Hybrid representation for audio effects

David Cournapeau

September 2003

Abstract

Frameworks to decompose signals into different parts for analysis/transformation/synthesis or coding are quite old in the audio DSP field. The idea behind all these frameworks is that some "basis sets" (here, bases should not necessarilly be taken in the strict mathematical sense) are well suited for some kind of signals, whereas other bases are better for other types of signal. The phase vocoder, introduced by Portnoff in the late seventies, was the first real tool for music DSP, and implicitly assumes that the signal is a sum of sinusoids in each frames. This assumption was the basis for the work of X. Serra in his famous thesis about sine + noise modeling (see [Ser97] for an introduction). This model, also known under the acronym SMS (for Spectral Modeling Synthesis), was the first parametric model for large classes of audio signals [1]

It is well known that additive synthesis with sinusoidal basis vectors can approximate the signal as well as we want; the drawback is well known, too: for some kinds of signals, like "Noise" or "transients", we need a lot of basis vectors to approximate them, and it is quite computive intensive. That's why more recent works, based on Xerra's ideas, try to decompose the signal in at least 3 differents parts, ie the "tonal" part, the "transients" part, and the residual (also called noise, because it is generally coded as a stochastic process). These kinds of models has been successfully used for analysis/synthesis for years in academic areas and in commercial products [2].

Laurent Daudet developed [Dau00]a parametric model for audio signals, essentially aiming at compression. This model splits the audio signal into three parts, tonal / transients / residual. But instead of using a sinusoidal model, it uses thresholding on the MDCT to extract the tonal part; in a very similar way, transients are extracted by thresholding on wavelet coefficients of the non-tonal part. Hard thresholding is basically a non-linear approximation[3], and is famous since its successful use in image compression and signal denoising[Mal98]. The idea of my internship is to adapt this model to analysis/synthesis, and to use it in some high-level transforms like time-scaling, and more complex schemes like tempo correction (ie changing the time location of some notes to adapt it to a fixed time-grid).

At first, we will present the basic framework: the principles, the results on some test samples and the drawbacks for analysis/synthesis purposes,

[1] the sinusoidal model was also studied a few years before for speech coding by McAuley and Quatieri [MAR86]

[2] See for example the realizer from PPG, figure 1 or more recently the neuron synthetiser project, figure 2

[3] See Annexe 1

Figure 1: The realizer, the first virtual synthetiser?

Figure 2: The Neuron synthetiser; may use some resynthesis method

mainly on the tonal extraction. The second chapter will present the work I did to try to avoid some of the main problems, like window switching, wavelet filtering and MDCT regularisation. The third chapter is a presentation of the further ideas one can develop to go around the tonal extraction problems, mainly about a "complex MDCT" which gives phase informations, and so phase can be used for steady state compenent exctraction. Finally, the last chapter will give some details about some implementations I did in matlab/C++ for my work.

L'idée de représentations paramétriques pour de larges classes de signaux audio n'est pas nouvelle. Toutes les représentations existantes à ce jour reposent sur l'hypothèse que certains types de signaux sont bien représentés dans certaines classes de vecteurs alors que d'autres types de classes sont plus adaptées à d'autres types de signaux (ces classes de signaux ne sont pas forcément des bases dans leur acception mathématique du terme, mais sont plus généralement un ensemble de vecteurs ni forcément générateur ni forcément linéairement indépendants, et pouvant à la limite dépendre du signal). Le vocoder de phase, outil introduit par Portnoff à la fin des années soixante-dix, fut le premier véritable outil permettant l'analyse synthèse, et repose implicitement sur l'idée que le signal est une somme de sinuoides dans chaque fenêtre (c'est par exemple l'hypothèse sur laquelle repose le time-scaling par vocoder de phase). Cette même hypothèse fut le point de départ du travail de X. Serra sur la modélisationn des signaux audiophoniques par ue somme de sinuoides plus bruit (voir par exemple [Ser97] pour une introduction). Ce modèle, également connu sous l'acronyme SMS, pour Spectral Modeling Synthesis, fut le premier modèle explicitement paramétrique d'une large classe de signaux audiophoniques [4].

On sait que la synthèse par somme de sinusoides peut approcher un signal donné avec une précision aussi bonne que voulue, sous reserve que l'on utilise assez de sinusoides; mais cette méthode a un inconvénient de taille, à savoir le nombre de paramètres mis en jeu: pour certains types de signaux comme le "bruit" ou les transitoires, le nombre de sinusoides nécessaires à une bonne approximation du signal original est gigantesque. C'est pourquoi des travaux plus récents que ceux de X. Serra tentent de décomposer le signal en au moins 3 parties: la partie tonale, les transitoires et le résidu (souvent appelé bruit, car souvent considéré comme la réalisation d'un signal stochastique). Ce type de modèles est déjà utilisé aussi bien dans les milieux académiques que dans des outils du commerce: voir par exemple certains systèmes de la firme PPG dans les années 80 (figure 1) ou vraisemblablement celui de la firme Hartmann-Neuron (figure 2), bien que le fonctionnement de ce dernier demeure mystérieux.

Laurent Daudet a développé dans sa thèse [Dau00] un système de décomposition hybride, dans une optique de codage du signal musical. Ce système décompose le signal en trois parties, la partie tonale, la partie transitoire et le résidu. Au lieu d'utiliser un modèle sinusoïdal ou un de ses dérivés, ce modèle extrait la partie tonale par un seuillage dur sur les coefficients de la transformée en cosinus discrete modifiée, ou Modified Discrete Cosine Transform; la partie transitoire est extraite de manière similaire, par seuillage dur des coefficients en ondelette de la partie non tonale (le résidu est le résultat de la soustraction de la partie tonale sur le signal original). Le

[4] le modèle sinusoidal fut également étudié quelques années auparavant par Mc Auley et Quatieri [MAR86] pour le codage de la parole

seuillage dur est en fait une approximation non-linéaire du signal [5], et a été utilisé avec succés dans les domaines de la compression d'images ou le débruitage[Mal98]. Le but de mon stage est d'adapter cette représentation à l'analyse/synthèse, et de l'utiliser dans des transformations de haut niveau, telles que le time-scaling ou la correction de tempo (ie changer la localistion de certaines notes sur une grille de tempo fixée; on pourrait appeler ce type de transformations quantisation audio).

Dans un premier temps, je vais présenter les bases de la représentation hybride utilisée: les principes, les résultats sur quelques échantillons sonores que j'ai utilisés comme références, et les principaux problèmes de la représentation pour l'analyse synthèse. Le deuxième chapitre explique les différentes techniques que j'ai étudiées pour résoudre certains des problèmes du premier chapitre, à savoir un changement de taille de fenêtre adaptatif, le filtrage dans le domaine des ondelettes, et la régularisation de la MDCT. Le chapitre 3 présentera le début du travail effectué pour résoudre certains problèmes liés à l'extraction de la partie tonale, reposant sur une MDCT complexe, permettant par là d'utiliser la phase.

[5] voire l'annexe 1

Acknowledgements

I first want to thank all the team of the DSP lab of Queen Mary for the good atmosphear of work. I particularly want to thank Nicolas and Paul, for too many reasons, Chris for stimulating discussions, and Juan, Down and Josh for checking my too many grammar and spelling English errors. I also want to thank Laurent Daudet, who helped me a lot during all my internship in Queen Mary.

Contents

List of Figures

Chapter 1

Hybrid representation: the basic framework

1.1 Why a parametric model?

In his thesis, L. Daudet developed a hybrid representation for audio signals : his aim was mainly to use this representation for compression. Before presenting the model in-depth, it is useful to recall why modeling is important in audio DSP field.

Today, most audio storing, broadcasting and diffusion is done in a digital way. The first, and the easier way to code an audio signal is to use a similar way to analog techniques, ie coding the amplitude in time: it is the PCM (Pulse Code Modulation) technique. It is used in the oldest and still the most successful digital support, the audio CD. As digital storage imposes to code the amplitude and time with a finite set of values, one uses discrete values. So it is important to be aware of the artefacts caused by time and amplitude discretisation.

1.1.1 PCM coding

Theoretically, the Shannon-Whitaker theorem shows that if the signal is band-limited, sinc interpolation can exactly reconstruct the signal from the regularly spaced time samples. But sinc interpolation demands a knowledge of all the samples of the signal to reconstruct each value, so more simple schemes are used for the digital to analog conversion. The sampling rate of the CD is fixed to 44.1 khz, which means that theoretically, all frequencies below 22.05 khz can be represented (generally, the human hearing cannot perceive any frequency higher than 20 khz). But in the real word, the signal must be filtered before its digitisation, so there is an anti-aliasing filter which tries to keep all frequencies unchanged below 22.05 khz, and remove all frequencies higher than 22.025 khz. Practically, it is impossible to have

such a filter, so one try to make filters which have a very steep slope; this steep slope causes big problems for transients: because the phase response of this kind of filter is highly non-linear around the cutoff frequency, the signal's time representation is heavily changed for high frequencies[1]. As most "transients-like" signals have a high frequency content, these non-linearities in the phase response tend to "blurr" the transients. That's the main argument to have 96 khz, or even higher sampling-rate convertors: that way, it is much easier to design the brickwall filter, with a more linear phase response in the audible range.

For the CD, the amplitude is coded with 16 bits length words, with a uniform and scalar quantizer. For this kind of coding, the word-length is linked to the round-off error. Basically, once the maximum possible value x_{max} is fixed, the smallest difference between two consecutive coded values Δ is around [2]:

$$\Delta = \frac{2 * x_{max}}{2^R} \tag{1.1}$$

where R is the number of bits per word. So the range of amplitudes which map into a single code depends on R. The bigger R is, the smallest this range is: this kind of error is called round-off error. If we assume that the amplitude of the input signal follows an uniform distribution, the round-off error will be likely in the range $[-\Delta/2, \Delta/2]$ (ie the probabilty density of the error signal ϵ is uniform in the range $[-\Delta/2, \Delta/2]$), and so the error power of the quantizer can be written the following way:

$$\begin{aligned} \mathbb{E}[\epsilon^2] &= \int_{-\infty}^{\infty} \epsilon^2 p(q) dq \\ &\approx \int_{-\Delta/2}^{\Delta/2} \frac{\epsilon^2}{\Delta} dq \\ &= \frac{\Delta^2}{12} \end{aligned} \tag{1.2}$$

Thanks to 1.1, we can write that:

$$\mathbb{E}[\epsilon^2] \approx \frac{{x_{max}}^2}{3 * 2^{2R}} \tag{1.3}$$

And for a signal with an input power of P_{in}:

[1] here, frequency must be understood in a DSP point of view, and not on a perceptive or even physical one

[2] in fact, the exact value depends of the type of uniform quantizer: midread, where a zero input is coded to zero, and midrise, with no zero output. See for example [MB03] for an introduction to scalar quantizer

$$
\begin{aligned}
SNR &= 10 * log_{10}(\frac{P_{in}}{\epsilon^2}) \\
&= 10 * log_{10}(\frac{P_{in}}{x_{max}^2}) + 6.021R + 4.771 \qquad (1.4)
\end{aligned}
$$

This last equation shows the relation between the relative power of the signal and the SNR. With 16 bits, for a sinusoid with a amplitude equal to x_{max}, we obtain a SNR of 97 dB. This value may seem quite big, but in fact, most professionals who are working with numeric-based tools had a lot of problems when mixing different signals (because most signals are quite below the maximum value, and there is a lack of "headroom"; the cinema industry was the first one to use 18, 20 and 24 bits resolution). That's why most recent formats (like the DVD audio) can work with a 24 bits resolution (that is the internal of most semi-pro and pro audio convertors, and the internal resolution of pro tools, which is one of the most used direct-to-disk tool in pro audio).

So, audio signals with such a coding scheme will have a bitrate of 24 * 96000 * n, where n is the number of channels. For a stereo signal, it means that the bitrate is around 500 ko/s, and so 1 hour of music is around 2 Gbyte ! For modern hard supports like DVD, it is not a problem. But for network broadcasting, or embedded tools (for example some small tools built upon PDA-like systems), it is still quite annoying. Even for audio-cd quality, the amount of data is still too big for some applications.

1.1.2 Audio compression

That's why audio codec receive a huge amount of interest in the last years [3]. Today, with the state-of-the-art waveform codec, like AAC, one can aim at a compression ratio to 1:15, with a very good audio quality. It seems very difficult to do better now with waveform codec, ie codec which don't use any model for signal, partly because we still don't know well how masking is behaving with complex signals.

Recently, some parametric techniques were developed to allow compression and compression-domain transformations, like [SL]. Parametric models, physical ones this time, are also used in Mpeg4 (the media-objects, see [MB03] for a presentation of Mpeg-4).

[3] everbody knows the so-called mp3 format, and it is certainly one of the invention in DSP field which has the most economic impact (mp3 was the first format to enable massive sharing of media contents, and was one of the ground for all the discussion about copyright, Digital Right Management, etc...)

1.2 Generalities about hybrid representations

1.2.1 What is a hybrid representation?

Fourier analysis is by far the most used tool in audio DSP: it enables to map audio signal into the frequency domain, amd one knows very fast algorithms to do it with discrete and finite signals based on the FFT algorithm, which enables to do the DFT *in place*. It is well suited for the analysis of pseudo stationary signals, but it fails with highly non-stationary signals. This drawback is known for ages, and several tools were discovered and studied to make a good time/frequency analysis. The famous Heisenberg relation, that is, for any $f \in \mathbf{L}^2(\mathbb{R})$:

$$\sigma_t{}^2 \sigma_\omega{}^2 \quad \geq \quad \frac{1}{4} \tag{1.5}$$

with σ_t and σ_ω the variance of f and its Fourier Transform respectively, shows that we can not have a time/frequency representation which is good in time and in frequency. So we have to deal with a trade-off, which is particularly annoying in audio DSP.

The other problem is that there is no unique definition of time-frequency energy density; theoretically, one can show that all time-frequency distributions can be obtained from the averaging of a quadratic form called the Wigner-Ville distribution [4].

So, instead of using *one* space representation with its basis, one can think about using a larger set than a basis, like frames, or sets of vectors which are build upon a dictionary (that is matching pursuit techniques); to be more general, we can also use not one but several kind of set to describe the signal: it is the principle of hybrid representation. One decomposes the signal into differents parts (for example the tonal and the stochastic part), and codes them into different sets of vectors. This can be used for source separation, denoising, compression, and other relevant transformation on signals.

Two examples of hybrid representations:

- signal denoising thanks to successive decompositions into trigonometric and wavelet packet bases [RCG94]. The idea is that the underlying signal and the noise are well represented in some bases, ie with a few and big coefficients, and very bad represented in other ones, ie with a lot of small coefficients. More important, the "good" bases should be different for the different kinds of signals. This scheme is implemented with thresholding on the coefficients [5].

[4] see the chapter 4 of [Mal98] for a theoric background about the relation between Gabor atoms, windowed Fourier Transform, wavelet and Wigner-Ville transform

[5] see Annexe 1

- SMS [Ser97]: it is an analysis/synthesis framework for audio signals. It splits the signal into two parts, the *tonal part*, modelised by additive synthesis of sinusoids, and the *stochastic part*, modelled at first by ARMA approximation, and by more complexe schemes after, to take into account the frequency resolution of the human hearing (the noise is taken through a bank filtered which models the so-called critical bands).

1.3 Presentation of our hybrid representation

1.3.1 Principles

The model we will develop here sucessively decomposes the signal into 3 parts: the tonal part, the transients, and the stochastic part. After each extraction, the extracted part is removed from the input signal, and the residual feeds the next extraction (see 1.1)

1.3.2 Tonal extraction

Filter Bank

We know that most relevant informations in audio signal are best seen in the frequency domain. The first audio codec in MPEG scheme used some filterbanks to do it. Filterbank is a set of filters which splits the input signal into different frequency bands; so one can do some processing on each band. There was a huge amount of research about filterbanks and multi-rate DSP processing (see for example the chapter ten of [Mit01] to see an introduction to multi-rate processing, filter bank and polyphase implementation). The main problems which arise when designing filterbank are efficiency and perfect or near-perfect reconstruction.

1.3.2 show a 2 bands filterbank: to avoid data-rate increase, each filtering is followed by a decimator, so that the total data-rate at the output of the filter bank is the same than at the input. If one imagine that the two filters are perfect, ie their slope is vertical, one doesn't loose any information in the process; indeed, as the spectrum is half wide as the input's one, the sampling theorem assures one doesn't lose by downsampling by a facto 2. But as filters with vertical slope don't exist, one can wonder if we don't loose anything with such a process. Remarkabely enough, there exist some filterbanks which enable perfect reconstruction, using an *aliasing cancellation* principle.

Knowing the basic Z-transform for decimation and interpolation, one can easily show that the Z-transform of the output $X'(z)$ is expressed thanks to the Z-transform of the input $X(z)$ by the following relation:

$$X'(z) = \frac{1}{2}(H_0(z)G_0(z) + H_1(z)G_1(z))X(z)$$

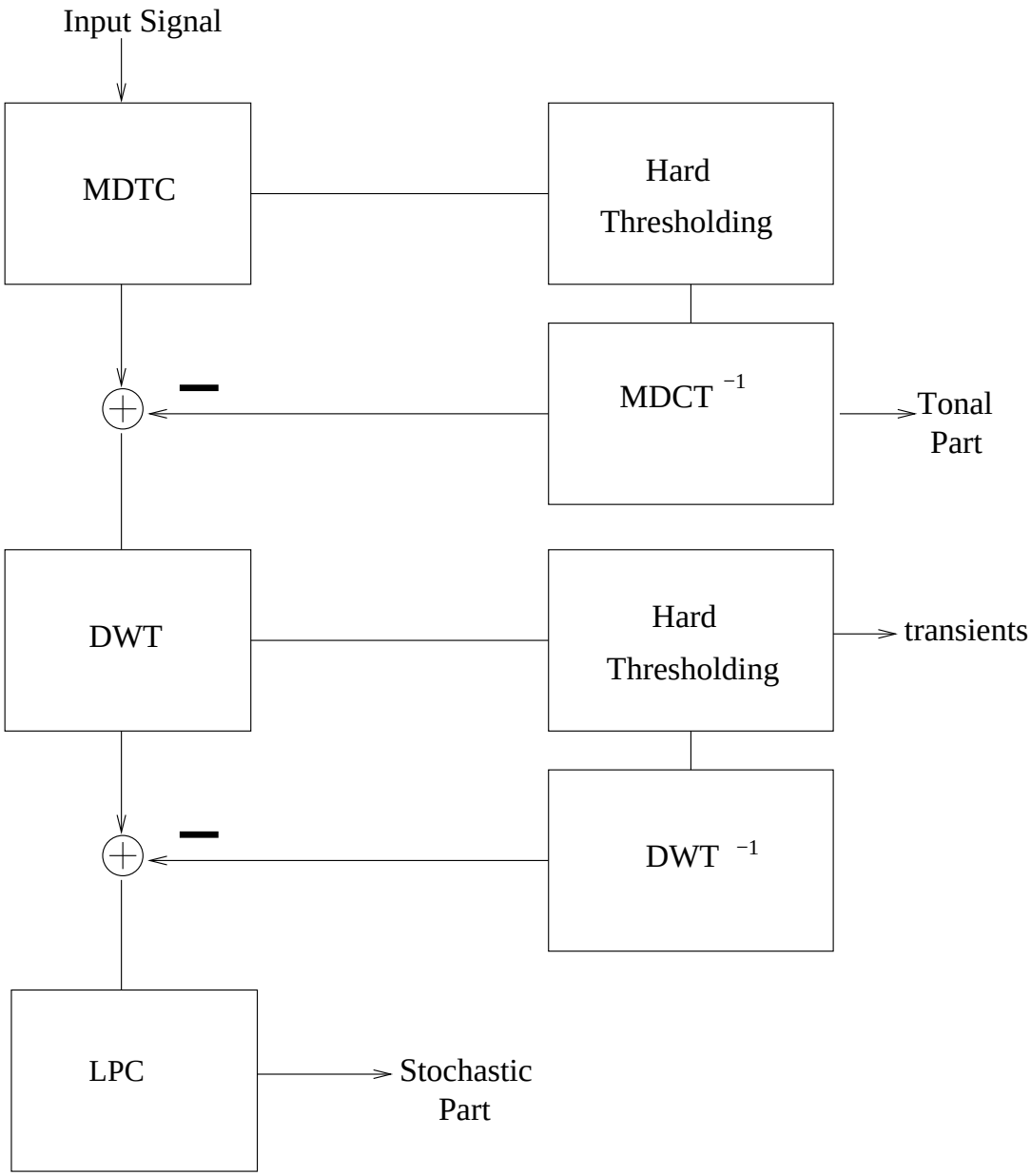

Figure 1.1: General scheme for the hybrid framework

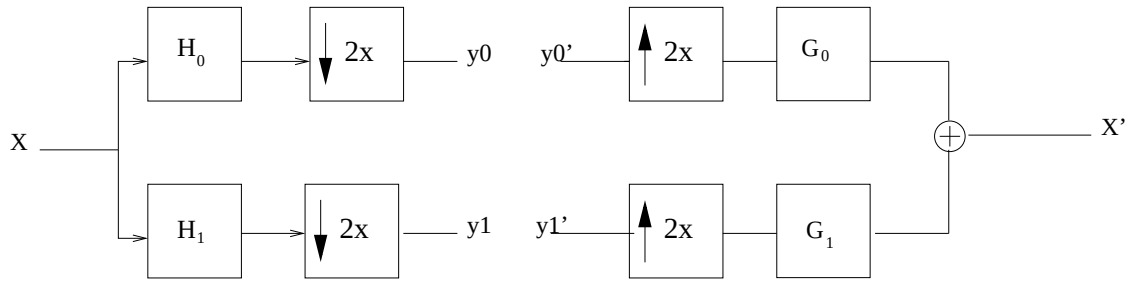

Figure 1.2: A 2 bands filter bank

$$+\frac{1}{2}(H_0(-z)G_0(z) + H_1(-z)G_1(z))X(-z) \tag{1.6}$$

The second term, proportionnal to $X(-z)$, is the aliasing term, so it must be cancelled for perfect reconstruction. The first way was to define the reconstructions filters G_i in terms of H_i:

$$\begin{aligned} G_0(z) &= -H_1(-z) && (1.7)\\ G_1(z) &= H_0(-z) && (1.8) \end{aligned}$$

With these relations, the aliasing term is obviously cancelled. Once H_0 is defined, we still need to find H_1 so the first term is a simple gain. The first solution is the Quadrature Mirror Filters, or QMF [MB03]. The filter H_0 is defined like 1.9:

$$\begin{aligned} H_1(z) &= -H_0(-z) && (1.9)\\ h_1[n] &= -(-1)^n h_0[n] \end{aligned}$$

1.9 means that H_0 must verify the QMF law, that is:

$$H_0(z)^2 - H_0(z)^2 = 2z^{-z} \tag{1.10}$$

For example, the Haar Filter is such a filter.[6]. Haar filter are FIR filters whuch have only 2 coefficients, so the frequency resolution is very bad (see the figure 1.3). Unfortunately, we still don't know any other FIR which can be used to build a perfect QMF filterbank; we have to use some FIR which approximate the QMF solution.

[6]see Annexe 3 for a brief theoritical background about Wavelet and Haar filters

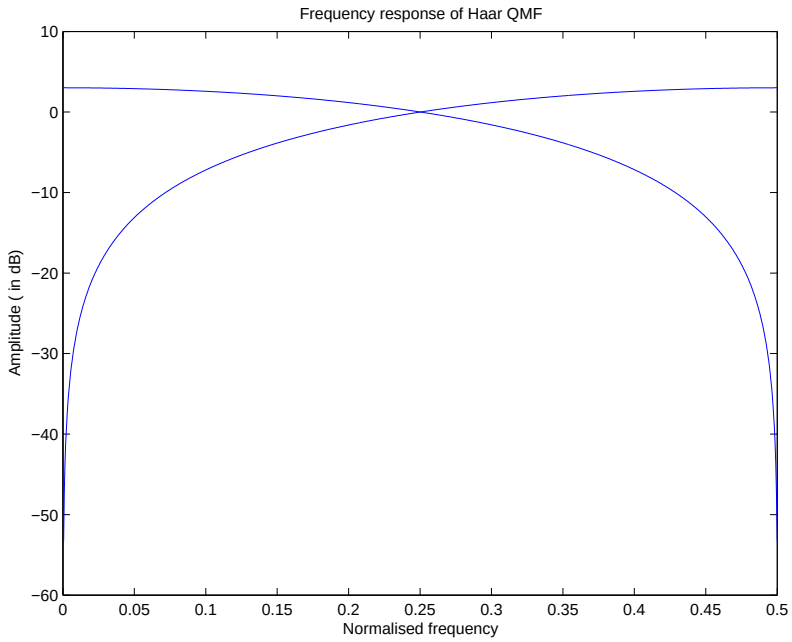

Figure 1.3: Frequency response of the Haar filterbank

QMF condition is not the only one to ensure perfect reconsruction. An other relation, called CQF (for Conjugate Quadrature Filter), is better suited for FIR filters. For example, the 32 bands PQMF filter bank from audio layers in MPEG1 and MPEG2 is derived from a 2 band CQF filter bank. See chapter 4 of [MB03] for a detailed explanation of the PQMF filter bank used in MPEG audio coding.

The MDCT

An other approach for time-to-frequency mapping has emerged from transform coding, the block-based transforms. Here, the main problem is to avoid edge effects between adjacent blocks: this is linked to the used windows, and the amount of overlapp between successive blocks (see annexe 2 for a quick introduction to windowing effect).

The Short Time Fourier Transform (STFT) is a well-known block-based tool: it takes some overlapping blocks from the input signal, windows it and computes the FFT, and the signal is resynthetised with another windowing and overlapp-add method. But if we take a N point Discrete Fourier Transform of each N-size time block, there is an increasing of the data-rate, because of the overlapp. It must be avoided in a coding scheme; that's why some people try to find some block transforms which don't increase the data-rate, without any lose in the reconstructed signal.

One way to accomplish this requirement is the so-called Time-Domain-Aliasing-Cancellation methods: this kind of transforms are not invertible, but overlapping and adding succesive transformed blocks cancels the reconstruction artefacts. The second annex briefly explains how this works for a special case: the Modified Discrete Cosine Transform, which can also be seen as a windowed version of the Discrete Cosine Transform. An other point of view on the MDCT is that it introduces less irregularities than DFT and DCT. The MDCT has been successfully used in the third layer of audio MPEG1, and is the fundamental tool for time-to-frequency mapping in more recent audio coders, like AAC.

It worths noticing that PQMF techniques and MDCT are in fact two different ways to describe the same process: mathematically, they are stricly the same; one uses one or another according to the number of channels.

The idea here is that steady-state like signals (ie the tonal part) are well represented by the biggest MDCT coefficients of the signal. So an hard thresholding on it should give us a good approximation of the tonal part. Once the MDCT is choosen, one have to choose the the window's length, the overlapping and the window.

- As we neither want to lose too much frequency resolution or time resolution, we have to choose a compromise: L. Daudet fixed it to 1024 samples (at a 44100 sampling rate, it means that each block is 23 ms long, and has a 44100/1024 resolution, ie 43 Hz), with an 50% overlapp.

- The two most used windows for the MDCT are the Sine window and the Kaiser-Bessel-Derived. The 2d annexe described the main properties of these two windows. To sum-up, the KBD window has a steeper slope but a wider first lobe than the Sine window (see 1.3.2); so the Sine window is better to separate two narrow bins.

The most difficult part of the approach here is to choose the threshold. The easiest way would be to impose a global threshold, but the results would be very bad: the threshold must be chosen adaptively according to the "dynamic" of the signal. L. Daudet suggested two different techniques to adapt the threshold at each frame.

The first technique is based on the fact that each tonal component will be represented by one big coefficient and its nearest neighbours. So we can fix the threshold τ according to a fixed ratio of the block's biggest coefficient:

$$\tau = \rho max|\hat{x}[k]|, k \in 1 \dots 1024 \tag{1.11}$$

But this method has a big disadvantage: τ can change a lot between consecutive blocks (fast transients often behave as a burst of energy); more

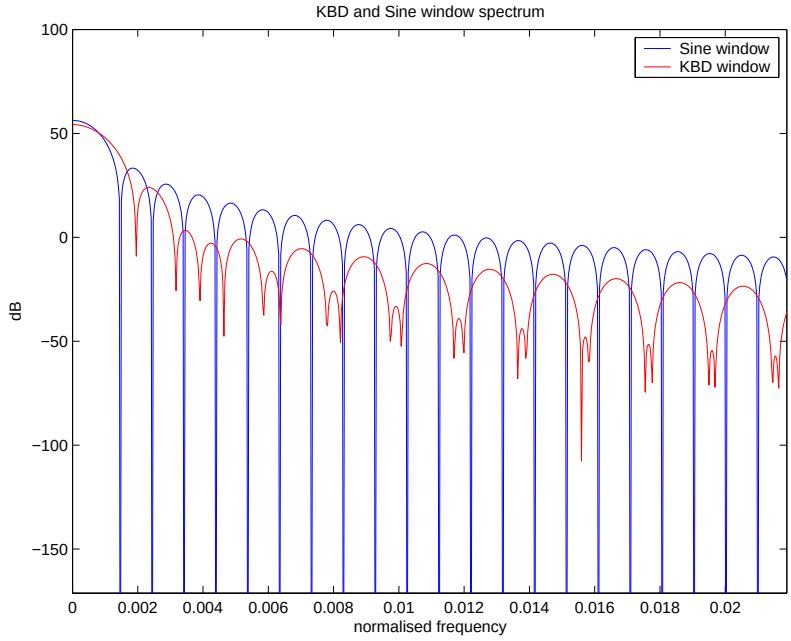

Figure 1.4: Spectrum of KBD and Sine windows

pedantly, the max operator's variance is high. As we are not focusing about compression here, it doesn't look very important. But in fact, it is, because of the nature of the MDCT: a simple sinusoid can be kept in one frame and "forgotten" in another because of the phase difference [7]. Even if, as we will see later, this problem can be partially solved with MDCT's regularisation (see 2.1.2), the second method, presented here, will be prefered.

This method uses quantile as a way to estimate the probability density, and fixes the local threshold τ as a fixed ratio of a fixed quantile. For example, if $\mathbb{P}$ is the probabilty density of the MDCT coefficients, we fixe the quantile to p, and try to estimate z_p such as:

$$\mathbb{P}\{|\alpha| \geq z_p\} \quad = \quad p \tag{1.12}$$

and so τ is fixed to the following value:

$$\tau \quad = \quad \rho * z_p \tag{1.13}$$

The longer the window is, the better is the estimation of $\mathbb{P}$, but it is fixed to the block's length. So, to have a proper estimation, we take a horizon

[7] We shouldn't forget here that MDCT is real, so the coefficients contain amplitude *and* phase information.

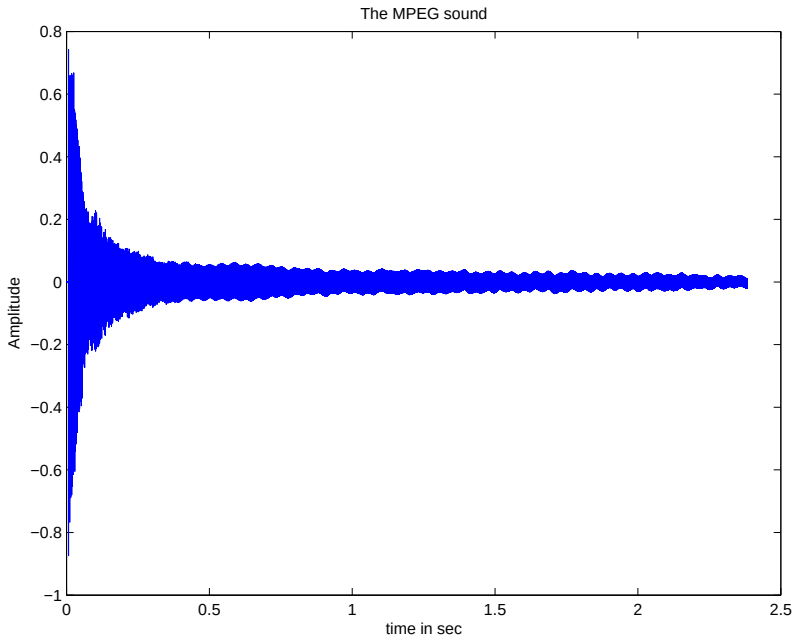

Figure 1.5: The time representation of MPEG sample

wider than one window: 4-5 times the window's size (ie around 100 ms for a sampling rate at 44100 Hz) seems to be sufficient in most cases.

Results

This first tonal extraction was implemented in matlab in the **transTonal.m** script, and uses the MDCT scripts I implemented [8].

I first try very simple sounds, like monophonic sound: I try to have the same results than in [LD02a], with the MPEG sample, and then try with more complexe sounds.

The first sound, used in [LD02a], is monophonic, has a rather "mellow" attack, and is quasi tonal after a short period, with a strong pitch (see 1.5 and 1.6). I try several values for p, between 0.9 and 0.999 (at 1, all is in the residual). Figure 1.7 shows the tonal extraction and the residual for a threshold at 0.995.

- The tonal extraction in itself is rather good: there is no residual harmonic sound in the non-tonal part when listening to it, and the tonal part is not heavily modified.
- There is a pre-echo effect, which was previsible: it is a well known

[8] see 4 for an explanation

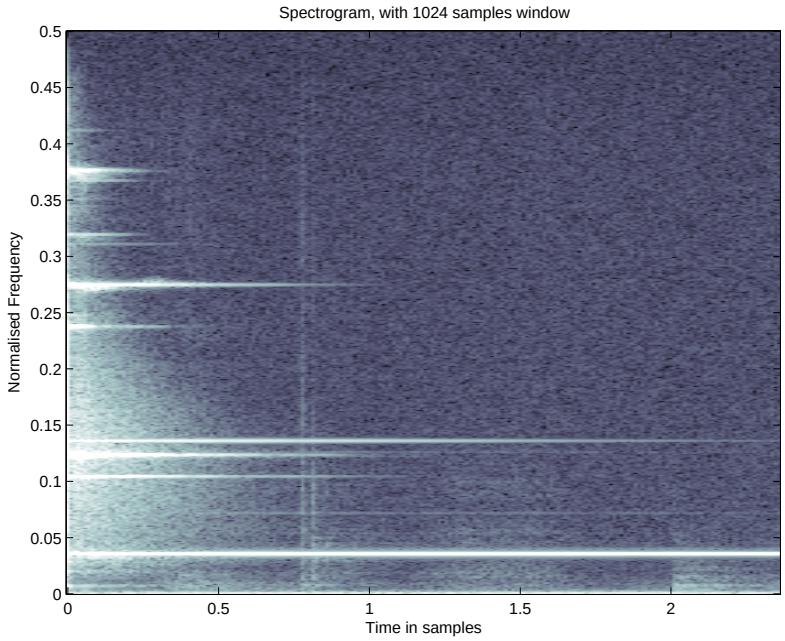

Figure 1.6: The spectrogram of MPEG sample

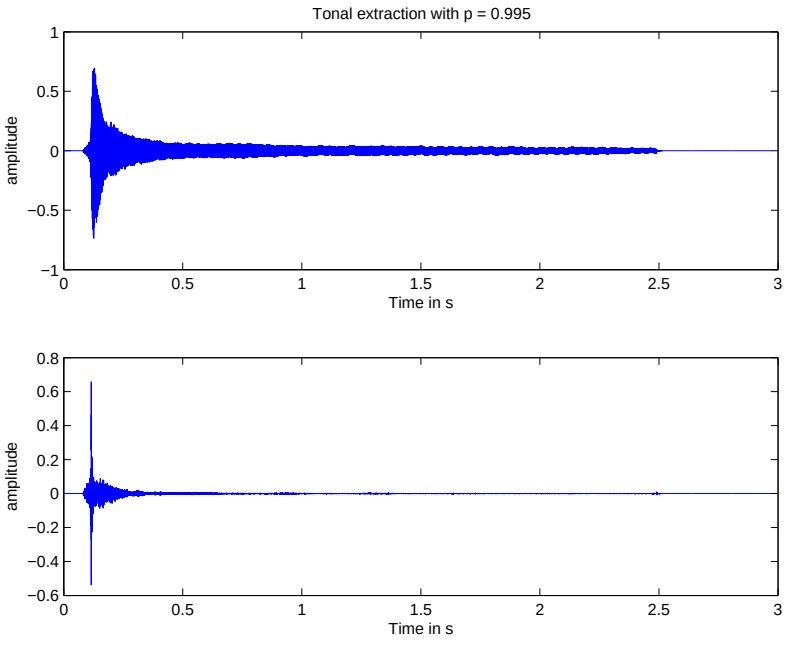

Figure 1.7: The tonal extraction: the up picture shows the tonal part, the below one shows the residual

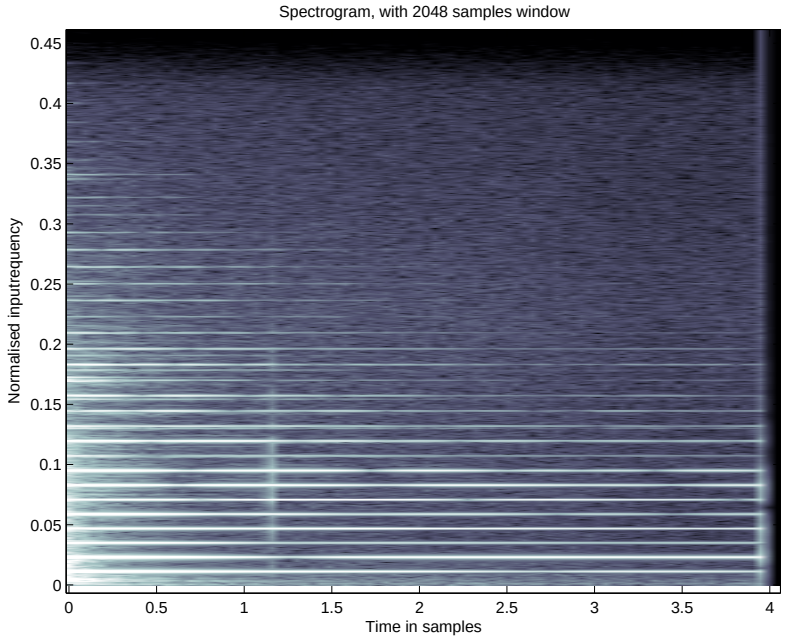

Figure 1.8: Spectrum of the original piano sample

artefact of all the block based audio codecs (See annexe 2 for a quick explanation of the pre-echo effect). The thresholding on the MDCT coefficients induces a spread of the 'quantization noise'. It is due to the thresholding and not to the quantisation. It is also known as "transients smearing", a big artefact of classic phase vocoder based transforms.

The second sound is a piano note. I take this sound because this instrument is known to be difficult to model: there are some noise coming from the hammer, the finger from the pianist, and there are a lot of partials, which are not necesserally in an harmonic order, perticularly in the high spectrum (see figure 1.8)

And as we may expect, the results are a lot more circumspect. I first try a tonal extraction with a 90% quantile (sound 2): there is no really extraction. In fact, except some artefacts at the sound's queue (warbling effect, see later for the study of this artefact), the tonal part sounds the same as the original one. With more heavy percentage, the artefacts became more and more presents. There is some high frequency artefacts, which can be seen on 1.9, begans quite randomly, and end quite randomly: it is the warbling effect. It is due to the fact that some partials can be present in one frame approximation, and not in an another, depending on the phase value at the beginning of the frame.

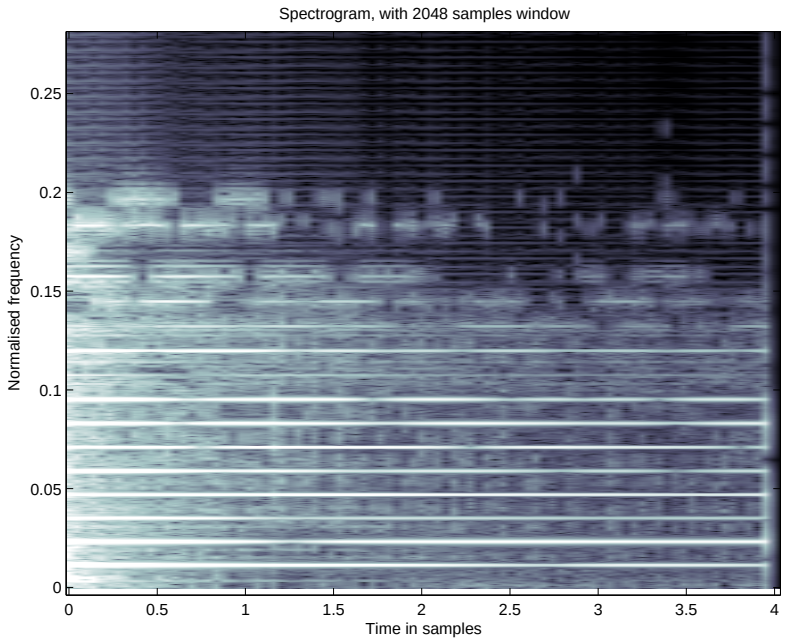

Figure 1.9: Spectrum of the tonal part of the piano sample

It is worse with more complex sounds. The warbling effect is previsible: it is normal with the techniques we are using. But the extraction, on a "perceptual" point of view, is really bad: the problem is that all the transients which are burst of energy will have big coefficients in the MDCT, even if there is no steady-state components in it. To illustrate this point, I used a congas audio loop, and applied different values for the quantile p. For the heaviest parameters values, one ends with a low pass filtered sound, still transients in it, and completely useless for audio transformations.

In conclusion, the tonal extraction doesn't sound good and doesn't work at all. We shall not be surprised here, as the original framework was developed for audio coding. In this context, the extraction is good if the compression rate is good, and is hardly linked to the perceptual separation; on the contrary, we want to use this framework for audio effects, perticularly time scaling, etc...

All those artefacts can be improved by some solutions I will present in the next chapter.

1.3.3 Transients extraction

When we hear music, we are used to think about it in events; for example, a note change, a percussive sound, etc... One generally call transients *isolated fast changes* in sound; it is quite different from onsets, which one can define

as the *beginning of a musical event*. Defining transients and onset in a useful DSP way is quite hard (see [CD03] for an overview about transient and onset detection).

The L.Daudet approach was to define transient not from a perceptive or musical view, but rather from structures in the Discrete Wavelet Transform of the signal. DWT is a part of the more general Wavelet transform framework, which receives great attention the last 15 years (see annexe 3 for an introduction about wavelet transform and [Mal98] for a complete overview). Is is successful in the analysis of signals which are fast varying, or localised in times, ie signals which Fourier Transforms fail to analyse well.

There are several types of Wavelet, and here, we are using the Discrete Wavelet Transform, which decomposes any discrete signal on a wavelet basis. There are a huge amount of litterature about wavelet bases and frames, here we shall only consider the orthogonal ones. The wavelet bases vectors are more and less localised in times. Some of them have compact support: as we want the maximum time localisation, we shall here use the most compact ones. Haar basis is the most compact orthogonal basis. We will limit ourselves to Daubechies basis [9].

The idea here is quite the same than for the tonal part, ie transients are signals which are well represented in wavelet bases, so an thresholding should give a good transients extraction. The threshold estimation is exactly the same than for the tonal part. For simplicity's sake, I used here a global wavelet transform on the whole signal, which of course is not applicable for practical implementations. We will see later a way to do the transform in blocks, without any edge effect; simply divide the signal into several blocks and computing their dwt is not a good idea, as the filter bank used in dwt (computed thanks to the Mallat algorithm) needs some samples of the previous and next blocks.

The transient extraction is implemented in **transExtract.m** , and is pretty the same as the **tonalExtract.m** , except the dwt, of course. The dwt is also a personnal implementation of the wavelet transform (Mallat algorithm). The transient extraction is done on the non-tonal part, which is the original signal minus the tonal part.

It is a bit difficult to judge the audio quality. Except at the beginning, the residual (ie the nontonal minus the transients) sounds like a white noise. The beginning of the noise is a bit tonal, but as the transient extraction didn't pick it, it is good. There is also a pre-echo effect, but it comes from the tonal extraction (since the transients extraction is done on the non tonal part).

With more complex sounds, the results are not that good, but it mainly

[9] which main property is having the most support compact for a fixed number of vanishing moment; the number of vanishing moments and the compactness of the support is linked

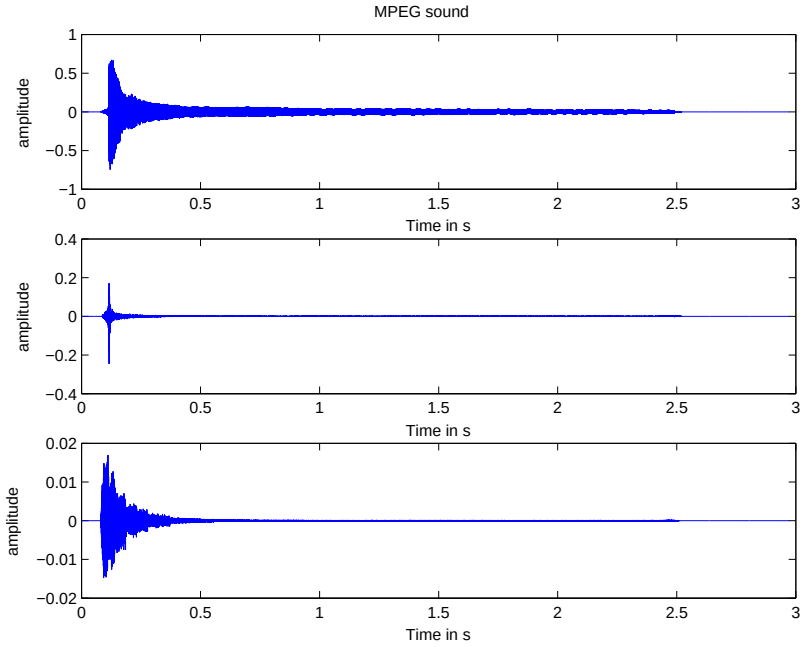

Figure 1.10: Tonal and transients extraction for the MPEG sample (take care to scale effects)

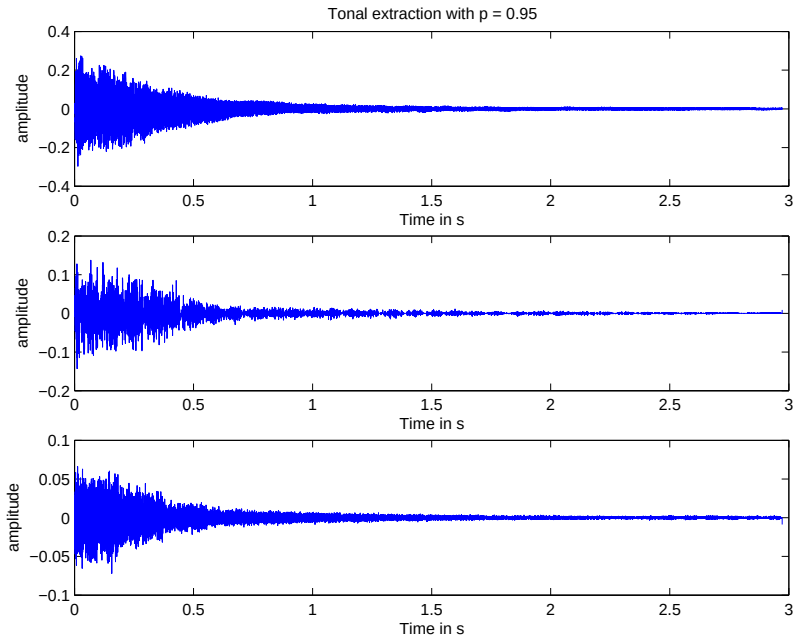

Figure 1.11: Tonal and transients extraction for the piano sample(take care to scale effects)

comes from the tonal extraction. It is rather difficult to judge the transients extraction, as the non-tonal part iself is bad. Most problems are coming from the bad tonal extraction. The scheme used here detects without problem the transients ; that's why I focused on the tonal extraction.

1.3.4 Noise modeling

The residual after the two extractions should be a white noise. It has already been pointed out in [Dau00] that for a large class of stochastic signals, the difference between residual from such hybrid frameworks and Gaussian white noise are negligeable

There are several methods to estimate the mean and variance of the Gaussian density for each frame. The cepstral method and LPC estimation suppose that for each frame, the signal is a stationnary process, and try to model the spectral enveloppe. The third method, which models the residual as gaussian processes through a bank filter, uses the critical bands hypothesis, that is the hear cannot perceive energy variations within some frequency ranges.

I didn't study much the noise modeling.

Chapter 2

Hybrid representation: some improvements

The former framework suffers from several problems, and this chapter presents a few directions to follow. All these methods are studied with the idea that we want to stay in the transform domains as much as possible.

2.1 The tonal part

The tonal extraction must be improved if we want to utilize our framework in a useful way. We have already seen the problems with the extraction process:

- Pre-echo: this is due to the thresholding itself. One good way to reduce it is to use shorter windows, but we loose frequency resolution, which is a bit disapointing if we want to extract the tonal part, where the partials exact values are important. I tried to implement a windows switching scheme analogous to the one used in MP3 and AAC: the basic idea is to switch to smaller windows when transients are detected.
- Warbling effects: it is due to the fact that a perfectly stable partial, because of the phase, can have a big coefficients in one MDCT frame, and one much smaller in the next frame.
- Too many transients in the tonal part: this problem is linked to the former one, and made me consider studying a complex MDCT, to use the phase information to track the tonal part.

2.1.1 Window switching

The trade-off between time-resolution (which demands short windows) and frequency resolution (which demands longer windows) is well known (see

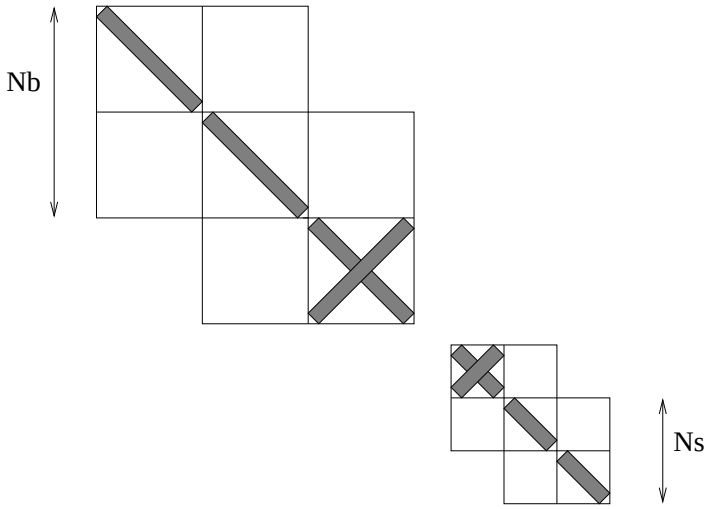

Figure 2.1: matrix representation for the block switch. Gray area are non-zero elements

B). One can increase the frequency resolution for a given window's size by interpolation schemes (see chapter3), but their precision is limited by the hypothesis made on the signal. One can also use zero padding to increase frequency resolution [1] but this is highly computionally intensive.

The MPEG-1 layer 3 adresses this problem by using a windows switching scheme. The idea is first to detect the transients, and then to adapt the window's size: generally, one doesn't need a good frequency resolution when there is a transient, and so one can use smaller windows, which provides better time resolution. The MDCT was conceived to enable window size switching [MB03].

There exists several techniques for switching block's size. The first one, used in the MPEG-1 layer 3 and , was invented by Edler. Let's see the normal matrix structure for MDCT/IMDCT, as reminded in B for the case of window's size switch; we note Nb the size of the big blocks, and Ns the size of the small blocks. Overlapping matrices of size Nb leads to identity matrices, except the last $Nb/2$ block before the block's size switch, which also has some anti-diagonal non-zero elements (see the figure 2.1), and is meant to be overlapped with a $Ns/2$ block. This $Nb/2$ block is equal to $W^{S_L}(\mathbf{1}+\mathbf{J})W^{A_R}$, with the same notations than in B.

The problem is that the antidiagonal of the big block is too big to be cancelled out by the following short block. The idea of Edler method is to put some values of the windows W^A and W^S to 0, so that the antidiagonal

[1] the N point fft of a N point signal is invertible, ie contains all the information of the signal. But when the FFT's size increases, the spectrum "tended" to the spectrogram

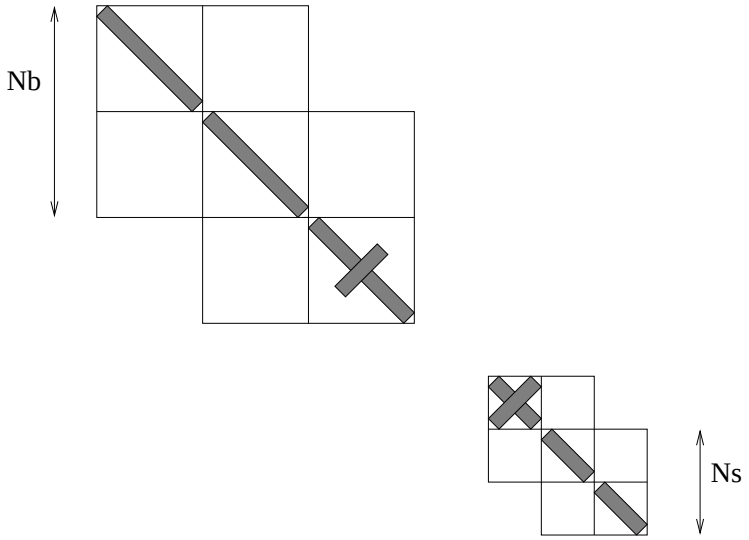

Figure 2.2: matrix representation for the block switch, after zeroing some elements in the long block's windows

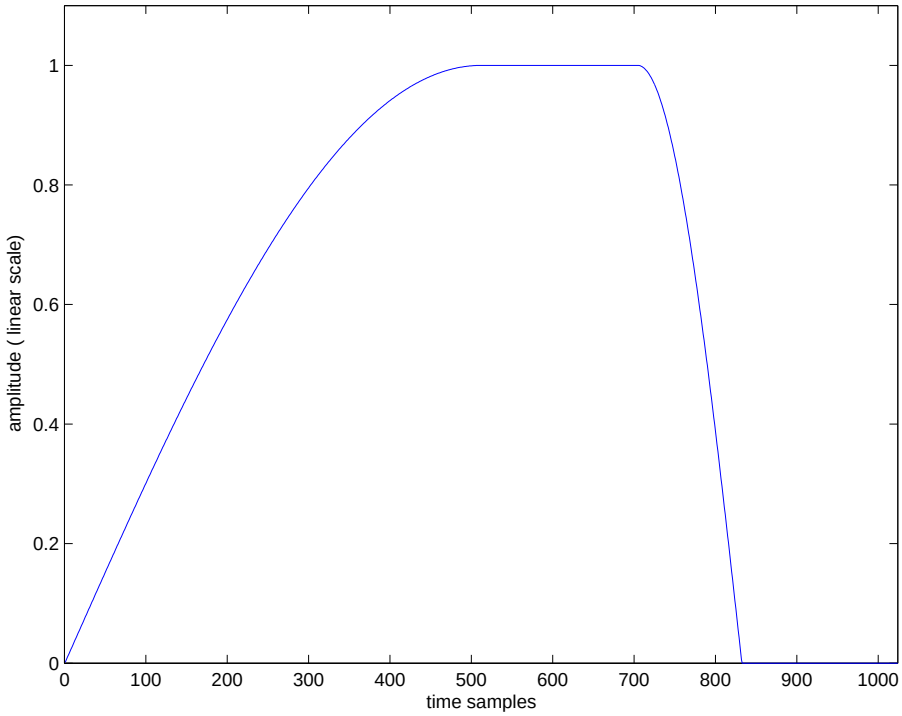

Figure 2.3: Transition window from long block (N = 1024) to short block (N = 256) with the edler method

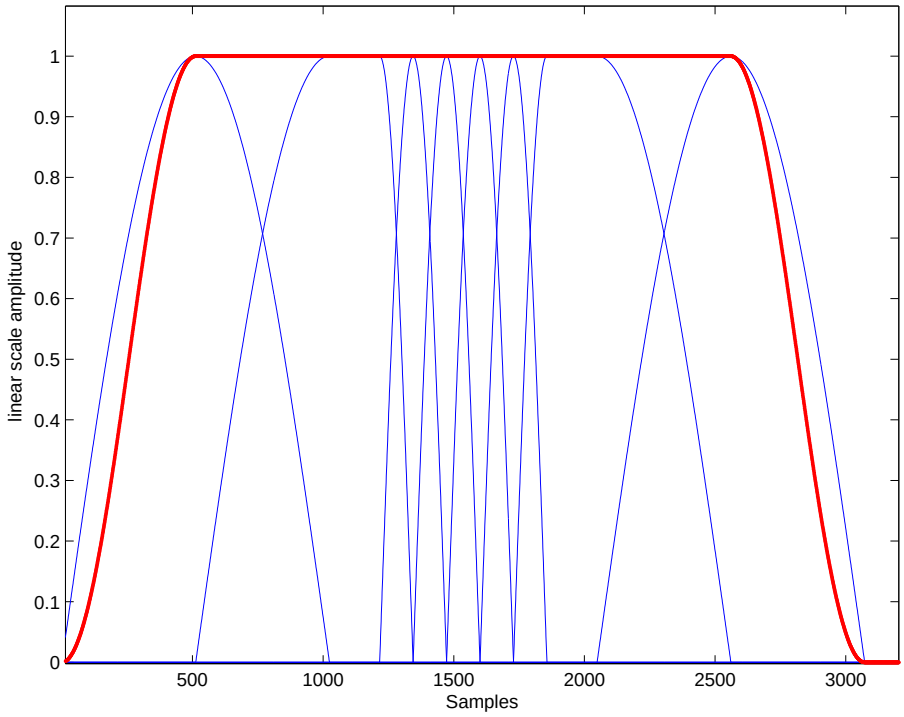

Figure 2.4: Example of a transition window: 1024 to 256 samples. The red part shows the first window condition, ie $w_i^a[n] * w_i^s[n] + w_{i-1}^a[N/2+n] * w_{i-1}^s[N/2+n] = 1$, is respected

part of the big block is as long as the short block's one. As the antidiagonal part is $Ns/2$ long for the short block, each side of the big windows must be zero outside $Nb/4 - Ns/4$; that way, the figure 2.1 becomes as the figure 2.2. For now, the anti-diagonal part is cancelled; to have perfect reconstruction, as the first $Nb/4 - NS/4$ samples of the right side of a long block don't overlap with the next shot block, they must be put to 1. The last part is simply the left side of the short window, so that overlapping it with the next block leads to the perfect reconstruction condition (the first one, analog to the STFT one). The design of the transition window assures the time domain aliasing cancellation; the other conditions for windows are easy to verify (see figure 2.4)

The implementation of this scheme is a bit difficult, because it is rather hard to find where errors are coming from. The details are given in 4; here is a short description: I first implemented a normal MDCT, thanks to **mdct_block.m** and **imdct_block.m**. These tweo scripts implement a single block MDCT/IMDCT with FFT. The hard part here is of course to handle the proper synchrnisation when a block switch has to be done; for simplicity's sake, I decided to fix the behaviour when a transient is detected,

that is:

- One transition window (big to small block)
- Four short windows
- One transition window (small to big block)

And so when a block switch is necessary, a special function is called, which implements the above scheme, and returns where to continue the "normal" MDCT. It should be better to have more flexibility, that is to have different size for short windows, etc... But it becomes very difficult, and is more of an engineering problem.

For the moment, I have some problems to have significant changes in quality with block's size switching. It works well in MP3, but here, as before, we are not aiming at audio coding, but at much higher level transformations. That's why I am thinking about using better transient detection (MP3 uses a simple High Frequency Content bases detector), as presented in [CD03]. I didn't have the time to study and to implement theses methods, though.

2.1.2 MDCT regularisation

The MDCT has a big drawback: it is not translsation invariant. One can show there isn't any convolution theorem analogous to the famous one in Fourier domain for MDCT-like transforms. Why is it a problem for us ? It means that a simple signal like a sinusoid with constant frequency and amplitude can have very different MDCT values, if it begins with a different phase at the beginning of the frame.

Laurent Daudet [LD02c] studied this non-invariant translation and some possible answers. If the input signal is a sinusoid (notation: $x[n] = sin(\omega n + \phi)$), the MDCT coefficients $d_{p,k}$, where p is the frame indice and k the frequency indice, can be approximated with this relation:

$$d_{0.k} \approx \lambda M_f(k) cos(\frac{\pi}{2}(k - k_0) + \psi) \tag{2.1}$$

Where:

- k_0 the integer part of the normalized f ($f = \omega L/\pi$, where $L = N/2$ is the half length of one MDCT block.
- $\lambda = \frac{\sqrt{2L}}{2\pi}$
- ψ is a variable which depends on f, k_0 and ϕ. So it is fixed for our example !
- $M_f(k) = -\frac{sin(\pi f)}{(f-k)(f-k-1)}$

This equation 2.1 shows that d_k can heavily depend on ψ. So, if we apply hard thresholding on the MDCT coefficients, some frames may contain steady state compenents, and some other frames may not, even if the amplitude and frequency values are the same (see the spectrograms of chapter 1, perticularly in the high frequency part).

As I think the threshold method is fundamentally flawed for tonal extraction, I didn't study this method in depth. [LD02c] explains the method and how to apply it to reduce warbling effects for audio coding schemes.

2.1.3 Using the phase information to detect tonal part

Most of tonal extraction's problems come from the fact that amplitude and phase information are hidden in the MDC coefficients. Approximating the tonal part (ie extracting steady state-compenants) by a non linear approximation isn't very appealing from a perceptive point of view: the advantage of non linear approximation by hard thresolding over classic linear approximation is that f can still be well approximated even if f has some local discontinuities (see A). But this advantage doesn't really make sense for tonal elements, which by definition are more or less constant within a certain time horizon.

That's why I begin to study several methods for tonal modelling and extraction, based on sinusoidal models: even if the idea itself comes from the mid eighties, more recent breakthroughs were made in the last years (for example peak picking by "pattern recognition" in [Rod97] or Markov Chains to do peaks tracking [XR93]), for different parts of sinusoidal modelling and additive synthesis. As I wanted to keep a MDCT-like transform, because of its better integration in MPEG and audio-compression frameworks, I decided to use a "complex MDCT", as defined in [Mal99]. Use of the complex MDCT means wer lose some of the advantages (such as keeping the data rate constant; this "complex MDCT" doubles the data rate), but we gain phase information; this transform has one big advantage: it is easy to recover MDCT coefficients from it.

The chapter 3 presents the work, mostly of a bibliographic nature, I've done on sinusoidal modeling.

2.2 The transient part: filtering in the wavelet domain

Discrete Wavelet transforms are not translation invariant: this is due the uniform sampling of the translation parameter of continuous time wavelet transform. In our case, it can be annoying if we want to do some delaying of the transients without going back to the time-domain representation.

Several methods exist here. I studied one method, based on a very

simple algebric principle, as described in [LD02b]. For any signal x, if $W = w_{kk=1...N}$:

$$x[n] = \sum_{k=1}^{N} \alpha_k w_k[n]$$

Filtering the time signal into $\tilde{x}$ with the filter h, if we call α_k the coefficients of $\tilde{x}$:

$$\begin{aligned} \tilde{x}[n] &= \sum_{k=1}^{N} \tilde{\alpha}_k w_k[n] \\ &= \sum_{k=1}^{N} \alpha_k \tilde{w}_k[n] \end{aligned}$$

where $\tilde{w}_k$ is the filtered basis. For any k, one can write

$$\tilde{\alpha}_{k_0} = \sum_{k=1}^{N} \alpha_k < w_{k_0}, \tilde{w}_k >$$

So we can implement any filtering by a matrix-vector multiplication, where the matrix M is a N*N matrix whose elements are $m_{i,j} =< w_i, \tilde{w}_j >$. The interest is that M is sparse for wavelet bases.

I studied this method by implementing some fast wavelet C++ classes; the big problem is that handling sparse matrixes, on a computational point of view, is very, very difficult. It is almost impossible to have faster schemes than the wavelet to time-domain/filtering/time-domain to wavelet domain, for small delays. The technique is more interesting for big impulse response, but here, all we want is delaying, which in itself is trivial to do in time domain.

I tend to think that working with translation invariant wavelet schemes would be more interesting for analysis / synthesis purposes. For example, one can think about matching pursuit methods, with a dictionnary of translation invariant functions. If the dictionary is translation invariant, then the representation itself is translation-invariant too. See chapter 9 of [Mal98] matching pursuit and translation invariant dictionnary. There are also complex wavelet transforms which can be interesting, but I couldn't study them.

Chapter 3

Tonal modelling

3.1 Sinusoidal model

I think that modeling tonal parts with thresholding on MDCT is not very useful for analysis purposes. There are answers for the different issues, but these issues don't exist with "normal" methods which use phase information. That's why I began to study all the issued involved in the sinusoidal modeling: by sinusoidal modeling, I mean not only the original work of X. Serra, but also all the following works on additive synthesis with sinusoidal signals, as the improvements on peaks tracking, peaks detection, etc... [SL] [Rod97] [XR93]

The reader may find the following chapter to be vague. It has been necessary to keep it that way, due to the vastness of the sinusoidal modelling field. Hence, this chapter should be viewed not as an exhaustive report, but more as a set of guidelines or start points for further research.

3.1.1 Presentation of the initial framework

The sinusoidal model assumes that tonal parts of the signal can be modelled with a sum of sinusoids, with constant parameters on frames; parameters changes between frames are interpolated to have smooths transitions.

The framework has several parts:

- first, one must computes the spectrum; it is easily done with a discrete Short Time Fourier Transform.
- After that, one has to find the underlying sinusoids; in the original Serra's work, it is done by finding the maxima in the spectrum.
- Once the peaks are found, we compute their phase/amplitude/frequency by interpolation schemes.
- A pitch is estimated for tonal frames

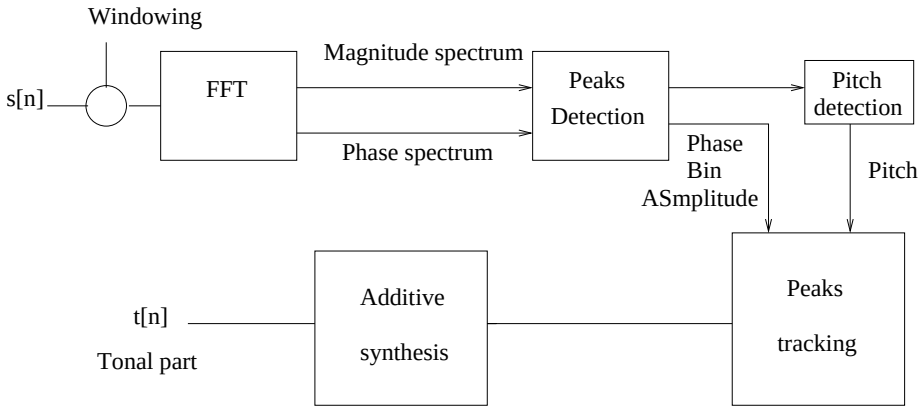

Figure 3.1: Structure of the SMS framework

- Finally, the peaks are tracked with a simple method [Ser97].

The model assumes that the tonal part can be modelled that way, where r is the current frame:

$$s[n] = \sum_{k=0}^{N_r} A_r[n]cos(\phi_r[n]) \quad (3.1)$$

$$= \sum_{k=0}^{N_r} A_r[n]cos(\omega_r n + \psi_r) \quad (3.2)$$

The analysis problem is to find all these parameters. The figure 3.1 shows the process.

3.1.2 Peak detection

Once the spectrum has been computed for the current frame, one needs to find the underlying sinusoids in it. The initial method of Serra was to look for the peaks in the spectrum. The windowing effects are as discussed previously : it is still mostly the trade off between frequency and time resolution which is significant.

Detecting the peaks can done the following way:

- First, fix the number of peaks we want to find to N
- Find all values in the spectrum which are greater than their neighbours
- Sort them, and take the N greater peaks.

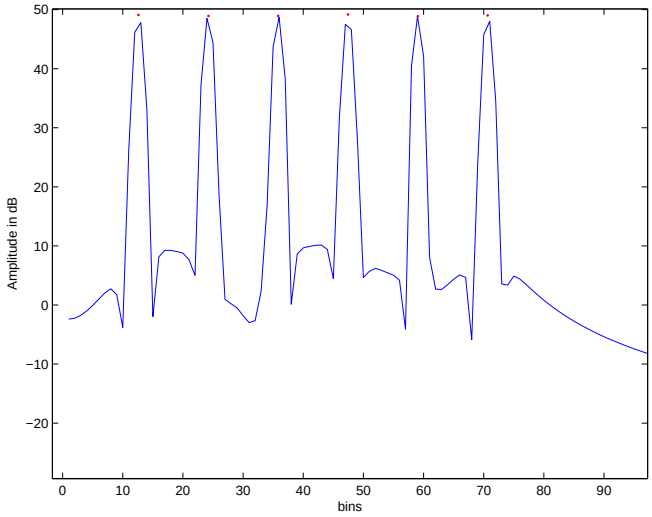

Figure 3.2: Peaks detection for a very simple sound

It is implemented in the **PickPeaks.m** script, which is adapted from the SMS scripts found in the dafx book site [1].

I first tried it for simple artificial signals, like sums of sinusoids. The problem is in the parameter N. First, I computed the peaks with the knwon value of N (ie N is set to the number of sinusoids). Hopefully, it works perfectly for such simple sounds (see figure 3.2). We see the problem for "normal" sounds: peaks in the spectrum, concurrent with our definition do not necessarily indicate the presence of sinusoid. we did, don't mean that there is a sinusoid in it (see figure 3.3)

A more recent approach, presented later, works with the shape of a sinusoid in the spectrum.

It worths noticing that interpolation schemes are used here to give more accurate values for bins and phase; the method is straightforward, it is parabolic interpolation of parameters.

3.1.3 Pitch detection

For harmonic sounds, it can be useful to find the underlying pitch. As I didn't want to restrict myself to monophonic sounds, and methods to find fundamental in polyphonic sounds are a subject of research in itself, I didn't look into that direction.

[1] http://www.dafx.de

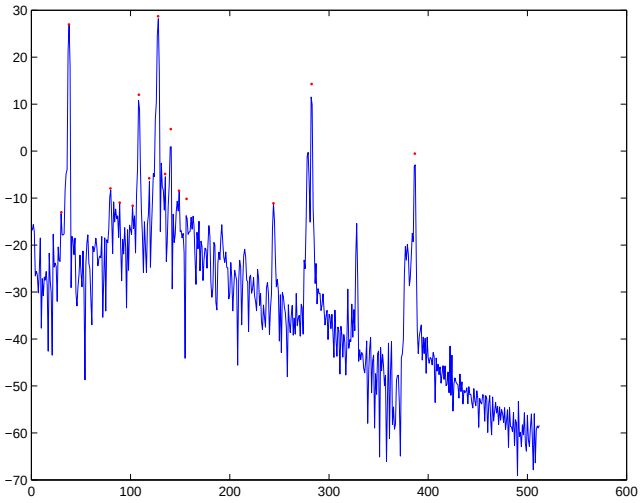

Figure 3.3: Peaks detection for the MPEG sound

3.1.4 Peaks tracking

Once all the parameters of the sinusoids are found, we still need to track them: the idea is that instead of resynthesing directly the sinusoids with the parameters, it is much better to track the partials, and keeping or rejecting the peaks with reference to their trajectory.

I studied one method, which is the one originaly found in the SMS framework, but I don't really like it, because it demands too much knowledge about the signal, and need too much feedback from the user. The idea is to follow the peaks with some "guides": if some partials of one frame are too far from the previous ones, the guides are killed, which mean that the partial is considered ended. In a similar way, births are defined.

Much more sophisticated methods exist, which use Markov models [XR93], but I couldn't study them.

3.2 Improvements in the framework for tonal extraction

If I had more time, the two aspects I really would want to study are Markov modeling for peaks tracking, and some techniques developed at IRCAM by the Analysis/Synthesis team of Mr Rodet. I just present here the reasons why I would like to go further in these directions.

3.2.1 Sinusoidal model improvements and waveforms models

Peak detection and estimation

As we saw earlier, peak estimation it is a weak point of the initial framework. The approach of X. Rodet to improve them is the following [Rod97]: if $h[n]$ is the window used in the STFT, finding a sinusoid in the frequency spectrum can be seen as a "pattern recognition" problem, or more precisely as finding the spectrum of h in the STFT spectrum. One basic technique to measure if two signals are similar is the correlation [2].

Elementary Waveforms analysis

This method models the sound by using some Elementary Waveforms, called EW. The idea itself is not new: the Formes d'Onde Formantiques is often used in synthesis languages like Csound (the FOF package is even one of the big reason for the sucess of Csound). The idea of this technique is to decompose the signal with some time-frequency atoms called EW, thanks to a matching pursuit algorithm. This kind of ideas could be used for sophisticated tonal modeling.

3.3 With the complex MDCT

Some "complex MDCT" methods were developed recently. The one present in [Mal99], developed by H.S Malvar, has some useful features:

- Fast to compute
- One can recover the MDCT from it (one just need to take the real part of the transform)

It is basically the complex sum of one DCT and one DST (S for sine) multiply by i (where $i^2 = -1$). The idea is to adapt the tonal modeling previously presented to this transform.

[2] It is for example heavily used in simple time-domain time scaling algorothms like SOLA, or WSOLA, which precisely find the best synchronisation point by a waveform similarity approach, based on correlation

Chapter 4

Implementations

Fo different reasons, I had to implement most tools presented before myself. I ended to program C++ classes for discrete wavelet transform, matlab code for MDCT (with block's size switching), DCT, DWT, the initial decomposition for Hybrid decomposition (adapted from L. Daudet's scripts) and SMS framework. I decided to put this part here arbitrarly.

4.1 C++ classes for dwt

There were several goals here when I implemented these classes: being reusable (easy to use) and fast. There are 2 algorithms for wavelet decomposition, the Mallat algorithm and the lifting scheme. The last one is theoratically faster, but depends on the wavelet basis.

I first tried to be "compatible" with wavelab, which was rather difficult, as the implementation of wavelab was extremely bloated (pre ANSI C). As DWT is not translation invariant, different implementations lead to different results, one has to follow the exact same scheme if one wants the same results with wavelab and C++ implementation.

There are basically two classes, one to handle the the QMF filter, and one which computes the dwt itself. One first have to construct the QMF class, then the DWT class, which is passed a QMF filter, the length of dwt and the number of level of decomposition. As the constructor handles the initialisation, once it is sucessfully called, one can begin to compute the dwt. There are several advantages for this scheme, and drawbacks:

- there is no dynamic allocation during the processing itself, so it is suited to Real-Time processing (it can be easily modified to be thread safe)
- Having one different class for QMF filter enables to extend it independantly of the dwt computing itself

- Having different sizes or different level require to construct several dwt classes

There are two implementations of dwt: one which is compatible with wavelab, and so a bit difficult to read, and one which implements the classic Mallat algorithm with circular convolution to handle the edge effext [1].

Several efficiency tricks were used (on x86 architecture, I do not know it these tricks can be apply for other architectures)

- All arguments passed to internal function are copied first before use. That way, most compilers can handle much better optimizations (copying pointer arguments for arrays can lead to a 50% faster code for convolution !)
- Instead of calling a generic convolution function for any size of filter, there is a test at the construction, and a function pointer points to optimized versions, using loop unrolling where possible.
- Several specific implementations have been written for classic QMF filter: if the values of the filter are directly written in the code, it can be much faster (I suppose for cache reasons, but I am not sure).

All these optimizations are of course transparant to the user, because easy of use was more important for me than efficiency. At the end, for 32 bits numbers, Daubechies 4 and 4 levels of decomposition, I managed to have a 25000 cycles consumption for 1024 input signals, which mean that one can do around 40000 dwt per second on a recent enough computer. This was a good reason to stop studying filtering in wavelet domain !

To understand the exact scheme of the processing, one can read the matlab sources of **FWT_PO.m** and **IWT_PO.m**, and **dwt.m** and **idwt.m** for a much easier implementation.

The C++ code was compiled with the GNU C++ compiler on linux. I used two versions, 3.2 and 3.3, and used the -O3 flag (heavy optimization). Performances differences were sensible between the two compilers (around 10% for the exact same code). I think there can be further significant speed improvements:

- the memory bandwidth is often a bootleneck for this kind of algorithms. There too much copy which can be removed.
- using architecture specific instructions, like MMX memcpy optimization and SSE/SSE2 vectorization of the code. This requires much more difficult code, which would depend of the architecture and the operating system, and I would lose the ISO C++ advantage, so I didn't use it.

[1] I learnt later that it existed far better methods to handle edge effets , as pointed in [Mal98]

The FB_WT class is in the namespace dsp, as the QmfFilter. The type used for computation is DataType; it is choosen at the compilation time, it is just a typdef on float or double. I could have used templates, but it demands to compile the class every time, as g++ still doesn't support pre-compiled header.

The following code shows that using the classes is not difficult:

```
const int  N = 4; // size of the qmf filter

// One creates one QmfFilter object of size N (Daubechies)

const dsp::QmfFilter qmf(N);

int Ni = 1024; // size of the input signal:
int No = 1024; // size of the output;
int lev = 4;

FB_WT::DataType *in, *out, *in2;

// Allocate the input/output arrays:

in  = new FB_WT::DataType[Ni];
out = new FB_WT::DataType[No];

// Create the wavelet object
dsp::FB_WT w (Ni, lev, qmf);

// Now, everything is normaly initialised, one can compute dwt:
// one computes the dwt of in, and put it i out.

w.do_FWT(in, out);
```

4.2 DCT,MDCT

4.2.1 DCT

For a signal $x[n]$ defined for $n = 0 \dots N-1$, the DCT I is defined as the DFT of $\tilde{x}$ defined as:

$$\tilde{x}[n] = \begin{cases} x[n] & \text{for } 0 \neq n < N \\ x[-n-1] & \text{for } -N \neq n \neq -1 \end{cases}$$

As $\tilde{x}$ is symmetric around -1/2, the DCT I is always real. Is can be written as the expansion on

$$
\begin{aligned}
e_k &= \lambda_k \sqrt{\frac{2}{N}} cos\left(\frac{\pi k}{N}(n+1/2)\right) \\
\text{with} & \\
\lambda_k &= \begin{cases} 2^{-1/2} & \text{if } k=0 \\ 1 & \text{if } k \neq 0 \end{cases}
\end{aligned}
$$

So one can write any signal of $\mathbb{C}^N$:

$$
\begin{aligned}
\hat{x}_I[k] &= \lambda_k \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi k}{N}(n+1/2)\right) && (4.1) \\
x[n] &= \frac{2}{N} \sum_{k=0}^{N-1} \hat{x}_I[k] \lambda_k \cos\left(\frac{\pi k}{N}(n+1/2)\right) && (4.2) \\
& && (4.3)
\end{aligned}
$$

In a similar way, one can define the DCT-IV, and it is written:

$$
\begin{aligned}
\hat{x}_{IV}[k] &= \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi}{N}(k+1/2)(n+1/2)\right) && (4.4) \\
x[n] &= \frac{2}{N} \sum_{k=0}^{N-1} \hat{x}_{IV}[k] \cos\left(\frac{\pi}{N}(k+1/2)(n+1/2)\right) && (4.5) \\
& && (4.6)
\end{aligned}
$$

which is easy to compute from the FFT. It is done by splitting the signal x between even $e[n]$and odd indices $o[n]$. By an analogue scheme, DCT-I are computed from DCT-IV. There are more powerful theorems which enable to compute a N DCT by a N/4 FFT.

4.2.2 MDCT

The MDCT is relatively easy to compute with FFT. If we note w the window, the N MDCT $X[k]$ of a signal $x[n]$ can be written:

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]w[n] \cos\left(\frac{2\pi}{N}(n+n_0)(k+1/2)\right) \\
&= Re\Big\{ \sum_{n=0}^{N-1} x[n]w[n] e^{-i\frac{2\pi}{N}(n+n_0)(k+1/2)} \Big\} \\
&= Re\Big\{ e^{-j\frac{2\pi}{N}n_0(k+1/2)} \sum_{n=0}^{N-1} [x[n]w[n] e^{-i\frac{2\pi n}{2N}}] e^{-i\frac{2\pi kn}{N}} \Big\}
\end{aligned}
$$

SO, to compute the MDCT, we just need to :

- pre-widdle the signal by $e(-i\frac{2\pi n}{/}2N)$
- compute the N point FFT
- post-widdle the data by $e^{-j\frac{2\pi}{N}n_0(k+1/2)}$
- Taking the real part

The IMDCT is decomposed with an analogue scheme. The scripts **mdct_block.m** and **imdct_block.m** implement theses algorithms. A MDCT on the whole signal is done by splitting the signal in N blocks, and by calling these functions for each block.

4.3 Hybrid representation

I first want to thank here L. Daudet who gave me all his scripts for the hybrid decomposition. There are two functions:

- **tonalExtract.m** which implements the mdct thresholding, with the probability estimation. The prctile script included in the statistical toolbox is used, but the function should be implemented in C/C++, as this function is not difficult, and very slow in matlab (several for loops). It uses my scripts for the MDCT decomposition.
- **transExtract.m** which implements the same scheme for the transients, with the wavelet decomposition of Wavelab.

All matlab scripts are much commented, it shouldn't be difficult to re-use them for a further use.

Appendix A

Non linear approximation in orthonormal bases

This quick introduction to linear and non-linear approximation will just present some strong results, which are useful to understand why DCT and MDCT are used over FFT in coding schemes.

Here, H will be an Hilbert space with a basis e_k , ie:

- a complex vector space.
- has a scalar product, denoted ¡¿: it enables us to speak about orthonormal bases and norm (derived from the scalar product).
- is complete: it is handful to write for any $f \in H$

$$f = \sum_{k \in \mathbb{Z}} e_k \alpha_k$$

(ie: if $f \in H$, it can be written as an "infinite linear combination")

If H has an orthonormal basis, one can decompose any $f \in H$ like this:

$$f = \sum_k < f, e_k > e_k \tag{A.1}$$

and

$$||f||^2 = \sum_k | < f, e_k > |^2 \tag{A.2}$$

In fact, in most approximation problems, only A.2 is useful. It is called the Parseval theorem for Fourier Bases. We can extend the notion of orthonormal bases to have a relation similar to, called frame. Frame theory is really useful in wavelet theory, for example: the idea of frame operators is how and at which conditions one can recover a vector from its scalar products with a set of vectors. Some ways to see frame theory is as a theoritical extension of the Shanon/Whittaker theorem or as a generalisation of orthonormal basis . See the chapter 5 of [Mal98] for a good introduction to frame theory.

A.1 Linear approximation

An obvious way to approximate f is to take the first M inner products, and so, if $\hat{f}_M$ is the M^th order approximation of f:

$$\hat{f}_M = \sum_{k}^{M-1} < f, e_k > e_k \tag{A.3}$$

The error $\epsilon_M = ||f - \hat{f}_M||$ is:

$$||\epsilon_M||^2 = \sum_{k=M}^{\infty} | < f, e_k > |^2 \tag{A.4}$$

It is easy to prove that this error tends to zero when M tend with ∞. But more interesting: *we can link the approximation error with the behaviour of the inner products* $< f, e_k >$. So, if we understand how these inner products behave with k, we can understand how the error approximation behaves. A strong theorem proves that this error behaves like $M^{s/2}$ if f is in a certain functionnal space:

$$W_{B,s} = \left\{ f \in H : \sum_{m=0}^{\infty} m^{2s} | < f, e_k > |^2 < \infty \right\} \tag{A.5}$$

In special bases, like Fourier basis or wavelet basis, this space becomes a *Sobolev space*, which is strongly linked to the smoothness of f. The more f is differentiable, the bigger s is, so the better the approximation is.

This approximation is clearly linear, as the inner products don't depend upon f. It is also referred as an *a priori* approximation.

A.2 Non linear approximation

Even if we want to keep a similar scheme, ie approximate any element of H by a linear combination of the basis set, the former approximation is clearly suboptimal. It would be a lot better to take the M vectors which maximise the inner product $< f, e_k >$, as it would minimize the error approximation. The approximation becomes *a posteriori*, and so becames non-linear (the M vector depend on f).

It is easay to write this kind of approximattion as a threshold on the value on the inner-products. Why is it useful ? Because this way, one can approximate f in a "good" way, even if f has some discontinuities. Links between the behaviour of ϵ_M and f are by far more difficult than in the previous section, and we won't present them here. To sum-up, *f can have some discontinuities, and still being "well" approximate by a few basis vectors.* This feature can be useful for transients signals, which generally present some discontinuities

Appendix B

Block based transforms: DCT and MDCT

Fourier transform is the most widely tool to do the time to frequency mapping. The Fourier series enable to reconstruct perfectly the signal from the Fourier coefficients if the signal is time-limited [1].

In real-life application, one want to deal with time *and* frequency limited signals, so one can use discrete sample of the time and the frequency domain instead of continuous signals. The question which arise here is: can we do it without any loss in the signal ?

B.1 Block Fourier transforms

B.1.1 Windowing effect

Let's x be a band limited signal, and that the sampling rate F_s is high enough to work with a sampled version of x, $x[n]$. We want to work with reasonably small blocks of x (ie a few hundred a samples), from $t = 0$ to $t = T$, so we window it with a rectangular window w_R of size T. Is this time-limited version still frequency limited ?

The figure B.1 shows spectrum of the rectangular window. The windowed signal is the convolution between the spectrum of $x[n]$ (the non time-limited version) and the rectangular window's spectrum. As the spectrum decays slowly, there is an high probability that some frequencies outside F_s are non negligeable, which means some frequency aliasing problems.

With the previous section, we know that the decay of the spectrum (that is, the decay of $< w_R, e_k >$, where e_k is the Fourier basis) is linked to the smoothness of w_R. So, if we build more smooth windows, the windowing effect will certainly produce less frequency aliasing. And indeed, as the figure

[1] one can consider this property as a Whitaker theorem in the time domain: if f is time limited, one can reconstruct the signal with a sampled version of the spectrum of f

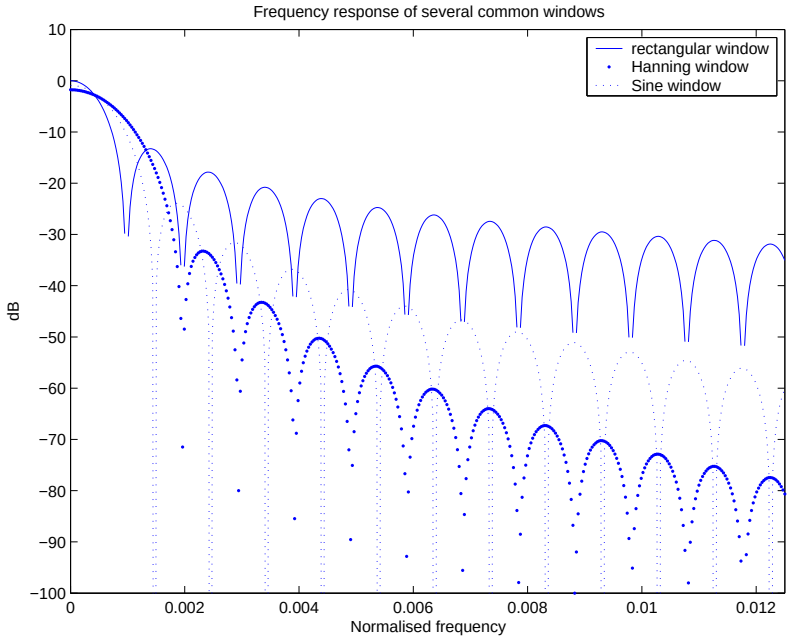

Figure B.1: Spectrum of rectangular, Hanning and Sine window

B.1 shows, Sine window and Hanning window are much better considering that point.

The N points Sine window w_s is defined as:

$$w_s[n] \quad = \quad sin[\pi(n + \frac{1}{2})/N] \quad \text{for n} = 0 \ldots \text{N-1} \tag{B.1}$$

Even if the main lobe around $f = 0$ is wider, the spectrum decays much faster, so time-limiting a signal with that window won't spread frequencies, which could cause some aliasing. The size of the main lobe is proportional to 1/T, ie the window's length: longer windows means better frequency resolution, whatever window used.

Kaiser-Bessel window is another type of window, which is defined so that the trade-off between main lobe's size and spectrum's decay can be adjusted. An N point Kaiser window w_{kb} is defined as:

$$w_{kb}[n] \quad = \quad \frac{I_0\left(\pi\alpha\sqrt{1.0 - \left(\frac{n-N/2}{N/2}\right)^2}\right)}{I_0(\pi\alpha)} \quad \text{for n} = 0 \ldots \text{N-1} \tag{B.2}$$

Where I_0 is the 0^th order modified Bessel function:

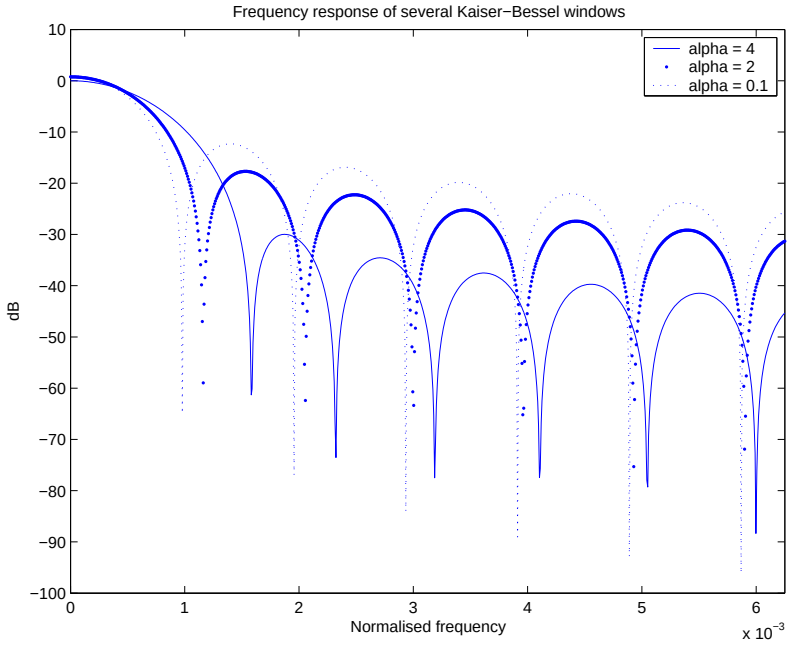

Figure B.2: Spectrum of Kaise-Bessel windows for different α

$$I_0(t) = \sum_{k=0}^{\infty} \left(\frac{(x/2)^k}{k!} \right)^2$$

It is rather difficult to compute the spectrum of this window, but B.2 shows the representation of the spectrum for different α values.

Sine and Kaiser-Bessel windows are used in a modified version for MDCT transforms.

B.1.2 The DFT

The Fourier transform becomes the Fourier series expansion for continuous time-limited or periodic functions: we can reconstruct the original signal from a proper discrete version of the spectrum. If the signal is band-limited, the discrete spectrum becomes a finite set of values. With the usual frequency normalisation, it means that:

$$x[n] = \frac{1}{T} \sum_{k=-\infty}^{\infty} X[k] e^{2j\pi kn/N} \tag{B.3}$$

$$= \frac{1}{T} \sum_{k=0}^{N-1} X[k] e^{2j\pi kn/N} \tag{B.4}$$

More, as the signal is band-limited, one can work with a discrete version of it, which means that the Fourier Spectrum becomes, written in a naive way, with T_s the sampling period:

$$X[k] = \int_{T/2}^{T/2} x(t)e^{-2j\pi kt/Tdt} \quad \text{(B.5)}$$

$$= T_s \sum_{n=0}^{N-1} x[n]e^{-2j\pi kn/N} \quad \text{(B.6)}$$

These two equations, B.4 and B.6, define what is called the Discrete Time Fourier transform. It is often computed with the FFT, which computes the DFT in a much more efficient way than the direct form.

B.1.3 DCT as a way to reduce block artefacts

We know thanks to A that the high frequency content (the $< f, e_k >$ for big values of k, with the same notation as in A) depends on the smoothness of f. But taking blocks of the signal $x[n]$ means windowing it with a rectangular window, which is not even continuous.

Let's imagine a square integrable function f over the time [01]. If we take its Fourier transform on this interval , the Fourier Serie expansion of the function is given by the forumula:

$$\hat{f} = \sum_{k=-\infty}^{\infty} < f(u), e^{2j\pi ku} > e^{2j\pi ku} \quad \text{(B.7)}$$

$\hat{f}$ is a periodic function of period 1, which is equal to f over [01]. But if $f(0) \neq f(1)$, $\hat{f}$ is discontinuous over [01], so there are some high frequency content, even if f is smooth.

That's why some other transforms were conceived, like the Discrete Cosine Transform I. Instead of taking the Fourier transform of f directly, it periodizes f over $[-11]$, so that the Fourier expansion is continuous over $\mathbb{R}$.

That's why some different transforms were conceived, like the Discrete Fourier Transform. The idea is the following: instead of taking directly the DFT of f, one takes the DFT on a 2 periodized function $\tilde{f}$, defined as:

$$\tilde{f}(t) = \begin{cases} f(t) & \text{for } t \in [0:1] \\ f(-t) & \text{for } t \in (-1:0) \end{cases}$$

That way, $\tilde{f}$ is continuous on $\mathbb{R}$. This transform has two advantages over DFT: it is real, and the decay of $\hat{f}$, the Fourier transform of f, is greater (which also means it is best approximate with linear approximation). The transform can be written as the expansion on the orthonormal basis:

$$
\begin{aligned}
e_k &= \lambda_k \sqrt{2} cos(\pi k t) \\
\text{with} & \\
\lambda_k &= \begin{cases} 2^{-1/2} & \text{if } k = 0 \\ 1 & \text{if } k \neq 0 \end{cases}
\end{aligned}
$$

One can also define different DCT, like the DCT IV. This one has the main drawback than the normal DFT, but its coefficients are easy to compute with FFT, and it one can deduce the DCT I coefficients from the DCT IV coefficients.

B.1.4 Block Fourier transform

As pointed out, we want to work with block of small sizes, so we have to divide the signal in successive blocks. Fourier block bases are not very good, because they introduce some discontinuities; DCT are better, but still, it far from ideal. As already seen before, in the windows discussion, using smooth windows enable us to avoid this kind of artefact. But one theorem, the Balian-Low theorem, shows that there is no differentiable window g with a compact support such that

$$
\left\{ g(t - nu_0) e^{jk\xi_0 t} \right\}_{k,n \in \mathbf{Z}} \tag{B.8}
$$

is an orthonormal basis of $\mathbf{L}^2(\mathbb{R})$. There are two solutions: either use some overlapp to have perfect reconstruction (in that case, there is no basis anymore. Trasnforms like STFT are highly redundant), or lapped projectors, which use also overlapping, but without being redundant (ie one can build ortonormal basis).

B.2 MDCT

B.2.1 Construction of an overlapped basis

Two approachs exist to build overlapped bases. The first one was discoverd by Coifman and Meyer for continuous time functions (see the eigth chapter of [Mal98] for a complete presentation of this method), the second one was discovered earlier by Malvar for discrete time signals. From my point of view, the second one is more straitghforward to understand, the first one is better to understand the reasons why it exists (I generally think most theorems in DSP are very difficult to understand; not that the proof itself is difficult, but rather that it is difficult to understand *why* it works. I think that continuous theroems are by far more adapted for a *description* of

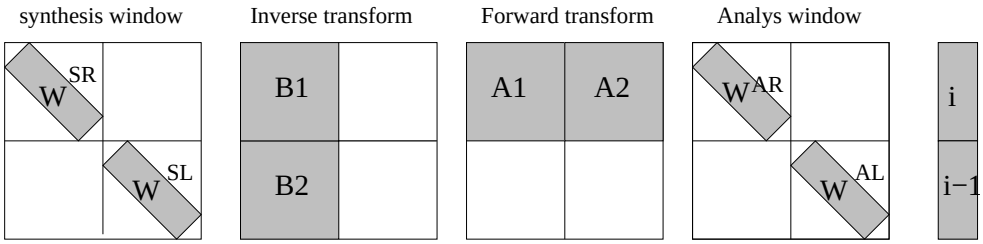

Figure B.3: Matrix expression of a lapped transform

the phenomenon, whereas discrete theorems are of course more adapted to applications [2]). I will present here the Malvar matrix approach, as it give an easy way to understand block's size swithching, as explained in 2. The following explanation is largely due to the MDCT chapter of [MB03]

The principle is to build a transform which maps N samples into N/2 "frequency" samples, and N/2 "frequency samples" into N time samples, so that overlapping inverse transformed block leads to perfect reconstruction, thanks to time-domain aliasing cancellation [3]. One can write this transform as shown in B.3. W^{AR} and W^{AL} are the right and left part of the analysis window; each part is N/2 long, and so the input block of the signal is N long. As we want to map them into N/2 blocks, the forward transform has zeros values in the N/2 last rows; for analog reasons, the inverse transform has the N/2 last columns to zero. At the end, W^{SR} and W^{SL} are the synthesis windows. All the process, ie the analysis windowing followed by a forward transform, an inverse transform and synthesis windowing can be written that way:

$$\begin{pmatrix} W^{S_R} & 0 \\ 0 & W^{S_L} \end{pmatrix} \begin{pmatrix} B_1A_1 & B_1A_2 \\ B_2A_1 & B_2A_2 \end{pmatrix} \begin{pmatrix} W^{A_R} & 0 \\ 0 & W^{A_S} \end{pmatrix}$$

Which can be rewritten:

$$\begin{pmatrix} W^{S_R}B_1A_1W^{A_R} & W^{S_R}B_1A_2W^{A_L} \\ W^{S_L}B_2A_1W^{A_R} & W^{S_L}B_2A_2W^{A_L} \end{pmatrix}$$

By overlapping such block transforms for two sucessive blocks i and $i+1$, we obtain a N*N matrix with 4 N/2*N/2 blocks, where the upper left and the bottom right blocks contain overlapping values from block i and $i-1$. If we want perfect reconstruction, this matrix must be identity. Though, we obtain the two following equations, which must be verified for all i:

[2] As the mathematician Rene Thom pointed out, "prédire n'est pas expliquer", ie prediction don't explain anything

[3] This kind of transform doesn't lead to perfect reconstruction; add and overlapping succesive block does

$$
\begin{aligned}
W_i^{S_R} B_1 A_2 W_i^{A_L} = W_i^{S_L} B_2 A_1 W_i^{A_R} &= \mathbf{0} \\
W_{i+1}^{S_L} B_2 A_2 W_{i+1}^{A_L} + W_i^{S_R} B_1 A_1 W_i^{A_R} &= \mathbf{1}
\end{aligned}
$$

MDCT is a special solution for these three equations. MDCT solution kernel is build in such a way that:

$$
\begin{aligned}
B_1 A_2 &= B_2 A_1 = 0 && \text{(B.9)} \\
B_1 A_1 &= \mathbf{1} + \mathbf{J} && \text{(B.10)} \\
B_2 A_1 &= \mathbf{1} - \mathbf{J} && \text{(B.11)} \\
& && \text{(B.12)}
\end{aligned}
$$

Where J is an antidiagonal matrix with ones on it. With this particular solution, the windowing conditions become:

$$
\begin{aligned}
W_i^{S_L} W_i^{A_L} + W_{i-1}^{S_R} W_{i-1}^{A_R} &= \mathbf{1} && \text{(B.13)} \\
W_i^{S_L} \mathbf{J} W_i^{A_L} &= W_{i-1}^{S_R} \mathbf{J} W_{i-1}^{A_R} && \text{(B.14)}
\end{aligned}
$$

The first equation B.14 is exactly the same condition that for an overlapped DFT. The equation B.14 is new, and is linked to time-domain aliasing cancellation. To understand exactly what B.14 means, it is useful to see that $\mathbf{J}D\mathbf{J}$ is a time reversed version of D when D is diagonal, and that (J) is involutive (ie $\mathbf{JJ} = \mathbf{1}$). So, B.14 demands that the synthesis window is the time-reversed version of the analysis window. Written in a non-matrix notation:

$$
w_i^a[n] * w_i^s[n] + w_{i-1}^a[N/2+n] * w_{i-1}^s[N/2+n] = 1 \qquad \text{(B.15)}
$$

and

$$
\begin{aligned}
w_i^a[n] &= w_{i-1}^s[N-1-n] \text{ for } n = 0 \ldots N/2-1 && \text{(B.16)} \\
w_i^s[n] &= w_{i-1}^a[N-1-n] && \text{(B.17)}
\end{aligned}
$$

Once the window's conditions are achieved for perfect reconstruction, one still have to find the MDCT kernel to lead to B.10, B.11 and B.12. The forward and inverse transforms for a N samples block of the signal $x_i[n]$ (the block starts at $n = 0$) are:

for $k \in [0N/2 - 1]$

$$X_i[k] = \sum_{n=0}^{N-1} w_i^a[n] x_i[n] cos(\frac{2\pi}{N}(n+n_0)(k+1/2)) \quad \text{(B.18)}$$

for $n \in [0N-1]$

$$x_i = w_i^s[n] \frac{4}{N} \sum_{k=0}^{N/2-1} X_i[k] cos(\frac{2\pi}{N}(n+n_0)(k+1/2)) \quad \text{(B.19)}$$

with $n_0 = (N/2+1)/2$.

So, now, we have a new transform which maps N samples into N/2 samples, with an overlapp of 50%, and enables perfect reconstruction. There is no data increase (ie MDCT is a basis). As the window's conditions are basically the same than for block DFT transforms, we can use the classic windows. To respect all conditions, we just have to build the synthesis window as the time-reversed version of the analysis window (this condition leads to the famous time-domain alisaing cancellation).

Appendix C

Brief overview about Discrete Wavelet Transform

Wavelet theory is a big scientific theory which comes from more than twenty years, ans is interesting for several reasons, not only purely scientific.

Even if the origin of wavelet can be set at the beginning of the twentieth century, with the work of the mathematician Haar C.3, the story really begins in the eighties, with the work of Morlet and Grossmann, who tried to find some analysis tools to study discontinuities in sismic signals. The initial discover of Haar was that it is possible to construct an orthonormal basis of $\mathbf{L}^2(\mathbb{R})$ from a single simple piece-wise constant function ψ:

$$\psi(t) \quad = \quad \begin{cases} 1 & \text{if } 0 \leq t < 1/2 \\ -1 & \text{if } 1/2 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}$$

with its dilatation and translations:

$$\left\{ \psi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \psi\Big(\frac{t - 2^j n}{2^j} \Big) \right\}_{(j,n) \in \mathbb{Z}}$$

The mathematician Meyer tried then to prove there doesn't exist any regular function ψ which can generate such an orthonormal basis; he fails, as he managed to construct several bases whose ψ is C^∞ !

This was the beginning of big researchs in applied mathematics, filter-bank theory, computer vision and so on. That's why wavelets are so interesting to study from my point of view: it becomes more and more difficult to do scientific researches without focusing only on one veru specific field, and wavelets are across several very different scientific disciplines.

The present section is only aiming at giving the basic definition of wavelet bases: that's the goal of the first section. The second section gives the link with filter bank theory, which enables to compute wavelet coefficients.

C.1 Multi resolution approach

The multi resolution approach was first developed by Meyer and Mallat, who take the initial idea from computer vision: the principle is to decompose a whole space not woth frequency/time atoms, but with the more "vision appealing" notion of scale/time atoms. The scale is defined as the dilatation of a certain set of vectors; the very surprising result here is that one can construct bases for big spaces like $\mathbf{L}^2(\mathbb{R})$ from only one function. [1].

C.1.1 The scaling function

The idea of the multiresolution approach is that for $f \in \mathbf{L}^2(\mathbb{R})$, the projection over the set $\psi_{n_0,k\in\mathbb{Z}}$ represents the error approximation between two succesive subsets of $\mathbf{L}^2(\mathbb{R})$.

The formal definition of a multiresolution approximation can be found in [Mal98]. I prefer here present a non formal definition, which is sufficient enough to understand where orthonormal bases are coming from. Let's take one function ϕ, with a null first order moment (ie a mean to zero), that is square integrable; we note $\phi_k(t) = \phi(t-k)$ the dilatations of ϕ. We then define V_0 as the closure of the vector space spanned by the set $\{\phi_k\}_[k \in \mathbb{Z}$:

$$V_0 = \overline{Span(\phi_k)} \tag{C.1}$$
$$\tag{C.2}$$

Which means that

$$f \in V_0 \rightleftharpoons f = \sum_{k\in\mathbb{Z}} \alpha_k \phi_k \tag{C.3}$$

If the family set $\{\phi_k\}_{k\in\mathbb{Z}}$ is orthogonal, it is an orthogonal basis of V_0, and we generally normalize the vectors to have a orthonormal basis. That way, one can rewrite C.3 such that:

$$f \in V_0 \rightleftharpoons f = \sum_{k\in\mathbb{Z}} < f, \phi_k > \phi_k \tag{C.4}$$

which is a lot more easier to handle.

We then define ϕ_j, a dilated version of ϕ, and its translated version $\phi_{j,k}$ by:

[1] if we are thinking about the way Riemann integral, and later Lebesgue integral are defined, with limits of piece wise approximations of some areas, it is not so surprising

$$\phi_j(t) = \sqrt{2^{j/2}}\phi(2^j t) \quad \text{(C.5)}$$

$$\phi_{j,k}(t) = 2^{j/2}\phi_j(t - k/2^j) \quad \text{(C.6)}$$

V_j is then defined as the closure of the space spanned by $\{\phi_{j,k}\}_{k\in\mathbb{Z}}$. Normally, when j increases, the spaces V_j increases, two. The factor j can be seen as a scale factor: one can see a more "detailed" version of f when it is projected on V_j, where j is increasing. In fact, what we really want is:

- $V_j \subset V_{j+1}$
- $V_{-\infty} = 0$
- $V_\infty = \mathbf{L}^2(\mathbb{R})$

To assure the first relation, we just have to verify it for the $\phi_{k,j}$ and $\phi_{k,j+1}$ for one fixed j, and a simple recurence proves it for all $j \in \mathbb{Z}$.

$$V_j \subset V_{j+1} \rightleftharpoons \phi_j \in V_{j+1}$$

For $j = 0$, this relation becomes:

$$\phi(t) = \sum_n h_s(n)\sqrt{2}\phi(2t - n) \quad \text{(C.7)}$$

This can be seen as a finite difference equation, and the $h_s(n)$ are called scaling coefficients.

C.1.2 The wavelet function

For now, we have some V_j, and orthogonal projections on them give several levels of precision for any $f \in \mathbf{L}^2\mathbb{R}$. But it would be more practical to work directly with the error of approximation between different V_j. That's exactly the definition of wavelet spaces. We define W_0 such that [2]

$$V_1 = V_0 \oplus W_0 \text{ with } V_0 \perp W_0$$

And all W_j are defined the same way. $\psi_{j,k}$ are the dilated/scaled functions derived from one function, the wavelet function ψ, and they spanned the $W_{j,k}$ spaces. As $\psi \in V_1$, one can write:

[2] As V_j are *a priori* linear spaces of infinite dimension, the unicity of orthogonal complement is not true. I didn't find any informations about that, and no wavelet book speaks about this problem... So I concluded it is not !

$$\psi(t) = \sum_n h_w(n)\sqrt{2}\psi(2t-n) \tag{C.8}$$

An example of expansion on such a multiresolution system, for $f \in \mathbf{L}^2(\mathbb{R})$:

$$f = \sum_k c_{J_0} 2^{J_0/2}\phi(2^{J_0}t-k) + \sum_k \sum_{j=J_0}^{\infty} d_j(k)2^{j/2}\psi(2^j y-k)$$

where $V_{J_0} \oplus W_{J_0} \oplus W_{J_0+1} \oplus \ldots = \mathbf{L}^2(\mathbb{R}$. J_0 is called the coarsest level, and where $c_j =< f, \phi_{j,k} >$ and $d_j =< f, \psi_{j,k} >$

The big problem was to find such ϕ (called the scaling function, or the father wavelet) and ψ (often called the wavelet function, or the mother wavelet), whose dilatation/scaling define orthogonal sets and spann $\mathbb{L}^2\mathbb{R}$. A rather technical theorem from Mallat and Meyer shows the relation between multi resolution analysis, the coefficients h_s and h_w and the Fourier transform of these coefficients (h_w and h_s can be seen as filter coefficients).

C.2 Filterbank approach

To build a multi resolution analysis system, we need to have the following relations:

$$\phi(t) = \sum_n h_s(n)\sqrt{2}\phi(2t-n) \tag{C.9}$$

$$\psi(t) = \sum_n h_w(n)\sqrt{2}\psi(2t-n) \tag{C.10}$$

$$\int \phi(t)\phi(t-k)dt = \delta(k) \text{ orthonormality for the } V_0 \text{ basis} \tag{C.11}$$

$$\int \psi(t)\phi(t-k)dt = 0 \tag{C.12}$$

One can shows[Mal98], [CSB98] that these relations imply

$$h_w(k) = (-1)^n h_s(1-k)$$

Which are exactly the relations between the two impulse responses of the analsys filter of a 2 band filterbank ! This relation it the key link between multi resolution analysis and filter bank theory. It was used to build several wavelet bases.

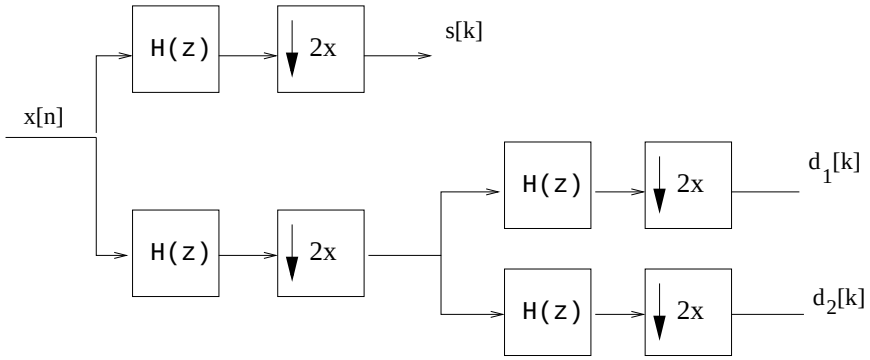

Figure C.1: A 2 level wavelet decomposition filterbank

C.2.1 Mallat algorithm

Here, I will just present the Mallat algorithm, which enables to compute the discrete wavelet transform for any $f \in \mathbf{L}^2(\mathbb{R}$. No formal proof is provided, just some elements to understand the algorithm. First, we derive from C.10 that

$$\phi(2^j t - k) = \sum_m h_s(m - 2k)\sqrt{2}\phi(2^{j+1}t - m)$$

For $f \in V_{j+1}$, we write the expansion on V_{j+1}, and as $V_j \oplus W_j = V_{j+1}$, we obtain a relation between two successive scales. If $c_{j,k}$ and $d_{j,k}$ are the coordinates of the projection on V_j and W_j, respectively:

$$c_{j,k} = \sum_m h_s(m - 2k)c_{j+1,k}(m) \quad \text{(C.13)}$$

$$d_{j,k} = \sum_m h_w(m - 2k)c_{j+1,k}(m) \quad \text{(C.14)}$$

We so obtain coefficient of a scale j from the scale j+1 thanks to filtering/decimating them with the filters h_s and h_w. To compute them, we need to define a level of precision (for discrete time signals, it is generally the sampling period), and we pass the signal through a dyadic bank filter (see figure C.1). As the number of level is normally limited to a few iterations, and don't depend upon the input signal's size, that means that the algorithm's complexity is O(N), where N is the size of the algorithm [3]

[3] it is worth noticing that concretely, it is not really faster that FFT, which can be in 3/2N*log(N), for usual sizes of signal.

Appendix D

Details about the tools used in my master thesis

All my work was done on a GNU/Linux system, Debian (thanks to Paul and Nicolas for helping me when I was almost depressed by the computer's mysteries). This master thesis was written in LaTeX, with the package AMS-FONTS (thanks again to Paul for giving me his LaTeXmakefile, and to Samer for his style files). The LaTeXfiles were edited with gvim the GTK version of vim, an efficient text editor with syntax highlighting.

All matlab scripts were done on the release 13 of Matlab for Linux, with the Wavelab package, the signal processing toolbox and the statistical toolbox for a few scripts. The figures were done on xfig.

The C++ code was written with gvim, and compiled with Gcc/g++, the excellent GNU preprocessor/compiler/linker suite.

The presentations were written with prosper, a great LaTeXpackage for presentations.

Bibliography

[CD03] Laurent Daudet Samer Abdallah Chris Duxburry, Juan Pablo. A tutorial on transient detection in audio signals, 2003. Not published, was submitted to the Elsevier Acoustics and Speech Signal Processing journal.

[CSB98] Haitao Guo C. Sidney Burrus, Ramesh A. Gopinah. *Introduction to wavelets and wavelet transforms, a primer*. Prentice Hall, 1998.

[Dau00] Laurent Daudet. *Représentation structurelles de signaux audiophoniques - Méthodes hybrides pour des applications à la compression*. PhD thesis, Université de Provence (Aix-Marseille I) U.F.R Sciences de la matière, 2000.

[LD02a] B. Torrésani L. Daudet. Hybrid representations for audiophonic signal enconding. *Signal Processing*, 82, 2002.

[LD02b] D. Darlington L. Daudet. Digital audio effects in wavelet domain. DAFX02, 2002.

[LD02c] Mark Sandler Laurent Daudet. *transactions on speech and audio processing*, 2002.

[Mal98] Stéphane Mallat. *A wavelet tour of signal processing*. Second edition. Academic Press, 1998.

[Mal99] H.S Malvar. A modulated complex lapped transform and its applications to audio processing. In *ICASSP*, 1999.

[MAR86] Quatieri Thf Mc Auley RJ. Speech analysis/synthesis based on a sinusoidal representation. *IEEE trans. on Acoust., Speech and Signal Proc*, 1986.

[MB03] Richard E. Goldberg Marina Bosi. *Introduction To Digital Audio Coding And Standards*. Kluwer Academic publishers, 2003.

[Mit01] Sanjit K. Mitra. *Digital Signal Processing, a Computer Based Approach*. McGraw-Hill International Edition, 2001. 2d Edition.

[RCG94] J berger R. Coifman and M. Goldberg. Removing noise from music using local trigonometric bases and wavelet packets. *J. Audio Eng. Soc*, 42(10), 1994.

[Rod97] X. Rodet. Musical sound signal analysis/synthesis : Sinusoidal+residual ans elementary waveform models, 1997.

[Ser97] Xavier Serra. *Musical Signal Processing*, chapter Musical Sound Modeling with Sinusoids plus Noise, pages 91–123. Swets and Zeitlinger Publishers, 1997.

[SL] Julius O Smith III S.N Levine. A sines + noise + transients audio representation for data compression and time/pitch scaling modification.

[XR93] G García X. Rodet, Ph Depalle. Tracking of partials for additive sound synthesis using hidden markov models. In *Procedings of ICMC*, pages 94–97. ICMC, 1993.