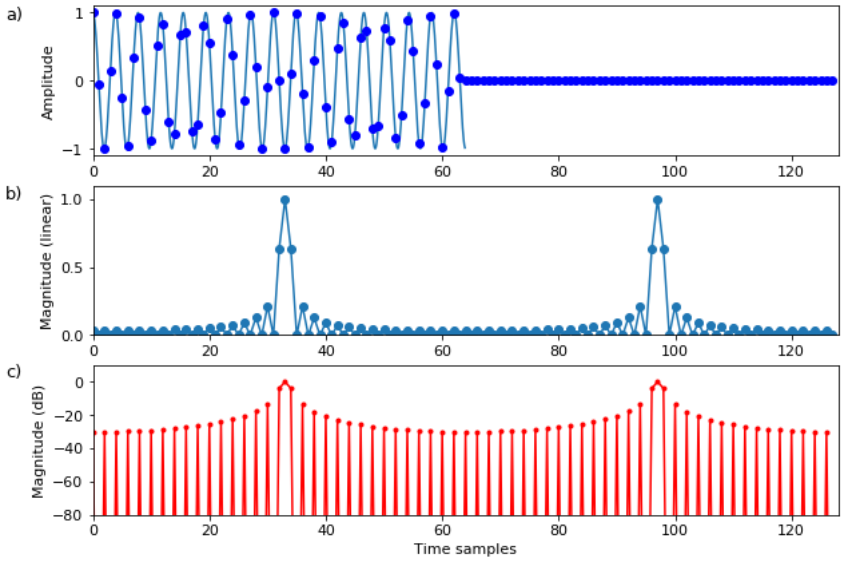

Figure 2.5: Zero-padded sinusoid at frequency $f = f_s/4 + f_s/2N$, $N = 64$ and zero-padding factor of 2. a) Time waveform. b) Magnitude spectrum. c) Magnitude spectrum in decibels.

Another technique used to improve the DFT computation is to remove the phase offset by *zero-phasing* the input signal. This is achieved by circularly shifting x to t=0 and allows to keep the magnitude spectrum intact. Let's consider the shift property of the DFT where a time shift - or delay - of Δ samples, will translate into a phase offset $e^{-jw_k\Delta}$:

$$x(n-\Delta) \longleftrightarrow e^{-jw_k\Delta}X(w_k) \tag{2.14}$$

keeping the magnitude spectrum unchanged

$$|X(w_k)| = \left|e^{-jw_k\Delta}X(w_k)\right|$$

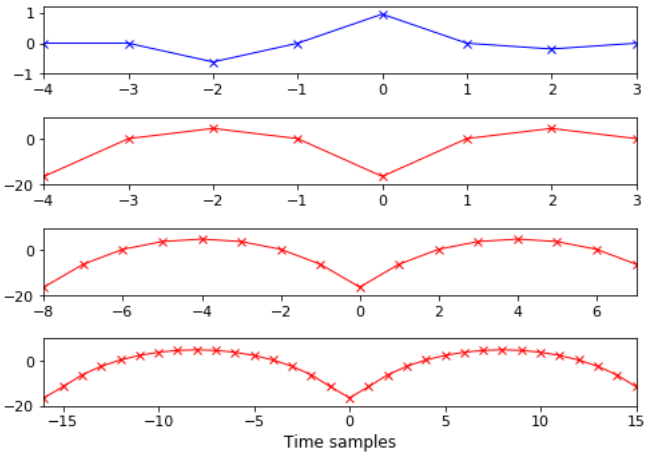

Figure 2.6: Interpolation of the magnitude spectrum of an input signal with M=8 for several padding factors. From top to bottom: input signal, N=8 (no zero-padding), N=16 and N=32.

The expression $e^{-jw_k\Delta}$ is called linear phase term because its phase is a linear function of frequency $\angle e^{-jw_k\Delta} = -\Delta w_k$. This property is relevant for the computation of the spectrum because the magnitude spectrum of any real and even signal is even, with imaginary part zero and linear phase between 0 and $n\pi$, which is indeed a zero-phase signal. So zero-phasing a signal means basically turning our real signal even. The simmetry properties of the DFT can be summarized as follows:

$x(n)$ is real $\longleftrightarrow$ $\Re\{X(k)\}$ is even, $\Im\{X(k)\}$ is odd, $|X(k)|$ is even, $\angle X(k)$ is odd
$x(n)$ is real and even $\longleftrightarrow$ $\Re\{X(k)\}$ is even, $\Im\{X(k)\} = 0$, $|X(k)|$ is even, $\angle X(k) = n\pi$

This way it is possible to discard the negative part of the spectrum for real signals and be able to recover the original signal from the positve half of the spectrum, ranging from 0 Hz to $fs/2$ Hz.

2.2 Short-time Fourier Transform

As discussed previously in this chapter, the DFT represents a discrete signal in the frequency domain. However, in the processing of real time varying acoustic signals is also necessary to determine the frequency distribution over the time. One way to achieve this is by computing the FFT of a windowed signal, segmented into independent time frames, and apply them a sliding window function with $N > M$. This approach is refered as *Short-time Fourier Transform* (STFT) and can be formally defined by

$$X_l(k) = \sum_{n=-N/2}^{N/2-1} x(n + lH)w(n)e^{-j2\pi kn/N}, \qquad l = 0, 1, 2... \tag{2.15}$$

where $w(n)$ is the window function, l is the frame number and H is the hop-size, which correponds

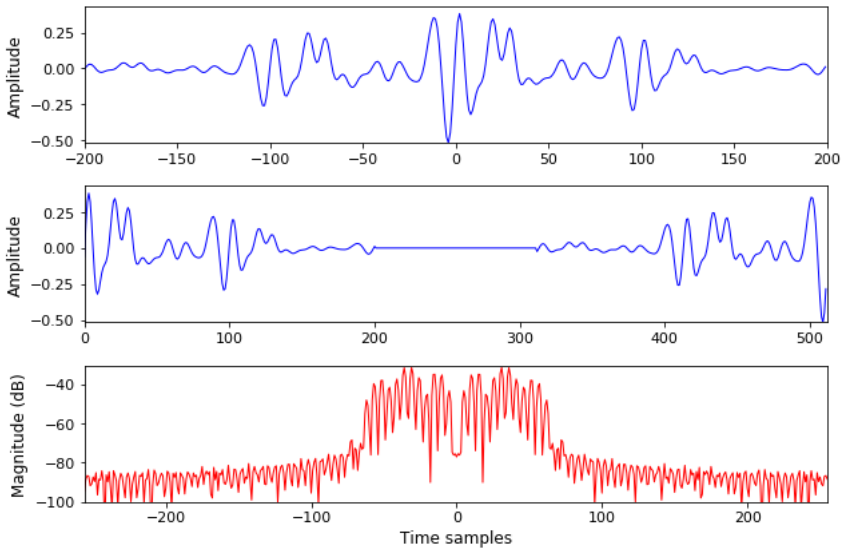

Figure 2.7: Zero-phase applied to an input signal with M=401. From top to bottom: input signal, zero-centered signal with N=512 and magnitude spectrum of the zero-phased signal.

to the number of sliding samples of the window between consecutive frames. From the definition (2.15), to a zero-phased input signal x is applied an also zero-phased window function w and computed its FFT. These steps are repeated for each time frame along the sliding window with step H, until all samples of x are analysed (Figure 2.8). The window overlapping factor for most of the functions take values in general between 25% and 50% in order to ensure that all the samples of x are processed. Figure 2.9 demostrates the windows overlapping effect. For a Blackman window, for example, we see that the maximum overlapping that allows all the samples of the input signal to be captured by windowing the signal is 25%. The wider is the window's mainloab, the smallest is the maximum allowed overlapping.

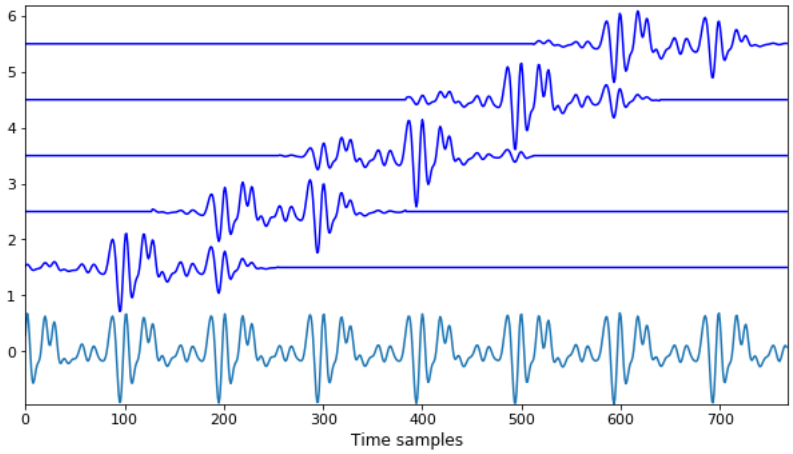

Figure 2.8: Fast Fourier Transform computation for H=128 and M=256.

Let's see what happens when a sinusoidal signal x is multiplied by a window w. Let's consider

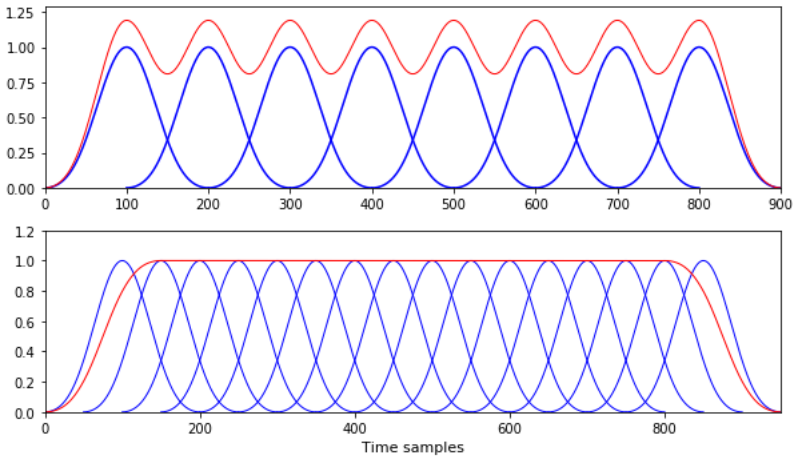

Figure 2.9: Overlapping on the Blackman window. On top: M=201 and H=100 (50% overlapping). On the bottom: H=50 (25% overlapping).

$x(n)$ is defined by

$$x(n) = A_0 \cos(2\pi k_0 n/N) = \frac{A_0}{2} e^{j2\pi k_0 n/N} + \frac{A_0}{2} e^{-j2\pi k_0 n/N} \tag{2.16}$$

Then the DFT of the windowed signal comes as

$$X(k) = \sum_{n=-N/2}^{N/2-1} w(n)x(n)e^{-j2\pi n/k} \tag{2.17}$$

Replacing (2.16) in (2.17) we finally can rewrite the spectrogram as

$$X(k) = \frac{A_0}{2} W(k - k_0) + \frac{A_0}{2} W(k + k_0) \tag{2.18}$$

where W is the DFT of the window function. Thus, from (2.18) we conclude that the DFT of a windowed signal is equivalent to the DFT of the window shifted to the frequencies of the signal $x(n)$. This is a property of the convolution theorem that states that multiplying two signals in the frequency domain corresponds to its convolution in the time domain. This effect is illustrated in Figure 2.10.

2.3 Window functions

The windowing technique used in the computation of the DFT determines the trade-off between temporal and frequency resolution which affects the smoothness of the spectrum and the detectability of the frequency peaks. The choice of the window function depends heavily on the size of the window's main lobe - which refers to number of samples in the main lobe - and its relation to

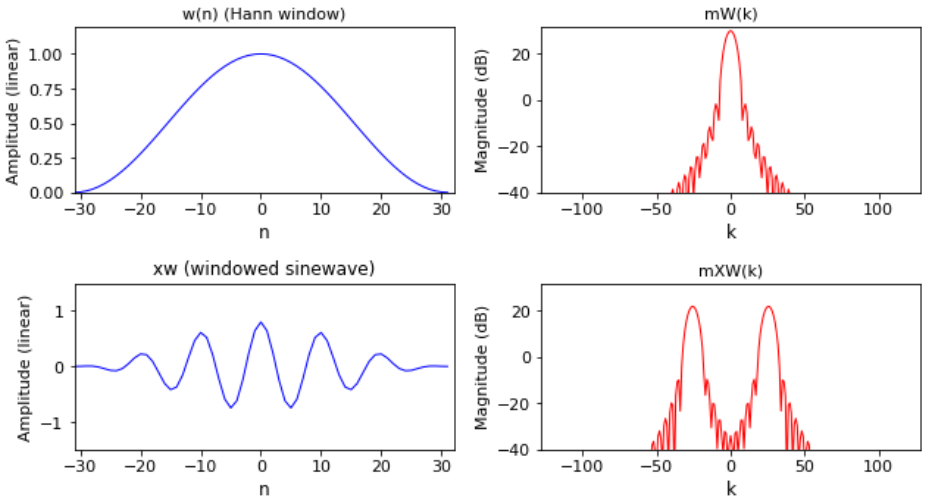

Figure 2.10: On top: Hann window and its DFT. On the bottom: windowed sinewave and its magnitude spectrum.

the side lobes amplitude. Windows with a narrower main lobe have a better frequency resolution but tend to have higher side lobes which is a source of *cross-talk* between the channels of the FFT. The simplest window is the rectangular window. However its efficiency for the audio signal processing purposes is low because of the low attenuation of its side lobes. For this reason there are other more efficient windows that have larger main lobes but allow a better time resolution, which will be discussed next: Hamming window, Hann window and Balckman-Harris windows.

To resolve two sinusoids of frequencies f_2 and f_1, separated by $\Delta f = f_2 - f_1$, we need two clearly discernible main lobes. In order to achieve this, we require a main lobe bandwidth B_f such that $B_f \leq \Delta f$. From (2.8) we can define B_f as

$$B_f = f_s \frac{K}{M} \tag{2.19}$$

where K is the main lobe bandwidth in bins and M is the window's length. Thus, we conclude the window should verify

$$M \leq K \frac{f_s}{\Delta f} \tag{2.20}$$

From (2.20) we see that the frequency resolution increases with the size of the window M. However it's interesting to notice that the number of main lobe samples is a feature of each window family and is not affected by the length of the window M. When using *zero-padding* for the computation the window's spectrum however ($N = \tau M$) there is an increase in the main lobe width by a scale factor of τ, which actually decreases the frequency resolution. On the other hand, has we will see, the main lobe should be narrow enough to resolve adjacent peaks but not narrower than the necessary in order to maximize time resolution in the STFT.

2.3.1 Rectangular window

The rectangular window w_R can be defined as

$$w_R(n) = \begin{cases} 1, & -\frac{M-1}{2} \leq n \leq \frac{M-1}{2} \\ 0, & \text{otherwise} \end{cases}$$

where M is the window's length, here considered odd for simplification. In the frequency domain the rectangular window comes as

$$\begin{aligned} W_R(w) = DTFT_w(w_R) &= \sum_{n=-\infty}^{\infty} w_R(n) e^{-jwn}, \qquad w \in [-\pi, \pi] \\ &= \sum_{n=-(M-1)/2}^{(M-1)/2} e^{-jwn} = \frac{e^{jw(M-1)/2} - e^{-jw(M+1)/2}}{1 - e^{-jw}} \end{aligned} \tag{2.21}$$

Applying the closed form for a geometric series

$$\sum_{n=L}^{U} z^n = \frac{z^L - z^{U+1}}{1 - z} \tag{2.22}$$

(2.21) comes as

$$W_R(w) = \frac{\sin(Mw/2)}{\sin(w/2)} = M \cdot asinc_M(w) \tag{2.23}$$

where $asinc_M(w)$ denotes the aliased sinc function, also know as Dirichlet function ou periodic sinc, defined as

$$asinc_M(w) = \frac{\sin(Mw/2)}{M \cdot \sin(w/2)} \tag{2.24}$$

The term *aliased sinc function* refers to the fact that it is obtained by sampling the length-τ continuous time rectangular window, which has Fourier transform $sinc(x) = \sin(\pi x)/\pi x$ in intervals of T seconds, which corresponds to aliasing in the frequency domain on $[0, 1/T]$ Hz. As the sampling rate goes to infinity, the aliased sinc function (2.24) approaches the sinc function.

$$\lim_{\tau \to 0} asinc_M(wT) = sinc(\tau f), \qquad MT = \tau \tag{2.25}$$

$$asinc_M(w) \approx \frac{\sin(\pi f M)}{M \pi f} = sinc(fM) \tag{2.26}$$

which has zero crossings at integer multiples of

$$\Omega_M = \frac{2\pi}{M} \tag{2.27}$$

The normalized rectangular window is illustrated in Figure 2.11, both in time and frequency domains. The width of the main lobe for the rectangular window is $2\Omega_M = 2 \cdot 2\pi/M$ radians, with a maximum side lobe attenuation of -13 dB, width $\Omega_M = 2\pi/M$ nd a -6 dB roll-off/octave.

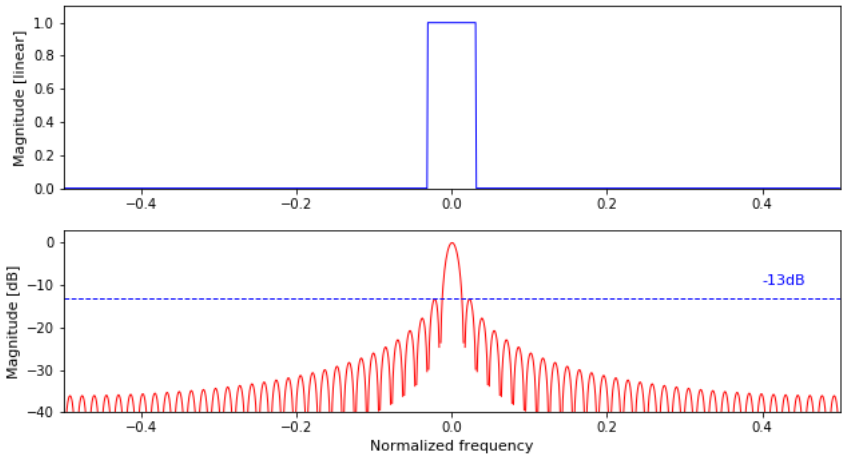

Figure 2.11: Zero-phased rectangular window with M=64 and N=1024. On top: In time domain. On the bottom: Magnitude spectrum.

2.3.2 Hamming windows

The generalized Hamming window family is constructed by multiplying the rectangular window by one period of a cosine, which produces side lobes with lower amplitude at the cost of a wider main lobe. The members of this group described here are the Hamming and Hann windows. These windows can be defined as a function of the already known rectangular window W_R and thus correspond to the sum of three $sinc$ functions in the frequency domain

$$W_H(w) = \alpha W_R(w) + \beta W_R(w - \Omega_M) + \beta W_R(w + \Omega_M), \qquad w \in [-\pi, \pi] \tag{2.28}$$

After applying the DFT shift theorem, the inverse from (2.28) comes as

$$w_H(n) = w_R(n) \left[\alpha + 2\beta \cos\left(\Omega_M n\right)\right], n \in \mathbb{Z} \tag{2.29}$$

The parameters α and β are assumed to be positive and define the windows of this family. The Hann or Hanning window plotted in Figure 2.12 has $\alpha = 1/2$ and $\beta = 1/4$, with a main lobe width of $4\Omega_M$, side lobe attenuation of -31.5 dB and aproximately -18 dB roll-off/octave. The Hamming window, on the other side, is generally defined by $\alpha = 25/46 \approx 0.54$ and $\beta = (1 - \alpha)/2 \approx 0.23$, and is represented in Figure 2.13. In this case the cosine is raised so that the negative peaks stay above zero, producing a descontinuity in the amplitude at the edges. This causes a slower roll-off rate in the side lobes (-6 dB) and its attenuation can be reduced to approximately -43 dB.

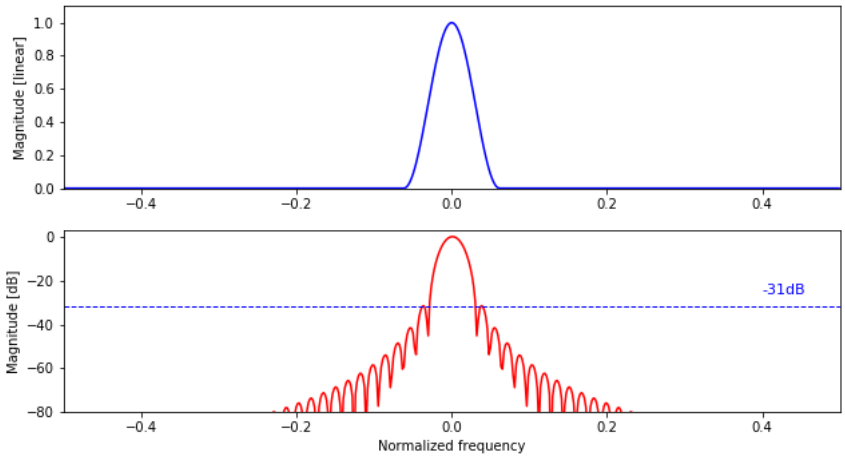

Figure 2.12: Hanning window with M=64 and N=512. On top: In time domain. On the bottom: Magnitude spectrum in decibels.

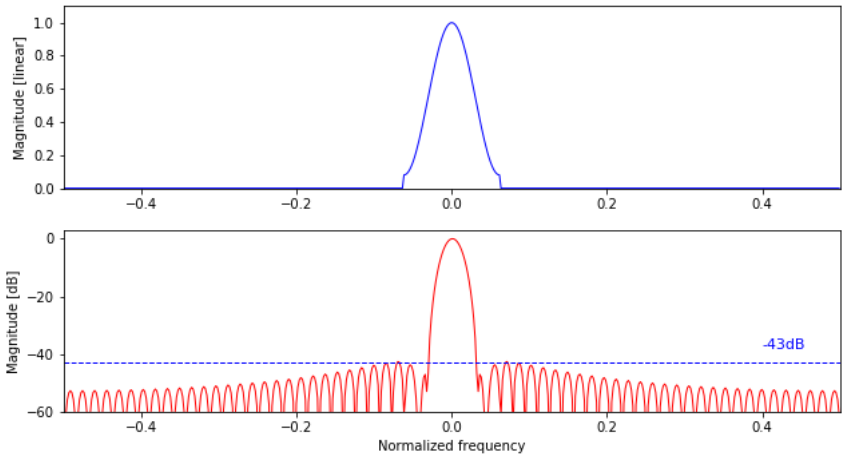

Figure 2.13: Hamming window with M=64 and N=512. a) In time domain. b) Magnitude spectrum in decibels.

2.3.3 Blackman-Harris windows

The windows from the Blackman-Harris family are a generalization of the Hamming windows and result on adding more shifted $sinc$ functions to (2.28).

$$w_{BH}(n) = w_R(n) \sum_{l=0}^{L-1} \alpha_l \cos(l\Omega_M n), \qquad n \in [-(M-1)/2, (M-1)/2] \tag{2.30}$$

and its transform is given by

$$W_{BH}(w) = \sum_{k=-(L-1)}^{L-1} \alpha_k W_R(w + k\Omega_M) \tag{2.31}$$

In (2.31) we verify that for L=1 we obtain a rectangular window, when L=2 we get the Hamming windows and for L=3 we have the Blackman window family which is defined as

$$w_B(n) = w_R(n)[\alpha_0 + \alpha_1 \cos(\Omega_M n) + \alpha_2 \cos(2\Omega_M n)] \tag{2.32}$$

The Hamming windows have only two degrees of freedom: one used to normalize the magnitude of the window, while the second is either used to maximize the roll-off rate (Hann window) or increase the attenuation of the side lobes (Hamming window). Now the extra degrees of freedom of the Blackman windows will allow us to optimize these features and define subtypes within the family. For example, the classic Blackman represented in Figure 2.14, has three degrees of freedom with $\alpha_0 = 0.42$, $\alpha_1 = 0.5$ and $\alpha_2 = 0.08$, which results in a side lobe attenuation of at least -58 dB and a roll-off rate of around -18 dB per octave. On the other hand, the Blackman-Harris window plotted in Figure 2.15, can maximize the attenuation of the side lobes on -92 dB with four degrees of freedom and parameterization $\alpha_0 = 0.35875$, $\alpha_1 = 0.48829$ and $\alpha_2 = 0.14128$ and $\alpha_3 = 0.01168$.

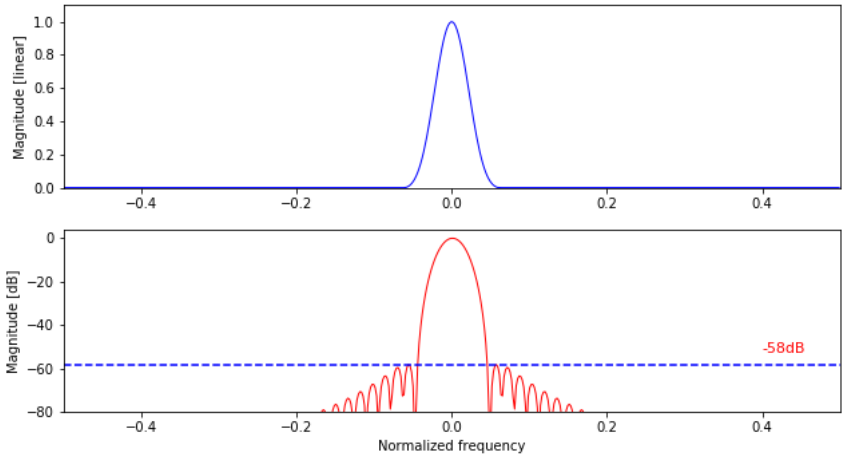

Figure 2.14: Classic Blackman window with M=64 and N=512. On top: In time domain. On the bottom: Magnitude spectrum in decibels.

The drawback of improving the side lobe attenuation is that the width of the main lobe also increases. The sum of more $sinc$ functions increases both the side lobe width and the degrees of freedom, as shown in Figure 2.16. This means that if on one side we gain frequency resolution, we loose time resolution because of the increase in the number of processed samples in each time frame. The Table 2.1 sintesizes the main features of the windows discussed above.

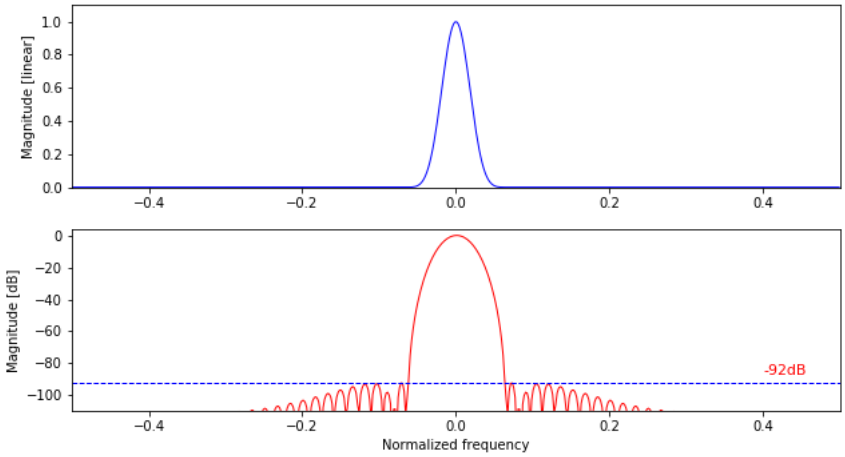

Figure 2.15: Blackman-Harris window with M=64 and N=512. On top: In time domain. On the bottom: Magnitude spectrum in decibels.

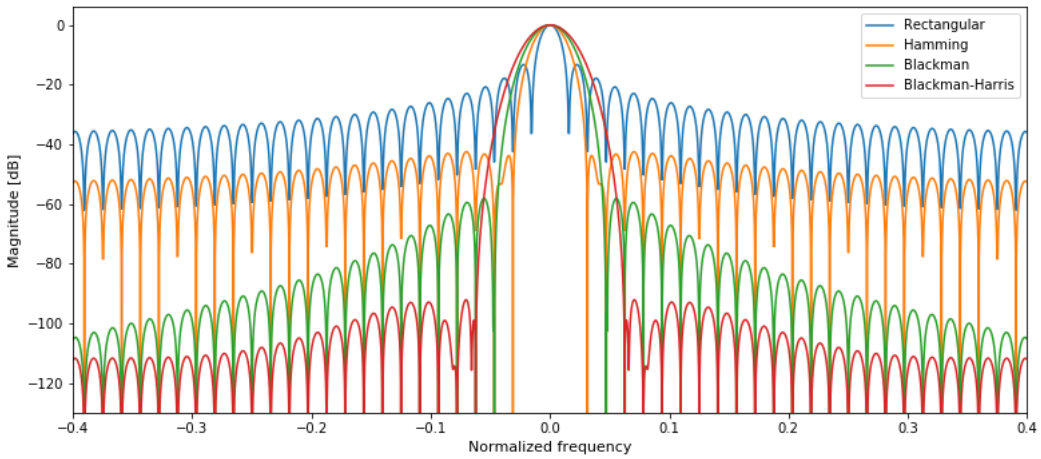

Figure 2.16: Comparison of the magnitude spectrum of the windows: Rectangular, Hamming, Blackman and Blackman-Harris.

	Rectangular	*Hann*	*Hamming*	*Blackman*	*Blackman-Harris*
Main lobe width	$2\Omega_M$ 2 bins	$4\Omega_M$ 4 bins	$4\Omega_M$ 4 bins	$6\Omega_M$ 6 bins	$8\Omega_M$ 8 bins
Side lobe attenuation	-13 dB	-31 dB	-43 dB	-58 dB	-92 dB
Side lobe roll-off/octave	-6 dB	-18 dB	-6 dB	-18 dB	-
Degrees of freedom	1	2	3	4	5
Max overlapping factor	100%	50%	50%	25%	20%

Table 2.1: Comparison of the windows Rectangular, Hamming, Blackman and Blackman-Harris.

Chapter 3

Music Theory

3.1 Harmonic series

Blowing air through a wind instrument or plucking the string of guitar produces a vibration that is perceived as sound. The pitch we hear depends on the frequency of the vibration which in turn is determined by the air volume in the instrument or the length of the string. The string however does not vibrate at only one frequency, the one of the perceived fundamental. Let's consider a freely vibrating body which can be described by the wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{3.1}$$

where $u(\mathbf{x}, t)$ is the displacement from rest at time t and at point $\mathbf{x} \in \mathbb{R}^n$, and ∇^2 is the Laplacian operator defined as:

$$\nabla^2 = \begin{cases} \dfrac{\partial^2 u}{\partial x^2}, & \text{on } \mathbb{R} \\ \\ \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}, & \text{on } \mathbb{R}^2 \end{cases} \tag{3.2}$$

Considering a vibrating string fixed at both ends, with motion perpendicular to the string - producing a transverse wave - and a displacement such that its slope at any point along the string length is small $\sin(\theta) \approx \theta$, then the string motion can be described by the wave equation (3.1) with boundary conditions

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = c^2 \dfrac{\partial^2 u}{\partial x^2} & \\ u(0,t) = u(L,t) = 0 & \text{(boundary conditions)} \\ u(x,0) = f(x), \qquad u_t(x,0) = g(x) & \text{(initial conditions)} \end{cases} \tag{3.3}$$

In order to solve the wave equation for a vibrating string as defined in (3.3), let's assume a

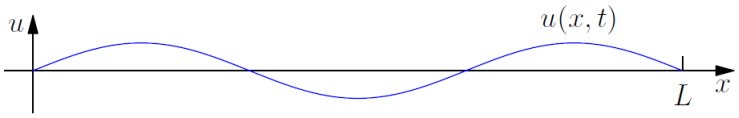

Figure 3.1: Transverse wave produced by a vibrating string.

separation of variables $u(x,t) = X(x)T(t)$ and apply the boundary conditions

$$0 = \frac{\partial^2 u}{\partial t^2} - c^2\nabla^2 u = XT'' - c^2 T\nabla^2 X \tag{3.4}$$

The equality (3.4) is independent of x and t, repectively, on the left and right sides of the equation, which means the equality takes a constant value λ

$$\frac{1}{c^2}\frac{T''}{T} = \frac{\nabla^2 X}{X} = \lambda \tag{3.5}$$

$$\begin{cases} T'' - c^2 T\lambda = 0 \\ \nabla^2 X - \lambda X = 0 \end{cases} \Leftrightarrow \begin{cases} T(t) = A\cos(\sqrt{\lambda}ct) + \phi, & A, \phi \in \mathbb{R} \\ \nabla^2 X = \lambda X \end{cases} \tag{3.6}$$

and by linearity we conclude the string motion is given by

$$u(x,t) = \sum_{n=1}^{\infty} A_n \cos(\sqrt{\lambda_n}ct + \phi_n) f_n(x) \tag{3.7}$$

where λ_n are eigenvalues and f_n eigenfunctions of ∇^2. So the string motion corresponds to the sum of time varying sinusoids with frequencies $\omega_n = \sqrt{\lambda_n}c$, where the smallest λ_n corresponds to the fundamental tone while other modes correspond to the upper partials. Solving the eigenvalues problem $\nabla^2 f = \lambda f$ for 1-D with the boundary conditions, it comes

$$\frac{d^2 f}{dx^2} = \lambda f \qquad \Longrightarrow \qquad f(x) = C_1 \sin(\sqrt{\lambda}x) + C_2 \cos(\sqrt{\lambda}x) \tag{3.8}$$

$$\begin{cases} f(0) = 0 \Longrightarrow C_2 = 0: & f(x) = C_1 \sin(\sqrt{\lambda}x) \\ f(L) = 0: & C_1 \sin(\sqrt{\lambda}x) = 0 \quad \Longrightarrow \quad \sqrt{\lambda}L = n\pi, \quad n = 0, 1, 2, ... \end{cases} \tag{3.9}$$

The frequencies of the vibrational modes are then given by

$$\lambda_n = \left(\frac{n\pi}{L}\right)^2 \qquad \Longrightarrow \qquad \omega_n = \sqrt{\lambda_n}c = \frac{n\pi c}{L}, \qquad n = 1, 2, 3, ... \tag{3.10}$$

and the eigenfunctions are consequently defined as

$$f_n(x) = \sin\left(\frac{n\pi x}{L}\right) \tag{3.11}$$

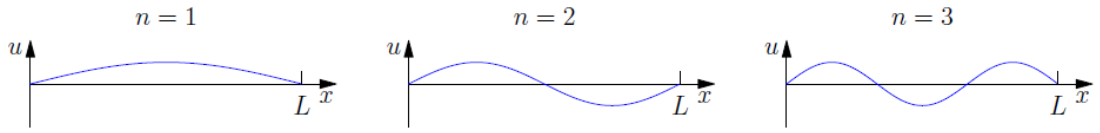

Figure 3.2: Vibrating modes in a plucked string. From left to right: fundamental, 2nd and 3rd harmonic standing waves.

Replacing (3.11) in (3.7), the string motion comes finally as

$$u(x,t) = \sum_{n=1}^{\infty} \left[a_n \cos(\sqrt{\lambda_n}ct) + b_n \sin\left(\sqrt{\lambda_n}ct\right)\right] C \sin\left(\frac{n\pi x}{L}\right) \tag{3.12}$$

So what we hear when we pluck a string is a superposition of pure tones at discrete frequencies $\omega_n = n\pi c/L = n\omega_1$, which are integer multiples of the fundamental frequency ω_1 and define the harmonic series. The sound perception though is here independent of the phase ϕ. Music instruments however, in general do not produce perfect multiples of the fundamental frequency, so it is more correct to talk about partials instead of harmonics. Some instruments like the drums or the piano are examples of naturally inharmonic instruments, as we will see in the next chapter.

3.2 Pitch and frequency

The twelve musical notes traditionally used in western music are widely defined in terms of the Helmholtz notation, consisting on the ordered sequence of letters A (la) to G (so), followed by the respective octave. For example, A4 corresponds to the A on the 4th octave above the pitch A0 (27.5 Hz). The standard frequency of A4 has varied according to the period of the history of music and was finally defined in 1939 as 440 Hz. In the case of musical transcription, the mapping of frequency - computed deterministically from the spectrogram - into musical notes, is achieved by computing the respective pitch number. The term pitch refers to a perceptive human quality - such as the duration, amplitude or timbre - that classifies the sounds depending on their height, and allows organizing the musical notes into ordered sequences, also called musical scales. The human auditive perception of musical distance between two notes is logarithmic in frequency, so the pitch p_n corresponding to the frequency f_n in the tempered scale - which is the most commonly used tuning system in western europe -, with reference to the lowest pitch of a piano keyboard (A0 = 27.5 Hz), is defined as

$$p_n = 12 \cdot \log_2\left(\frac{f_n}{f_{A0}}\right) \tag{3.13}$$

This way, the musical interval $\delta p = p_1 - p_2$ corresponds to a frequency ratio of f_1/f_2 which defines a geometric series in the frequency

$$p_0, p_0 + \delta p, p_0 + 2\delta p, p_0 + 3\delta, ... \Longleftrightarrow f_0, \alpha f_0, \alpha^2 f_0, \alpha^3 f_0, ... \tag{3.14}$$

It's common practice to state musical intervals in cents by equally dividing the semitone interval between consecutive notes into 100 parts

$$f^{p_n/100} = (\sqrt[12]{2})^{p_n/100} \tag{3.15}$$

$$p_{nc} = 1200 \cdot \log_2 \left(\frac{f_n}{f_{A0}} \right) \tag{3.16}$$

The cents notation provides a usefull way to compare intervals in different temperaments and to decide wheter those differences are musically significant. Usually the criterium for acceptable tuning accuracy is set at 5 cents which corresponds to the *just-noticeable difference* (JND) for the human auditory system.

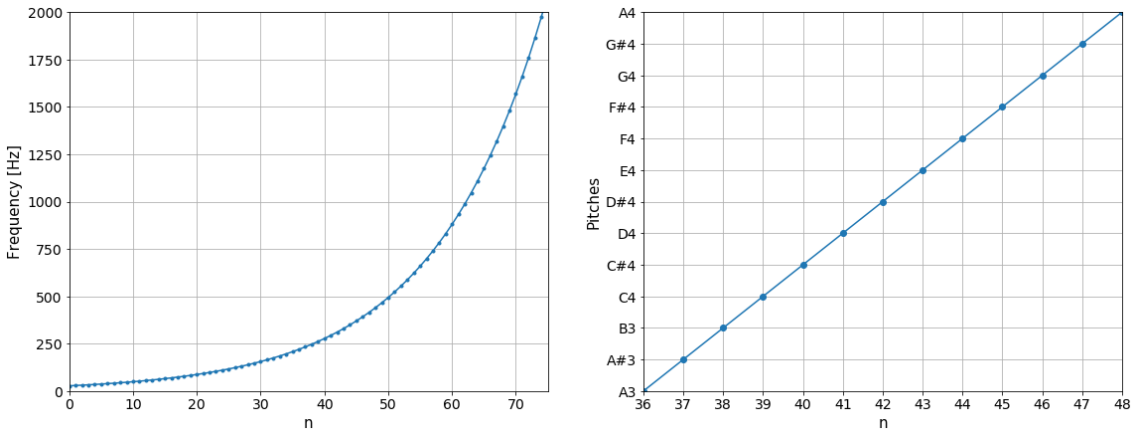

Figure 3.3: On the left: Frequencies of the tempered scale. On the right: Pitches for the A4 chromatic scale with reference to A0 (27.5 Hz).

3.3 Tuning systems

In the VI century B.C., Pythagoras discovered that when two similar strings under the same tension are plucked together, they'd give a particularly consonant sound if their lenghts are the ratio of two small integers. At this time the term consonance referred essentially to the relationship between pitches in a melodic context, so there was no harmony in the modern sense of simultaneously sounding notes. Later on the XIX century, a theory that explained the phenomena of consonance was proposed by Helmholtz [2] and experimentally verified by Plomp and Levelt [1], to be related to the beating effect and the matching partials of the sounds rather than the ratio of the harmonics. However for Pythagoras, the octave (2:1) and the perfect fifth (3:2) intervals were

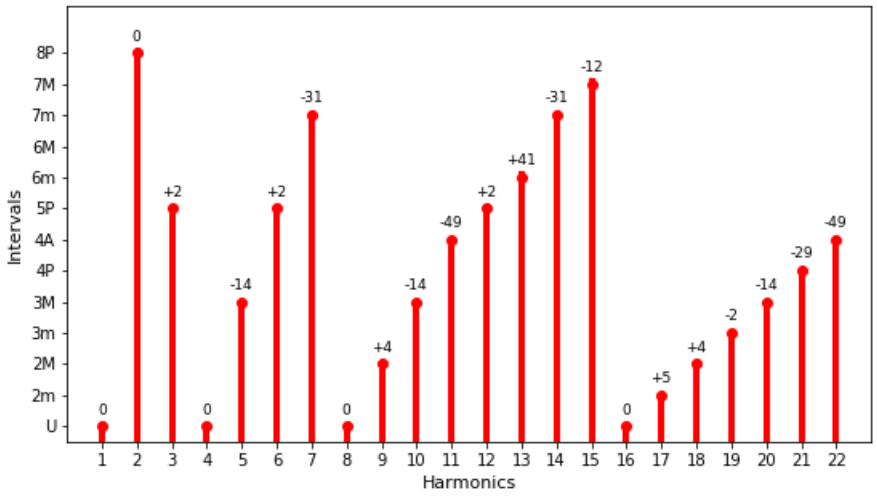

Figure 3.4: Comparison between the first 22 harmonics and the 12 tones of the tempered scale. The labels correspond to the error in cents.

inspirationally balanced in it's ratios simplicity, as should be the laws of nature, so he defined a scale using a tuning sequence of fifths, which can also be referred to as the *cycle of fifths*, as depicted in Figure 3.5. This was the first known example of a law of nature ruled by the arithmetic of integers, and greatly influenced the intellectual development of his followers.

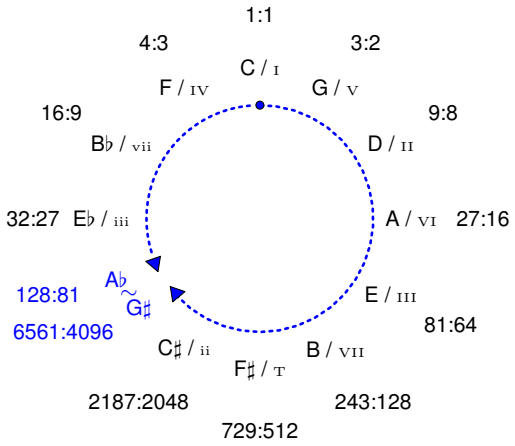

Figure 3.5: The cycle of fifths as proposed by Pythagoras.

This tunning system can be extended to a twelve tone scale by moving up eight times and four times down a 3:2 ratio from the initial note until we reach the same pitch. This doesn't happen however. As observed in the fifth cycle of Figure 3.5, the frequencies of A♭ and G♯, also called enharmonic notes, do not match. In fact they differ from each other on a significantly audible ratio designated as Pythagorian comma

$$\frac{6561/4096}{128/81} = \frac{3^{12}}{2^{19}} \approx 23.46 \text{ cents} \tag{3.17}$$

One of the issues of the pythagorian tuning system, is that only the perfect fifths and fourths were highly consonant. This imposed a limitation on the complexity of the music composition. So during the Barroque period, with the expansion of contrapuntal composition and the use of thirds and sixths, the major triads were adjusted in order to be more consonant with frequency ratios of 4:5:6. This is the idea at the base of the so called *just intonation* which referes to a tuning system in which - unlike the pythagorian tuning - the frequencies of the notes are related by ratios of small integers. In reallity these ratios do not perfectly correspond to the natural harmonics, but are actually closer to the harmonic series than the previous tuning systems.

The problem of both the just intonation and the pythagorian tuning is that it is designed to work on a particular key signature, so in order to allow the modulation to a different key - which is a common practice on western music - many of the tones used would have to be discarted and more tones would have to be provided. From the instrument point of view, it would translate into more strings, frets or holes to accomodate all the possible keys and their combinations, which is impractical. The solution for this problem came with the tempered tunning system, that allowed to compromise the pure intervals to meet other more practical requirements. The widely adopted tuning system in western music - the equal temperament - divides the octave into twelve equal semitones with constant ratio $\sqrt[12]{2} \approx 1.06$. Thus the frequency $f(n)$, corresponding to the n-th semitone in relation to a reference frequency f_0, comes as

$$f(n) = f_0 \sqrt[12]{2}^n = f_0 2^{n/12} \tag{3.18}$$

Tables 3.1 and 3.2 illustrate the comparison for the A4 chromatic scale in three tuning systems: tempered scale, just intonation and pythagorian tuning.

				Tempered scale			Just tuning			
Note	Interval	HF	Harmonics	Frequency [Hz]	Cents	Ratio [f_n/f_0]	Frequency [Hz]	Cents	Ratio [f_n/f_0]	Error [cents]
A4	Unison	I	1	440.00	0.00	1.00	440.00	0.00	1:1	0.00
A♯4	Minor second	ii	17	466.16	100.00	1.07	469.33	111.73	16:15	11.73
B4	Major second	II	9,18	493.88	200.00	1.12	495.00	203.91	9:8	3.91
C5	Minor third	iii	19	523.25	300.00	1.20	528.00	315.64	6:5	15.64
C♯5	Major third	III	5, 10, 20	554.37	400.00	1.25	550.00	386.31	5:4	-13.69
D5	Perfect fourth	IV	21	587.33	500.00	1.33	586.67	498.04	4:3	-1.96
D♯5	Tritone	T	11, 22	622.25	600.00	1.40	616.00	582.51	7:5	-17.49
E5	Perfect fifth	V	3, 6, 12	659.26	700.00	1.50	660.00	701.96	3:2	1.96
F5	Minor sixth	vii	13	698.46	800.00	1.60	704.00	813.69	8:5	13.69
F♯5	Major sixth	VI	27	739.99	900.00	1.67	733.33	884.36	5:3	-15.64
G5	Minor seventh	vii	7, 14	783.99	1000.00	1.80	792.00	1017.60	9:5	17.60
G♯5	Major seventh	VII	15	830.61	1100.00	1.88	825.00	1088.27	15:8	-11.73
A5	Octave	I	2^n	880.00	1200.00	2.00	880.00	1200.00	2:1	0.00

Table 3.1: Comparison of the tempered scale with the just tuning.

				Tempered scale			Pythagorian scale			
Note	Interval	HF	Harmonics	Frequency [Hz]	Cents	Ratio [f_n/f_0]	Frequency [Hz]	Cents	Ratio [f_n/f_0]	Error [cents]
A4	Unison	I	1	440.00	0.00	1.00	440.00	0.00	1:1	0.00
A♯4	Minor second	ii	17	466.16	100.00	1.07	469.86	113.69	2187:2048	-13.69
B4	Major second	II	9,18	493.88	200.00	1.12	495.00	203.91	9:8	-3.91
C5	Minor third	iii	19	523.25	300.00	1.20	521.48	294.13	32:27	5.87
C♯5	Major third	III	5, 10, 20	554.37	400.00	1.25	556.88	407.82	81:64	-7.82
D5	Perfect fourth	IV	21	587.33	500.00	1.33	586.67	498.04	4:3	1.96
D♯5	Tritone	T	11, 22	622.25	600.00	1.40	626.48	611.73	729:512	-11.73
E5	Perfect fifth	V	3, 6, 12	659.26	700.00	1.50	660.00	701.96	3:2	-1.96
F5	Minor sixth	vii	13	698.46	800.00	1.60	704.79	815.64	6561:4096	-15.64
F♯5	Major sixth	VI	27	739.99	900.00	1.67	742.50	905.87	27:16	-5.87
G5	Minor seventh	vii	7, 14	783.99	1000.00	1.80	782.22	996.09	16:9	3.91
G♯5	Major seventh	VII	15	830.61	1100.00	1.88	835.31	1109.78	243:128	-9.78
A5	Octave	I	2^n	880.00	1200.00	2.00	892.01	1223.46	3^{12}:2^{18}	-23.46

Table 3.2: Comparison of the tempered scale with the Pythagorian scale.

Chapter 4

Algorithm

4.1 Overview

This chapter describes a simple algorithm for transcribing an excerpt of a piano recording, in this case, the three voice Fugue I of J. S. Bach. The signal processing tasks involve computing the log spectrogram of the input signal, extract the rhythmic information prior to any filtering, then remove the background noise from the spectrum, compute the fundamental frequencies spectrogram and finally detect the melodic contours. The algorithm described in the below sections was programmed in python and complemented with already available open source code [16] designed for music and signal processing applications.

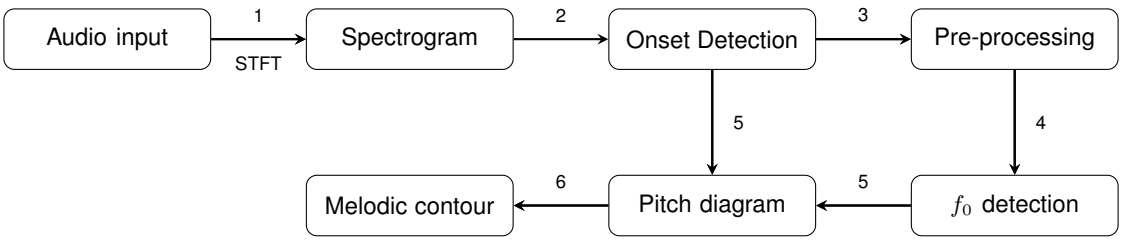

Figure 4.1: Architecture of the proposed transcription algorithm.

4.2 Spectrogram

The computation of the STFT produces two spectrograms based on complex numbers: one for the phase $\angle X(w_k)$ and another for the magnitude $|X(w_k)|$. As far as music sounds are concerned, the human hear is unsensitive to the phase shifts and for that reason the phase information was discarded and considered only the magnitude spectrum. The STFT is usually visualized in a 3 dimension diagram where the horizontal axis corresponds to the time, the vertical axis

corresponds to the frequency bins and the magnitude is defined by the color range of the diagram at each point.

For the computation of the spectrogram were considered WAV mono-channel signals sampled at 44100 Hz. Once read in python, the samples are converted into a floating array and normalized so that $x \in [-1, 1]$. In was used a Hamming window with M = 2^{13} = 8192, without zero-padding to allow higher frequency resolution and an overlap-factor of 25%, which is less than the 50% limit for this window but allows the increase of time resolution. Despite the roll-off of the side-lobes not overcoming -43 dB, in practise the Hamming window had a better performance that the Blackman-Harris windows when it comes to the compromise between time and frequency resolution. First, it's possible to balance the weight of the samples in the FFT computation and reduce the *cross-talk*, with an overlap-factor equivalent to that of the Blackman-Harris window. This allows to increase time resolution for the Hamming window, although at a higher computational cost. On the other hand, the frequency resolution is naturally better with a Hamming window because its main lobe is narrower. The size of the window was dimensioned so that the spectrogram would resolve the lowest frequency of the piano keyboard A0 (27.5 Hz) and Δf was defined as 21.53 Hz

$$\Delta f = \frac{k}{M} f_s = 21.53\ Hz, \qquad M = N = 2^{13} \tag{4.1}$$

This resolution proved to be enough for the current purposes, considering we are analysing tonal piano music where the lowest pitch is not expected to go below A♯2 (116.54 Hz), neither to be part of a chord with an interval lower than the third minor - the smallest consonant interval. This corresponds to have, in worst case scenario, at least an A♯2 (116.54 Hz) played at the same time as a D3 (146.83 Hz). The python code to compute the STFT is open source and available at [16]. The algorithm is described below.

1. Read the input signal x and convert it to a normalized floating point array.
2. Add $M/2$ zero-padding at the end and begining of the input signal x. Considering the window is not rectangular, this step allows to preserve the $M/2$ initial and final samples. Without this step, the FFT would be computed only for the total number of samples of x minus one window.
3. Read M samples of the signal into a buffer of size $N \geq M$, in case zero-padding is applied.
4. Multiply the signal by the normalized window w.
5. Zero-phase the windowed signal xw so that its phase is zero before computing the FFT.
6. Select only the positive half of the spectrum. As the FFT size is even, we need $(N + 1)/2$ samples to include the sample on t=0.

7. Move forward H samples on the input signal x and repeat the steps 3 to 5 until all the samples of the input signal are processed.

The magnitude spectrogram was then converted into logaritmic scale in order to correspond the frequency bins with the human perception of musical pitches, as shown in Figure 4.2.

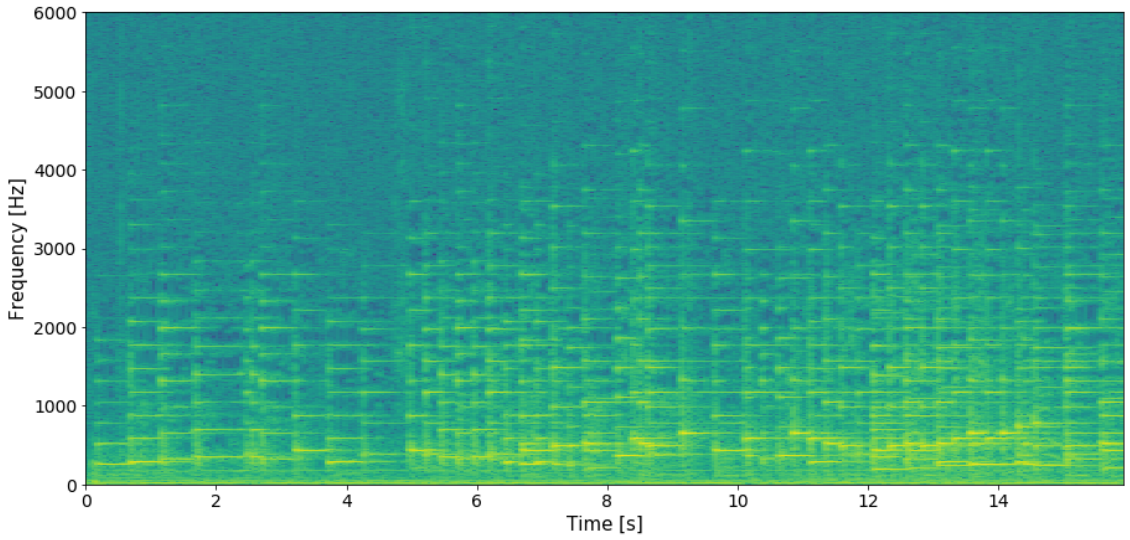

Figure 4.2: Log-spectrogram in decibels of the four first bars of the Fugue I in C Major from the *Well Tempered Clavier* BWV 846 (Book I) from J. S. Bach.

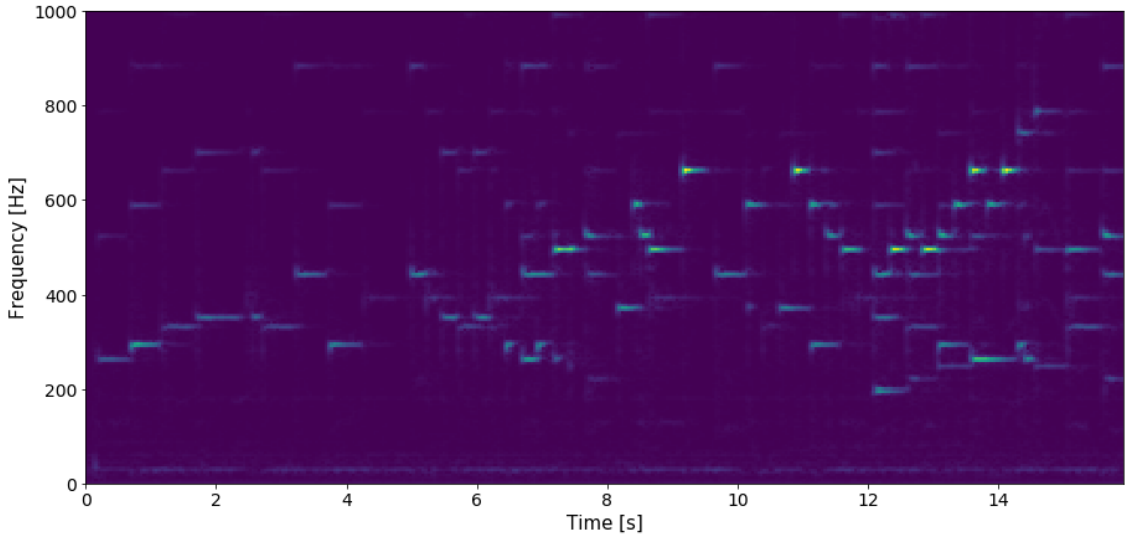

Figure 4.3: Log-spectogram of the input signal in linear units.

4.3 Rhythmic information extraction

The first step after computing the spectrogram is the extraction of the rhythmic information. This task requires the presence of all spectral information, including the higher harmonics region, and

for this reason it has to be performed prior to any spectral filtering. Rhythmic information extraction requires a vertical analysis of the spectogram and aims to determine the frames when a new musical note is played. This is achieved by computing a simple onset detection funcion (ODF), which determines the difference of energy between adjacent frames l on a specific frequency band [16]

$$ODF(l) = E(l) - E(l-1), \qquad l \geq 1 \tag{4.2}$$

where

$$E(l) = \sum_{l=0}^{N-1} x(l)^2 \tag{4.3}$$

Considering we are only interested in the onsets - when the energy increases - and not the offsets - when the energy decreases - we consider a half wave rectified ODF given by

$$ODF'(l) = \begin{cases} ODF(l), & \text{if } ODF(l) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

Three frequency bands were chosen for this analysis: the first one between 20 Hz and 200 Hz, the second between 201 Hz and 1000 Hz and the third ranging from 1001 Hz to 4000 Hz. The frequencies used to define the bands were choosed in order to comprise the frequencies produced by a piano keyboard, ranging from 27.5 Hz to 4186.01 Hz. The lower band allows the detection of lower notes, such as the bass, while the higher band provides information on notes that somehow have lower energy on the lower band - such as the missing fundamentals - but do have higher energy on the upper partials. We observe that the lower band in particular adds several false onsets, so the best results were obained by considering only the two highest frequency bands. We started by finding the common onsets, reducing the initial 140 value to 44 as shown in blue in Figure 4.4. Joining frequency bands means removing the duplicated onsets, multiplying the ODF' functions to discard null energy points and averaging their amplitude for both bands. When we have two onsets separated by only one frame, we chose the one with the highest amplitude, assuming that this is the lowest time resolution. In order to fill in the still missing onsets, we added the values above averge on both bands has demonstrated in Figure 4.4). This increased the onsets to 52, which is very close to the real value of 50. The final results are plotted in Figure 4.5. where we verify that we are still missing onsets (on frame 311, for example), and have as well false detections as in frame 48. Time onset, however, is a fundamental part of the audio transcription process followed here. Without accurate information on the rhythmic events the calculation of the averaged f_0 frequencies in the next section will be compromised. For this reason, the onsets were ultimately validated by inspection.

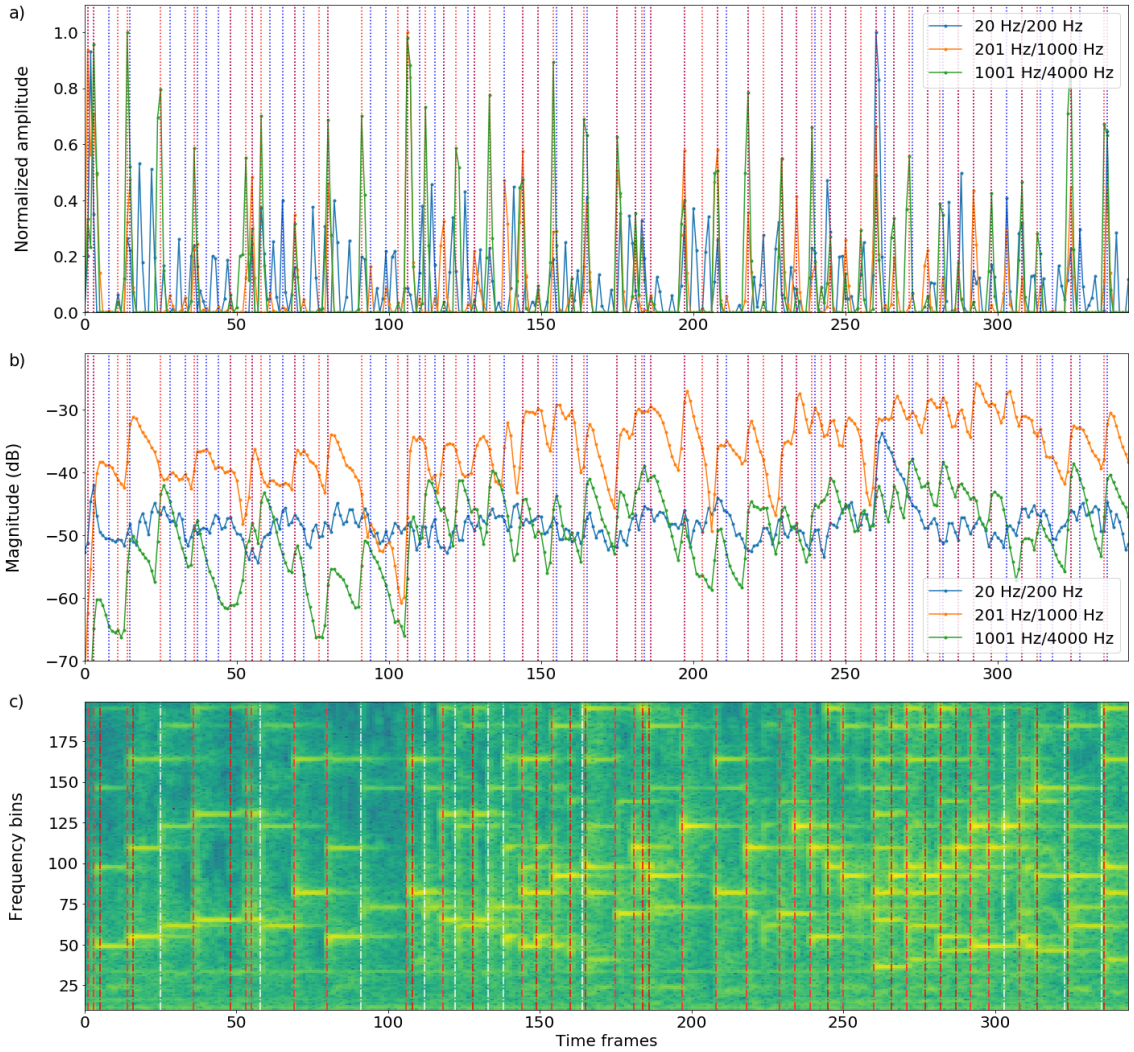

Figure 4.4: Onset detection for three frequency bands: 20 Hz to 200 Hz (blue), 201 Hz to 1000 Hz (orange) and 1001 Hz to 4000 Hz (green). a) Onset detection functions. b) Energy envelope. c) Log-spectrogram of the input signal. The vertical white lines correspond to the 37 onsets common to the two higher frequency bands and the red vertical lines are the joined peaks above average (44).

4.4 Melodic information extraction

4.4.1 Pre-processing

To estimate the background spectrum it was used a simplified method similar to the spectral whitening referred in [12]. This step is possible because we are transcribing a solo instrument and so the timbre information can be discarded. This filtering consisted on subtracting the average of the spectrum μ_k in an octave interval with step 100 bins, divide it by the standard deviation σ_k and filter out the negative amplitudes in the spectrum.

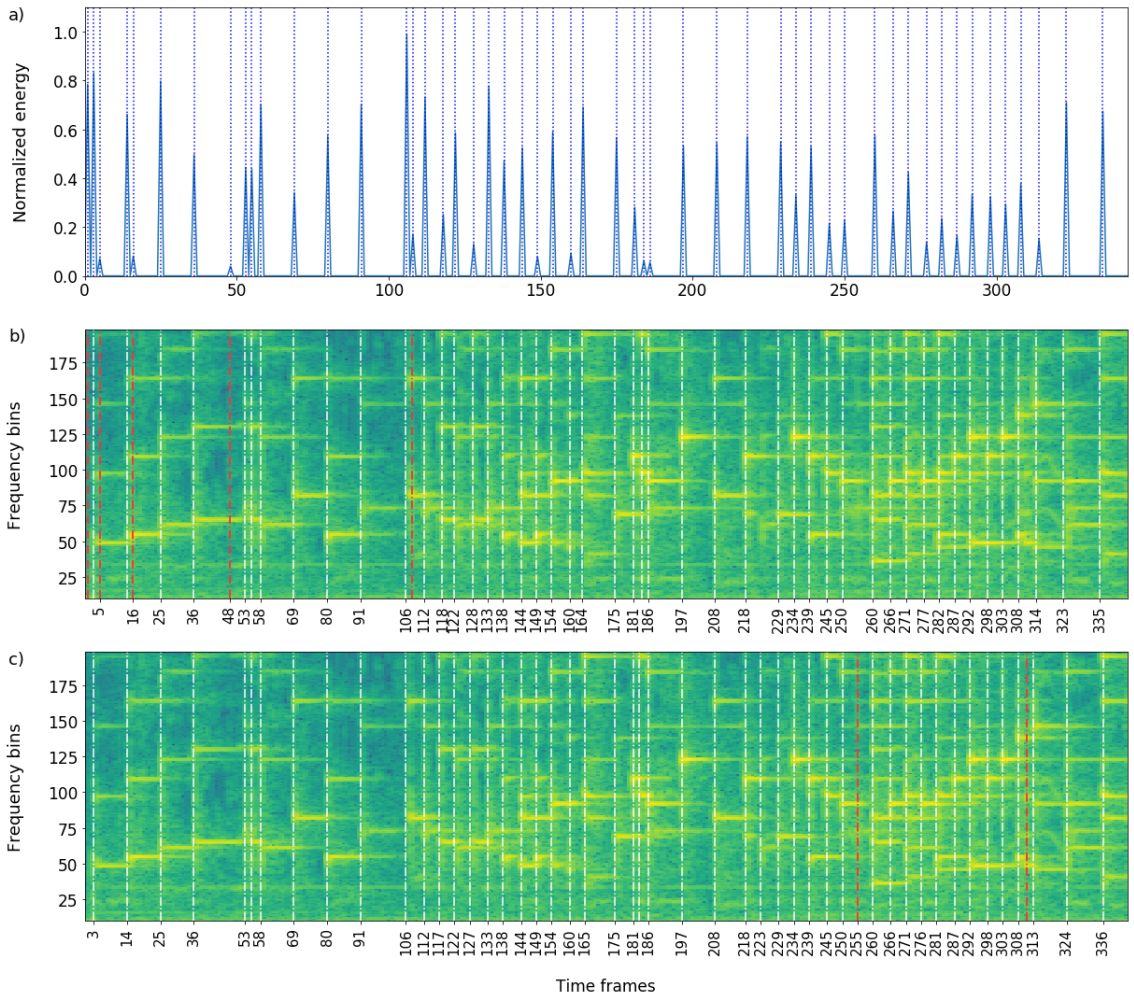

Figure 4.5: Onset detection for two frequency bands: 201 Hz to 1000 Hz and 1001 Hz to 4000 Hz. a) Joined onset functions for both frequency bands. b) Log-spectrogram with the 52 detected onsets. Marked in red are the 4 incorrectly detected onsets. c) Log-spectrogram with the 50 real onsets. Marked in red are the 2 missing onsets.

$$Y_k = \begin{cases} \dfrac{X_k - \mu_k}{\sigma_k}, & \text{if } X_k - \mu_k > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

Experimentally we verified that the best results were obtained by using a step between 50 and 100 bins. Visually inspecting the magnitude spectrum of Figure 4.6, the background noise is significantly reduced in the lower bins region and the amplitude of the partials is increased - because of the division with the standard deviation - which will significantly improve the peak detection. On the higher frequencies region the spectrum becomes very noise because there are no detected partials from a minimum amplitude threshold. At this point the spectrum is also cleaned up for the lower bins where the frequency resolution goes lower 1 bin for the frequencies of the tempered scale.

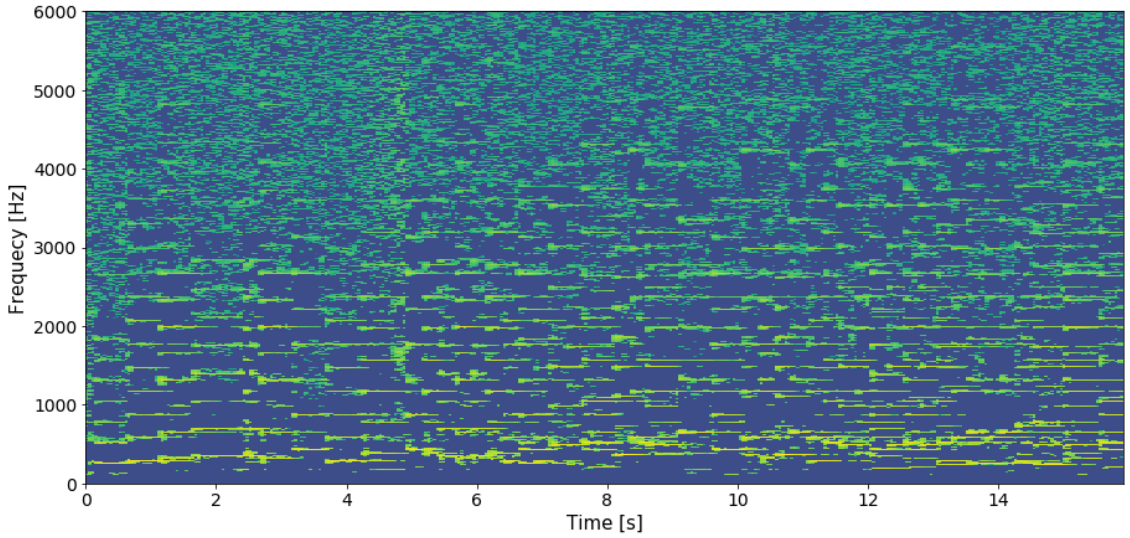

Figure 4.6: Removal of the background spectrum.

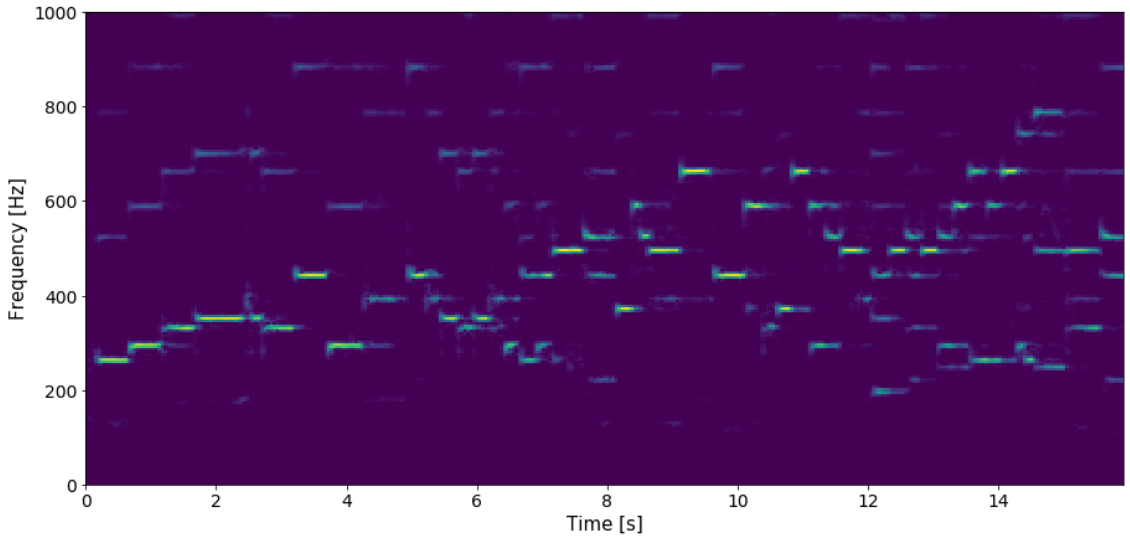

Figure 4.7: Pre-processed log-spectogram in linear units. Besides removing the background noise, after pre-processing the spectral amplitudes are amplified when compared to Figure 4.3.

4.4.2 Fundamental frequencies detection

Once we have the rythmic information, the second step is to determine which notes are played. In order to do so, we need to find the fundamental frequencies for each melodic line from the spectrogram. The problems here are several. First, musical instruments do not produce pure sounds so it's necessary to distinguish both the fundamental frequencies and their partials present in the spectrogram. As an example, consider the piano note A4 with frequency 440 Hz. The spectrogram will not only show a high magnitude peak at the fundamental frequency, but also at the harmonics with frequencies around 880 Hz and 1320 Hz. As seen in Figure 3.4, if the second harmonic corresponds to the octave, the third harmonic is very close to the perfect fifth, in this case

E5 (659.26 Hz). So even if only one note is being played, a simple spectrum frequency matching would conclude both notes A4 and E5 are being played. Of course in theory the amplitude A_n of the piano harmonics decreases exponencially with the harmonics order n [6]

$$A_n = A_1 \cdot \alpha^{n-1}, \qquad n \geq 2 \tag{4.6}$$

where α can vary between 0.1 and 0.9. However, in practice this criterium is not always true. We may have higher harmonics with a higher amplitude than the fundamental frequency and then of course, there's the problem of dimensioning the parameter α, which may also vary with the instrument and the fundamental frequency we are processing.

A second problem is that the piano is an inharmonic instrument. This means its harmonics are not multiples of the fundamental frequency and their spacing increases with the order of the harmonic instead. One of the models that describes the behaviour of the piano partials is [5]

$$f_n = n f_0 \sqrt{\frac{1 + Bn^2}{1 + B}} \tag{4.7}$$

where f_n is the frequency of the n-th harmonic, f_0 is the fundamental frequency and B is the inharmonicity coefficient. B is a characteristic parameter of the instrument. It is experimentally determined and generally takes values between 10^{-3} and 10^{-6} [5]. The approach for f_0 detection followed here is based on [14]. The ideia is to determine a set of f_0 candidates, selected from the peaks of the normalized spectrogram in Figure 4.6. Several magnitude thresholds where tested and the best results were obtained for values between -60 dB and -70 dB. Higher values will exclude most of the detected frequencies, while lower values will produce too many partials which will turn out to cause duplicated pitches and false notes in the detection of the voice contours on section 4.4.3. Harmonic sounds with missing fundamentals will not be considered here although they do seldom appear in practical situations. This approach would require the prediction of the fundamentals based on the patterns of partials, for example, while here we will only consider the already existing spectral peaks as potencial f_0 candidates. The lowest frequencies or the bass voice are usually harder to detect because the frequency resolution of the linear spectrogram is constant (2.20), in contrast with its logaritmic representation that matches the human perception of sound. For this reason, the higher pitches have larger intervals and consequently more bins, so they are detected with more precision. A way to increase the resolution on the lower pitches is to compute a constant-Q spectrogram where the window size changes during the computation of the DFT. The ideia is to obtain a constant frequency to resolution ratio $Q = f/\delta f$ also designated by quality factor. The computation of this transform however would be less efficient than the optimized python FFT function used to compute the STFT, and the results would not be translated into relevant gains. The score here considered does not have a wide frequency span so the lower voices are correctly detected by properly dimensioning the STFT.

After the peaks detection (Figure 4.10) the next step is to validate the f_0 candidates with the partials patterns present in the spectrogram. In order to do this, the piano inharmonicity needs to be taken into account. However, the dimensioning of B can be a complex task, considering it depends not only on the instrument but also on the frequency itself. Its modeling would require a previous knowledge on the spectral behavior of the piano recording - which is in a way the information we are trying to retrieve - and limit the algorithms use to a single instrument or recording. So another approach based on a generalization of B is followed. The ideia is to increase the current partial candidate f_{n+1} by a factor $f_n(n+1/n)$ and search for the first partial match within a positive varying margin Δ_{n+1}/Δ_n. This margin is maximized by using an optimized inharmonicity coefficient, which experimentally was set to $B = 3e^{-3}$, close to the general experimental maximum as expected [5]. This parameter however, although general for all frequencies, deeply conditions the performance of the algorithm. Slightly different values like 10^{-3} or $5x10^{-3}$, for example, significantly decrease the detection of the fundamental frequencies. The detection of the partials f_{n+1} is directly derived from equation (4.7)

$$f_{n+1} = f_n \frac{n+1}{n} \frac{\Delta_{n+1}}{\Delta_n} \tag{4.8}$$

where

$$\frac{\Delta_{n+1}}{\Delta_n} = \sqrt{\frac{1 + B_{max}(n+1)^2}{1 + B_{max}n^2}}$$

corresponds to the search interval for harmonic f_{n+1}. The next detected harmonic f_{n+1}, will be the first matching peak in the margin interval. In case no peak is found, the next harmonic is considered as the start of the search interval $f_{n+1} = f_n(n+1)/n$. For implementation purposes, and in order to increase efficiency of the partials detection, it was added to this margin one bin left to compensate deviations in the discretization of the frequencies or an imperfect tuning. This process is demostrated in Figure 4.9 for the detection of the partials of C3 (261.63 Hz) in frame 10. With the exception of the 9th partial which is below the amplitude threshold, the first 11 partials are detected. For this method to be satisfactorily efficient however, we need to correctly detect the peaks location. For this reason it was used the peak interpolation algorithm available in [16]. This method detects the real peaks of the magnitude spectrum and then approximates these peaks to a parabola, as displayed in Figure 4.8. This happens because the spectrogram represents discreate frequency values, distributed along discrete frequency bins, and they may not match the real frequencies of the signal. So it's necessary to approximate the peak location using some criterium, which in this case is considering the peaks a parabola.

After the detection of the partials, each f_0 candidate is weighted based on the sum of the magnitude of its partials. The best results were obtained considering at least 15 partials. Lower values were translated into not enough information to highlight the real f_0 from sporadic random matches. On the other hand, all candidates missing the second or third harmonics (octave or fifth)

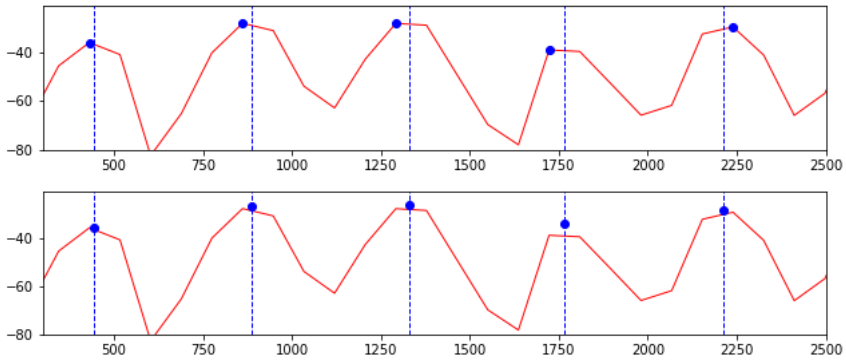

Figure 4.8: DFT with peak detection. On top: Real peaks. On the bottom: Interpolated peak detection.

are discarded as well as the frequencies matching less than 4 partials. The resulting spectrogram is represented in Figure 4.11.

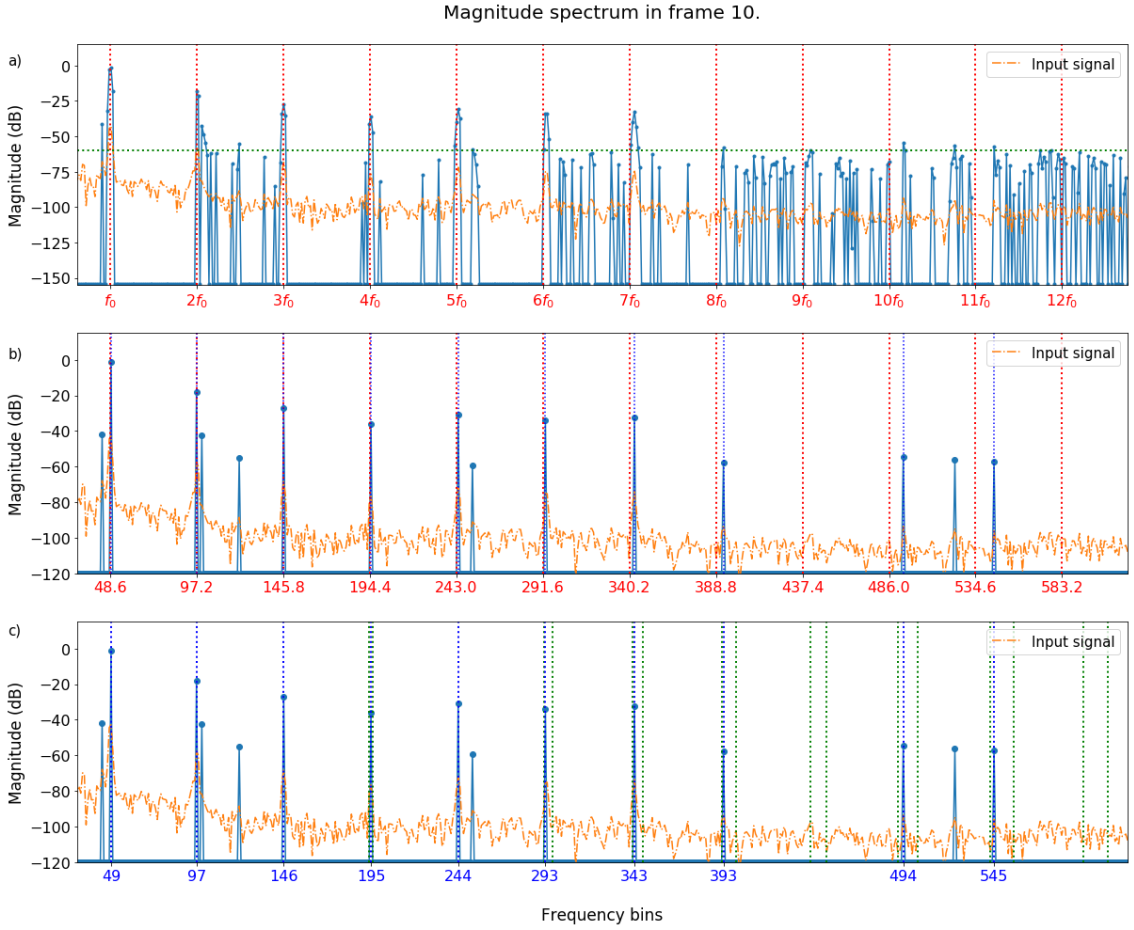

Figure 4.9: Partials detection for C3 (261.63 Hz), k_0=48.6, in frame 10. a) Pre-processed signal (blue) with the natural harmonics of C3 (red) and peak threshold of -60 dB (green). b) Detected peaks (blue) with the natural harmonics (red). c) Detected peaks (blue) with the partials search interval (green).

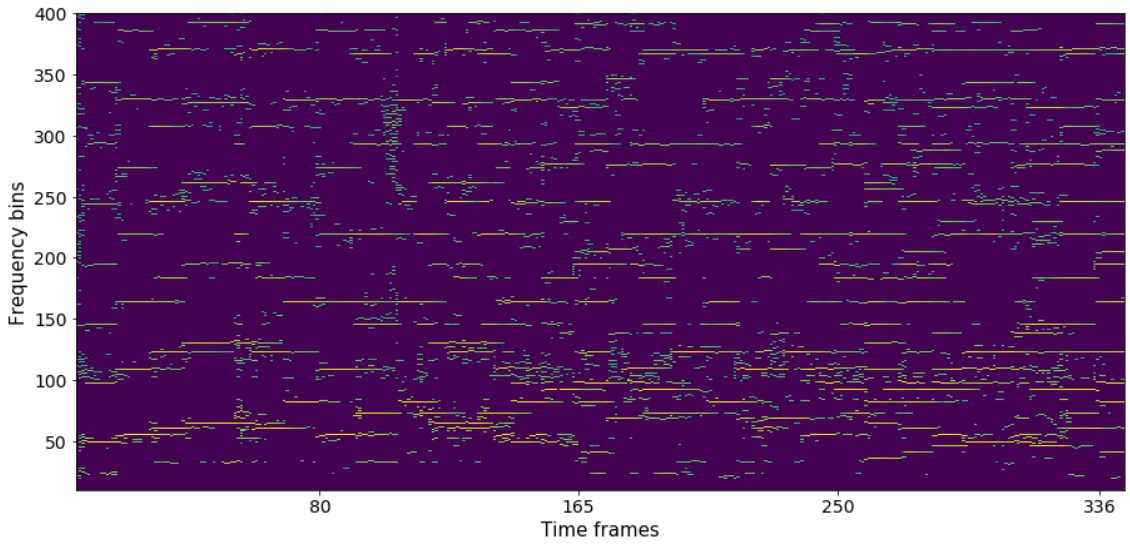

Figure 4.10: Peaks magnitude spectrogram in decibels for a magnitude threshold of -60 dB.

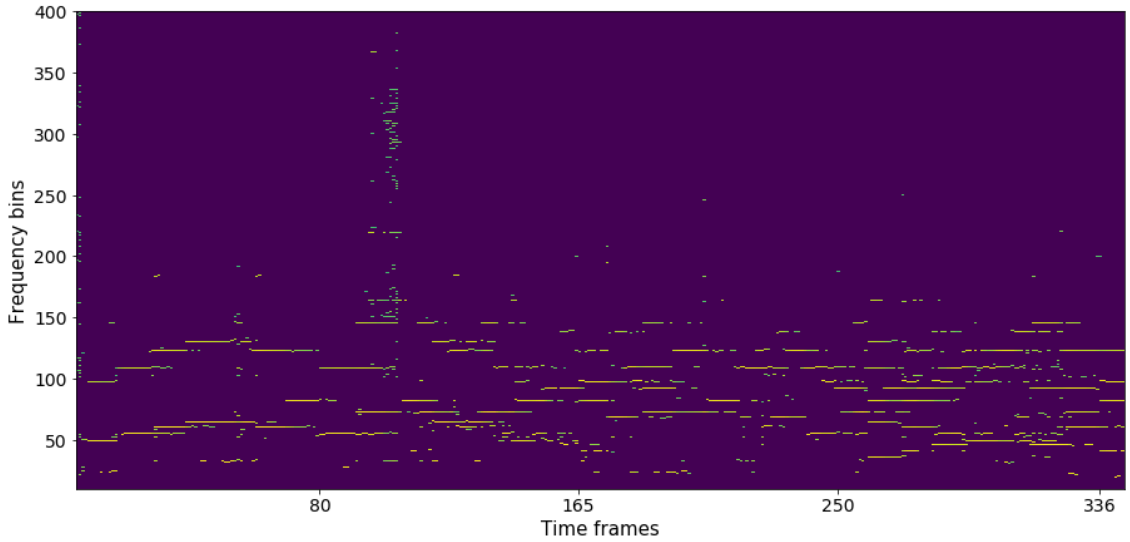

Figure 4.11: Fundamental frequencies magnitude spectrogram in decibels with peaks filtering at -60 dB threshold.

Considering each musical note is not expected to change within a *tempo* frame, the frequency bins are considered valid candidates only if the number of non null points within the *tempo* frame is larger than the number of null points. This simple criterium allows to discard sparse peak detections and provides a first base for extracting the melody. This method, however, is highly dependent on the precision of the onsets: if the onsets are not properly detected, the spectrogram will produce incorrect averaged bins which will definitely compromise all the transcription task. After this step, the data is converted from bins to the pitches of the tempered scale, by computing the middle interval in bins between each frequency of the tempered scale, and then match each one of the f_0 candidate bins to these intervals. The resulting pitch diagram is shown in Figure

4.12.

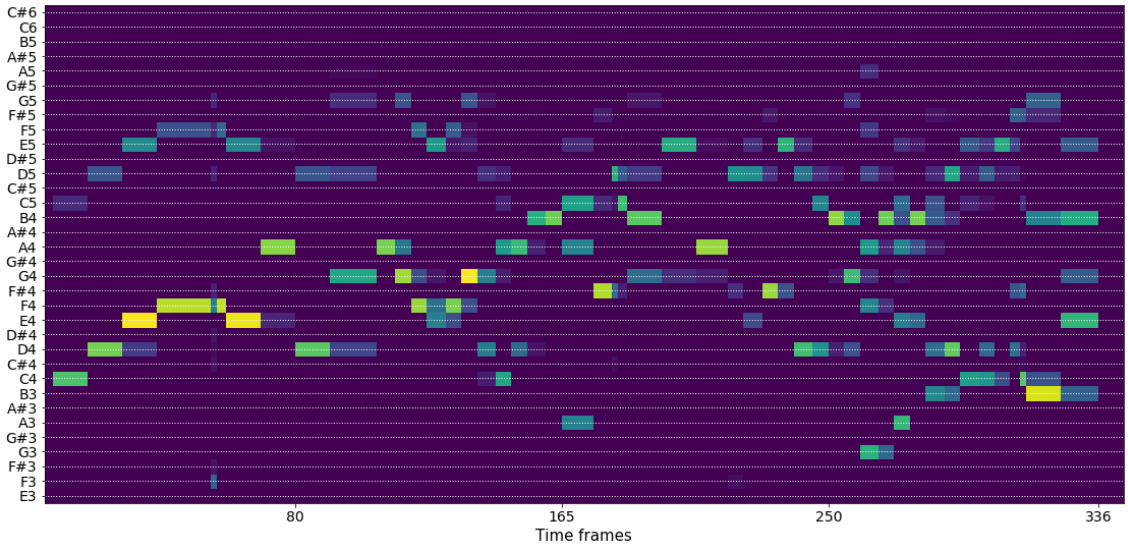

Figure 4.12: Averaged pitch magnitude spectrogram in linear units.

4.4.3 Melodic contour detection

Once the fundamental frequencies are computed, the next challenge is to determine harmonically coherent melodic lines from the available pitches. The fundamental frequencies filtering of Figure 4.11 allowed to significantly reduce the number of potencial pitch candidates from the initial spectrum (Figure 4.10). However, there are still pitches that may correspond to higher partials or sparse detections, such as echos produced by previously played notes. In order to determine which pitches are more likely to correspond to real fundamentals and which ones may eventually be discarded, the pitches along each *tempo* frame are stored in two buffers P^+ and P^-, if their magnitudes are, respectively, above or below the frame's average. One step that considerably reduces the number of potencial partials and improves the melodic contours detection is considering the octave information. The octave interval, corresponding to the second harmonic, if often considered as a potencial fundamental frequency because its harmonics match the even harmonics of the fundamental. So in practice, from the spectrum perspective, both pitches - the fundamental and the octave - can be valid played music notes, differing only on the sum of magnitudes. So the octaves of the pitches in P^+ are flagged and added to P^- buffer for further processing. In [14] a more sofisticated method was used for detecting the melodic contours, taking advantage of the octave information to detect repeated patterns matching a lower voice. The resulting diagram is represented in Figure 4.13.

The fugue is a contrapuntual composition technique that develops a set of independent melodical lines based on a common starting musical theme, which then repeats and transmutes itself along the composition. The challenge of this technique is related to the requirements of coher-

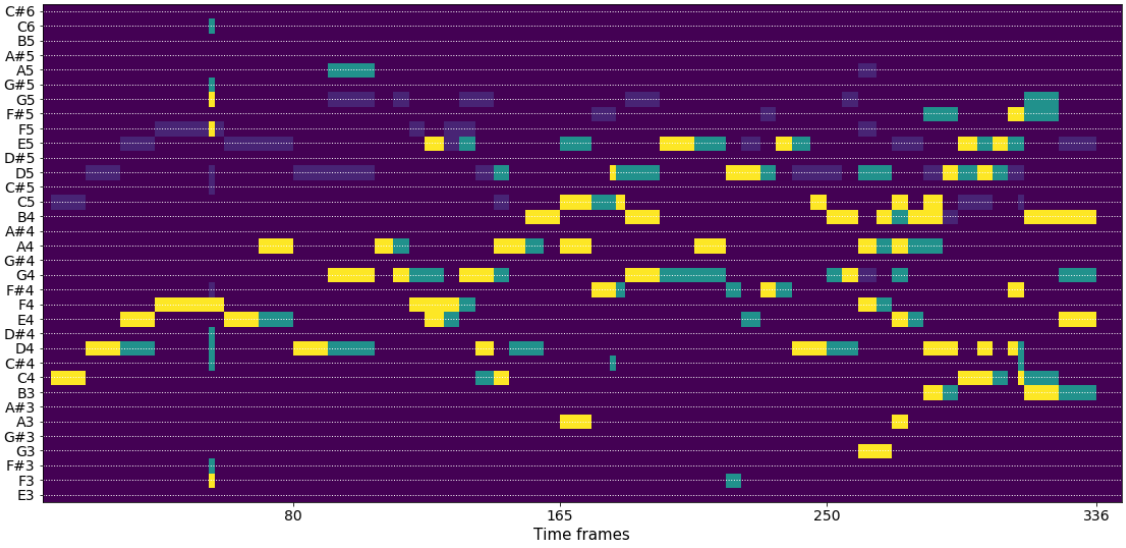

Figure 4.13: High magnitude pitches stored in P^+ (yellow), low magnitude pitches (light blue) and flagged octaves from P^+ (dark blue), both stored in P^-.

ence both horizontally - on each voice's line - and vertically, which means the voices movement has to respect the rules of tonality. Taking advantage on the fugues horizontal independance on the melodic lines, we can expect each voice contour to span vertically along a relatively limited neighbourhood of pitches, which allows us to predict the most likely next pitch of the voice path. This approach however, does not consider any musical concepts or pattern matching methodologies to identify the voice paths, through for example, the common motives like the theme of the fugue. The approach followed here is simpler. The ideia is to select the first unprocessed element of P^+, define a search interval spanning from a perfect forth, both above and below the reference pitch, and find the closest matching pitch on the next tempo frame. This process is repeated for every frame until a path is found and, once finished, the processed pitches are removed from P^+ and the next P^+ candidate is selected for path validation. The P^- buffer stores the secondary pitches that are used for filling up the path in case there is no P^+ candidate available. In case there are no pitches available at all - when the fundamental is missing, for example - it's allowed a one frame jump, assuming the current pitch as the next candidate. This step allows to discard fake paths, mostly consisting on echos of previously played notes and sparse detections. If there are multiple candidates and their weight is the same (considered as a magnitude ratio below 90%), the criterium is to simply choose the highest amplitude pitch from peaks diagram (Figure 4.12). This helps differentiating what may be considered an echo produced by the previously played note - which is the closest candidate to the reference pitch - and the actual played note. At this point we introduce the only harmonic criterium considered in this algorithm which significantly improves the transcription results. When choosing between P^+ candidates, the rejected pitch is removed from P^+ if the pitches are in dissonance, meaning, if they are spaced an interval below a third minor. This is rooted in the assumption that, considering the fugue's tonal structure, there's a high

probability these partterns are not meant to be a dissonant chord but instead two different notes played at different time instants. This step is necessary because several notes - specially the faster ones - still have significantly high magnitude when the next note is played and are wrongly assumed as elements of some voice line. On frame 298 for example, we have a long pitch C4 - corresponding to the correct bass note - and a faster pitch D4 which is the echo of the note played two *tempos* before, reinforced with the partials produced by the D5 played in the soprano one octave up. The decision wheter C4 should remain or change to D4 in the next *tempo* is based on the amplitude of both candidates. As C4 has a higher amplitude than D4, the algorithm selects C4 and removes D4 from P^+.

As for the margin interval set on a perfect fourth, of course a voice can jump larger intervals but due to the proximity of the voices, it will eventually match some valid pitch from another voice. Our goal is not really to detect each single voice accurately. Voice detection, specially the bass, is a complex problem by it's own. What we really aim here is to take a simple approach to validate the pitches which may actually being played, starting from the simple assumption of voice continuity. In this fugue, for example, the first voice starting the theme C4-D4-E4-F4-G4-F4-E4-A4-D4-G4 is the middle one. The bass only starts in the fourth bar. So before that, what we detect is the middle voice which gets mixed with the bass line starting at frame 160. The way these pitches are detected is irrelevant as long as the played notes are detected.

4.5 Discussion of the results

After computing and adding up all the voice contours, we obtain the diagram of Figure 4.14. The best results were achieved with a peak threshold of -60 dB, a reference interval of a perfect fourth and a minimum voice length of five notes. When comparing the resulting transcription diagram (Figure 4.14) with the original music score (Figure 4.15), we conclude the algorithm performs quite satisfactorily. The approach followed here, based on the melodical structure of the fugue, revealed to be enough for the transcription task proposed. Although the vertical analysis proved to be useful regarding the dissonances on specific transition notes, it's by no means a way of actual harmonic validation. We observe mostly two types of transcription errors: either missing or wrongly detected pitches. The transcription errors corresponding to missing notes, marked in green on Figure 4.14, and are mostly due to missing fundamentals (frame 160 for example) which were never detected in the first place by the algorithm of 4.4.2 because of its very low and irregular magnitudes in the unprocessed spectrogram (Figure 4.2). These results for short duration, low frequency pitches is an expected consequence of the time *vs.* frequency trade-off when dimensioning the spectrogram: the lower the frequency, the lower will be its resolution in a log-spectrogram, agravated by the fact that the fast rhythmic behavior worsens the accuracy of the onsets. One possible approach to deal with this trade-off would be to use a varying resolution

spectrogram, like the constant-Q, in order to increase the frequency resolution in the low pitches.

Another parameter that significantly affects the transcription results is the search margin size. If the interval is set to low, we may miss pitches along the path - which may be invalidated if below a minimum length - and if set too high we may end up loosing track of the real voice contour, by ultimately following only the high amplitude pitches P^{+}. This results are very senstive to changes in the margin intervals. For example, if we increase the intervals to a perfect fifth we will miss out the obvious G5 on frame 313 for B4, simply because this pitch has higher magnitude caused by the harmonics of B3 played in the previous *tempo*. As a consequence of increasing the margin size, we may end up selecting wrong pitches as well. In frame 40, for example, the played note corresponds to a D4 but the voice contour algorithm detects only its second partial which is inside the perfect fourth interval. So the algorithm ends up detecting two distinct voice lines at that frame. One way to fix this, could be to perform an octave clean up at the end of the voices computation for the frames with more than the expected pitches. For example, until frame 133 we have two voice fugue, so expect to find at maximum two simultaneous notes on each frame.

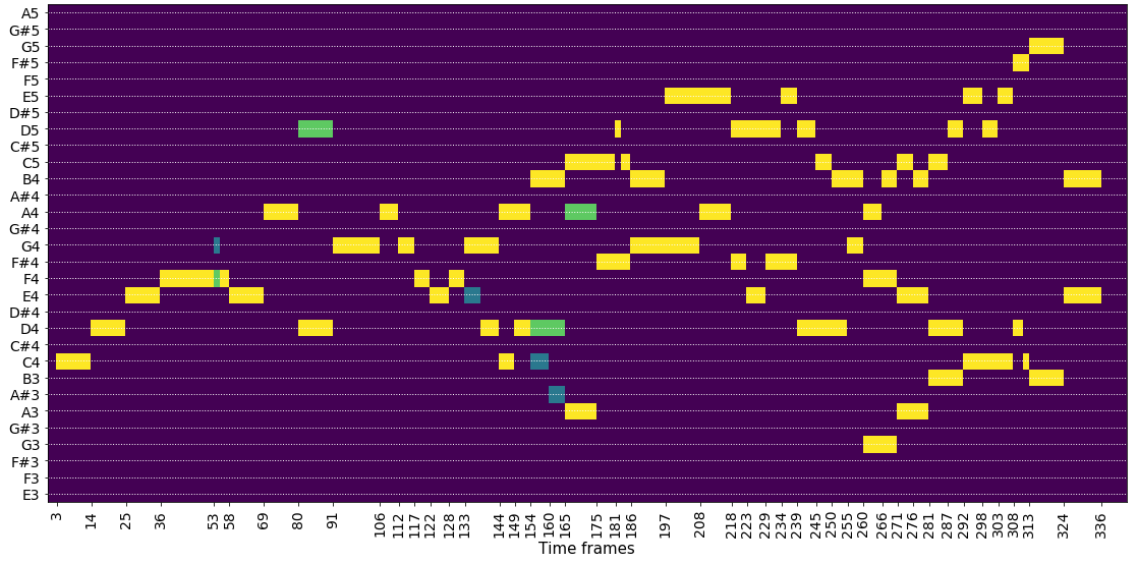

Figure 4.14: Melodic transcription of the first four bars of the Fugue I from J.S.Bach. The green marks correspond to wrongly detected pitches, while the blue marks identify the missing notes. Peak threshold of -60 dB, a reference interval of a perfect forth and a minimum voice length limit of 5 notes.

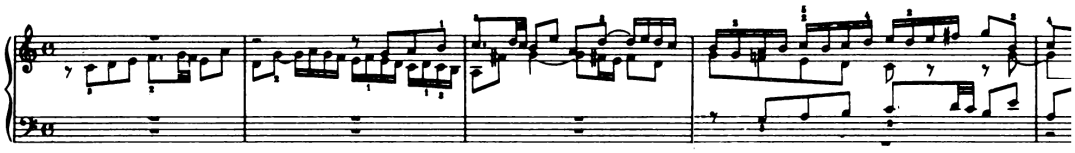

Figure 4.15: Score of the first four bars of the Fugue I in C Major from the *Well Tempered Clavier* BWV 846 (Book I) from J.S.Bach.

Chapter 5

Conclusions

5.1 Conclusions

The algorithm developed did satisfactorily transcribe the excerpt of the proposed fugue recording using simple techniques of audio signal processing. The approach followed here is based on the melodical structure of the fugue, which revealed to be enough for the transcription task proposed. The solution implemented has however a quite limited scope. Firstly, the methodology followed is a simplification of the music transcription problem and is based on several previous assumptions on the data to be transcribed, so it's not generalized to other composition techniques. This method performs well for Bach's fugues, based on a voice independance and continuity assumption. Voice independance allows not to be concerened so much about harmonic validation, through vertical swap of the spectrum. This would make the problem much more complicated if both dimensions had to be considered simultaneously.

This solution also is highly dependent on static parameterization. The correct dimensioning of the reference inharmonicity parameter or the threshold for peak detection are critical for the correct detection of the fundamental frequencies and the performance of the algorithm. The onset detection, for example, is based on pre-defined frequency bands that should be dimensioned according to the type of instruments and frequency span considered for the audio signal. One critical step on the transcription in the computation of the spetrogram. One of the main challenges when dimensioning the spectrogram is the time vs. frequency trade-off. The STFT performance depends heavily on the choice of the window type and size. If a signal has a wide range of frequencies, specially when it concerns the lower frequencies, it requires a very large N to satisfy the frequency resolution requirements, which can be computationlly inefficient and definitely compromise the time resolution. This could be improved by computing a log-spectrogram with a variable size sliding window, in order to improve the resolution on the lower pitches. Instead of implementing a pure constant-Q transform, there are other similar algorithms than allow to improve the frequency resolution using the more computationally efficient FFT function. The time-

frequency trade-off also may cause that fundamental frequencies are not detected at all. So this would require some predictive analysis either on the fundamental frequencies detection step or in the melody tracking. The fundamentals can be detected by matching the upper partials patterns to a defined fundamental pitch, while the simultaneous analysis of melody and harmony would allow to predict the notes with higher probability in the harmonic context of the time frame.

So indeed music transcription is quite a challenging task, specially when dealing with requirements such as automatization, where no human interaction in the middle steps is expected, and becomes increasingly more complex with the variety of music and instruments that are to be transcribed. In this sense, the probabilistic methods and machine learning approaches may perform better on solving such problems.

Bibliography

[1] R. Plomp ad W. J. M. Levelt. Tonal consonance and critical bandwidth. 1965. Institute for Perception RVO-TNO, Soesterberg, Netherlands.

[2] D. Benson. Music: A mathematical offering. Web version, http://www.maths.abdn.ac.uk/bensondj/, December 2008.

[3] T. Cheng, S. Dixon, and M. Mauch. Modelling the decay of piano sounds. Centre for Digital Music, Queen Mary University of London, London, United Kingdom, February 2015.

[4] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation 19 (1965), 297-301*, 1965. https://doi.org/10.1090/S0025-5718-1965-0178586-1.

[5] H. Fletcher. Normal Vibration Frequencies of a Stiff Piano String . *The Journal of the Acoustical Society of America*, 36, 1964. http://dx.doi.org/10.1121/1.1918933.

[6] E. Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[7] D. J. Grout and C. V. Palisca. *História da Música Ocidental*. Gradiva, 2007.

[8] C. Hausner. Design and evaluation of a simple chord detection algorithm. Master's thesis, University of Passau, Faculty of Computer Science and Mathematics, 2014.

[9] J.O.Smith III. Mathematics of the Discrete Fourier Transform with audio applications, second edition. https://ccrma.stanford.edu/ jos/mdft/. Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, 2007, ISBN 978-0-9745607-4-8.

[10] A. Klapuri. Automatic transcription of music, Tampere University of Technology, Department of Information Technology, 1997.

[11] A. Klapuri and M. Davy. *Signal Processing Methods for Music Transcription*. Springer, 2000.

[12] M. Matthias and D. Simon. Approximate note transcription for the improved identification of difficult chords. 2009. Queen Mary University of London, Centre for Digital Music.

[13] M. Matthias and T.Cheng. Modelling the decay of piano sounds. 2015. Queen Mary University of London, Centre for Digital Music.

[14] A. Pertusa and J. M. Iñesta. Efficient methods for joint estimation of multiple fundamental frequencies in music signals. *EURASIP Journal on Advances in Signal Processing*, 2012. doi:10.1186/1687-6180-2012-27.

[15] J. Pons X. Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017. doi:10.1109/ICASSP.2017.7952601.

[16] X. Serra and J. O. Smith III. SMS tools - sound analysis/synthesis tools for music applications. https://github.com/MTG/sms-tools. Universitat Pompeu Fabra Barcelona, Stanford University, 2017.

[17] C. Yeh. *Multiple Fundamental Frequency Estimation Of Polyphonic Recordings*. PhD thesis, École Doctorale Edite Université Paris Vi - Pierre Et Marie Curie, 2008.