Mixing Perceptual Coded Audio Streams

Diploma Thesis

submitted by
Stefan Bayer

Institute for Electronic Music and Acoustics(IEM)
Graz University of Music and Dramatic Arts
A-8010 Graz, Austria

Advisor: Ao.Univ.-Prof DI Winfried Ritsch
Reviewer: O.Univ.-Prof Mag. DI Dr. Robert Höldrich

Graz, February 2005

Abstract

Perceptual audio coders based on the Modulated Discrete Cosine Transform (MDCT) and utilizing psychoacoustically based noise shaping for irrelevancy removal are widely used today.

After giving an overview of algorithms of perceptual audio coders and current coding standards, this thesis explores the possibilities to mix two audio streams based on the MDCT without completely decoding the streams back into the time domain. For this methods for adjusting block lengths of streams and mixing them together within the MDCT domain are developed. The devised scheme is investigated in terms of latency, computational complexity and effects on psychoacoustic processing.

As an example application a simple mixer for combining two Ogg Vorbis files with fixed window lengths is developed in MATLAB.

Zusammenfassung

Wahrnehmungsangepasste Audiokompressionsverfahren basierend auf der Modulierten Diskreten Cosinus Transformation (MDCT) sind seit geraumer Zeit etabliert und in weiter Verwendung.

Nachdem ein Überblick über Algorithmen wahrnehmungsangepasster Audiokodierungsverfahren und aktueller Kodierungsstandards gegeben wird, untersucht diese Diplomarbeit die Möglichkeiten, zwei Audioströme, die auf der MDCT basieren, zu mischen, ohne sie komplett in die Zeitdomäne zu dekodieren. Dafür werden Algorithmen für die Änderung von Blocklängen der Ströme und deren anschließende Mischung in der MDCT Domäne entwickelt. Diese werden in Hinsicht auf Latenzen, nötigen Rechenaufwand und Auswirkungen auf die psychoakustische Verarbeitung untersucht.

Als Beispielandwendung wird ein simpler Mischer für die Zusammenführung zweier mit Ogg Vorbis komprimierter Audiodateien mit fixen Blocklängen in MATLAB® implementiert.

Acknowledgements

This thesis is dedicated to my parents, who always supported me in doing the things my way, and for their nearly infinite patience when it comes to waiting for their son to finish his studies.

I'd like to thank Prof. Robert Höldrich who encouraged me to finish this work and my advisor, Prof. Winfried Ritsch for his support.

I further like to say a big thank to my sister Barbara, who has always been my best friend and advisor and all of my friends, especially Werne and Dani, who nudged me forward endlessly.

Contents

List of Tables

List of Figures

1. Introduction

Perceptual audio coders gained much interest since their gestation in the early 90's. They are widely used in audio applications that use the Internet, where bandwidth limitation is still an issue. A lot of the audio codecs in use, especially the ISO MPEG audio coding standards (the ubiquitous "MP3", in reality MPEG 1 Layer III, being one of them) are based on the Modulated Discrete Cosine Transform (MDCT).

1.1. Latency

For most applications where perceptual coded audio is used today, latency is not much of an issue, so its no problem to buffer much data and completely decode it into the time domain before it is further processed and/or played.

But there are several possible applications where latency is an issue, be it two way communication, musicians in different places performing together, or digital wireless applications like wireless microphone systems where the signal is encoded inside the microphone and the coded signal transmitted wireless to an receiver/mixer.

For this applications latency has to be very small, so one has to try to minimize this latency and possible processing time by decoding the signal not completely into the time domain, do the processing and reencode but decode only as far as needed to do the processing.

This thesis tries to achieve this by deriving algorithms to process the audio data within the MDCT domain with the focus on mixing audio streams based on MDCT coding.

1.2. Licensing and Patents

Although it is clear that coding standards developed an patented by single companies (like Dolby Digital or Sony ATRAC) can only be used after paying license fees, also the ISO MPEG audio coding standards can only be used after paying license fees since most algorithms incorporated into this standards were patented and contributed by commercial companies which have naturally an interest in generating revenues out of their developments. The only royalty free usage is allowed for the non-commercial distribution of coded audio material.

On the other hand the Open Source and Free Software concepts have also inspired the development of audio coding algorithms free of license fees, the most prominent is possibly Ogg Vorbis[1], which was chosen for an example implementation of the developed algorithms in this thesis. The decision was made because all encoding and decoding algorithms are available as source code and the BSD-style license allows to use and

modify this algorithms not only for other open source projects (as would be forced by e.g. the GPL license) but also in proprietary, closed source and commercial applications without any license fees.

1.3. Structure of this Thesis

Chapter 2 gives an overview on algorithms of perceptual audio coders, excisting audio coders and quality assessment of audio signals.

Chapter 3 derives the algorithms for mixing MDCT based audio streams, and investigates them in terms of latency, computational complexity and influence on the psychoacoustic model for reencoding.

Chapter 4 gives an overview of the implemented example application "Vorbis file mixer for MATLAB" and its evaluation.

Chapter 5 provides a conclusion of the work done.

2. Algorithms of Perceptual Audio Coders[2]

2.1. Introduction

Audio coding or *audio compression* algorithms are used to obtain compact digital representations of high-fidelity(wideband) audio signals for the purpose of efficient transmission or storage. The central objective in audio coding is to represent the signal with a minimum number of bits while achieving transparent signal reproduction, i.e., generating output audio that cannot be distinguished from the original input, even by a sensitive listener ("golden ears")[2].

The data rates of first generation (CD,DAT) digital audio, though providing high-fidelity, dynamic range and robustness, exceeded and still mostly exceed the available band-widths of network and wireless multimedia digital audio systems. Due to this constraints, during the last two decades considerable research was carried toward formulation of compression schemes that can satisfy simultaneously the conflicting demands of high compression ratios and transparent reproduction quality for high fidelity audio signals, leading to several standards.

2.2. Generic Perceptual Audio Coding Architecture

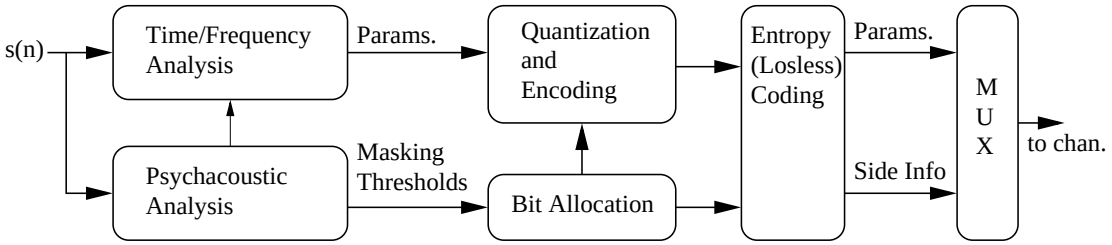


Figure 2.1.: Generic perceptual audio encoder

This chapter considers several classes of analysis-synthesis data compression algorithms, including those that manipulate transform components, time-domain sequences from critically sampled banks of bandpass filters, sinusoidal signal components, linear predictive coding (LPC) model parameters, or some hybrid parameters. Within each algorithm class, either *lossless* or *lossy* compression is possible.

Before considering different classes of audio coding algorithms, we note the architectural similarities that characterize most perceptual audio coders. The lossy compression

systems described in this chapter achieve coding gain by exploiting both *perceptual irrelevancies* and *statistical redundancies*. Most of these algorithms are based on the generic architecture shown in Fig. 2.1. The coders typically segment input signals into quasistationary frames ranging from 2 to 50ms in duration. Then, a time-frequency analysis section estimates the temporal and spectral components on each frame. Often the time-frequency mapping is matched to the analysis properties of the human auditory system, although this is not always the case. The objective is to extract from the input audio a set of time-frequency parameters that is amenable to quantization and encoding in accordance with a perceptual distortion metric. Depending on overall system objectivities and design philosophy, the time-frequency analysis section might contain a:

- unitary transform;
- time-invariant bank of critically sampled, uniform, or nonuniform bandpass filters;
- time-varying (signal-adaptive) bank of critically sampled, uniform, or nonuniform bandpass filters;
- harmonic/sinusoidal analyzer;
- source-system analysis (LPC/multipulse excitation);
- hybrid transform/filter bank/sinusoidal/LPC signal analyzer.

The choice of the time-frequency analysis section always includes a fundamental tradeoff between time and frequency resolution.

The psychoacoustic model (see 2.3) delivers masking thresholds so that the time-frequency parameters can be quantized without audible artifacts.

The quantized parameters can be further compressed by removing further redundancies trough lossless coding (e.g. Huffmann or arithmetic coding).

2.3. Psychoacoustic principles[3]

Most modern audio coders use psychacoustic effects to remove *irrelevant* signal information, that can't even be detected by a trained listener, to achieve greater compression ratios. This information is identified through applying psychoacoustic principles, including absolute threshold of hearing, temporal and simultaneous masking, critical band analysis and the spread of masking to the signal.

Before going into detail it is necessary to define the *sound pressure level* (SPL). The SPL gives the level (intensity) of sound pressure of an acoustical stimulus in decibel (dB) relative to an internationally defined reference level,

$$L_{SPL} = 20log_{10}(p/p_0) \qquad (2.1)$$
$$p_0 = 20 \cdot 10^{-5} Pa$$

2.3.1. Absolute Threshold of Hearing

The absolute threshold of hearing in quiet indicates the minimum sound pressure level a pure tone must have to be heard by an average listener in an otherwise quiet sourrounding. It is easily obtained in listening tests and averaged over a large number of test persons. It is dependent on the frequency, as can be seen in figure 2.2. The threshold is well approximated by the nonlinear function

$$T_q(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4 dB, \qquad (2.2)$$

where f is expressed in Hz.

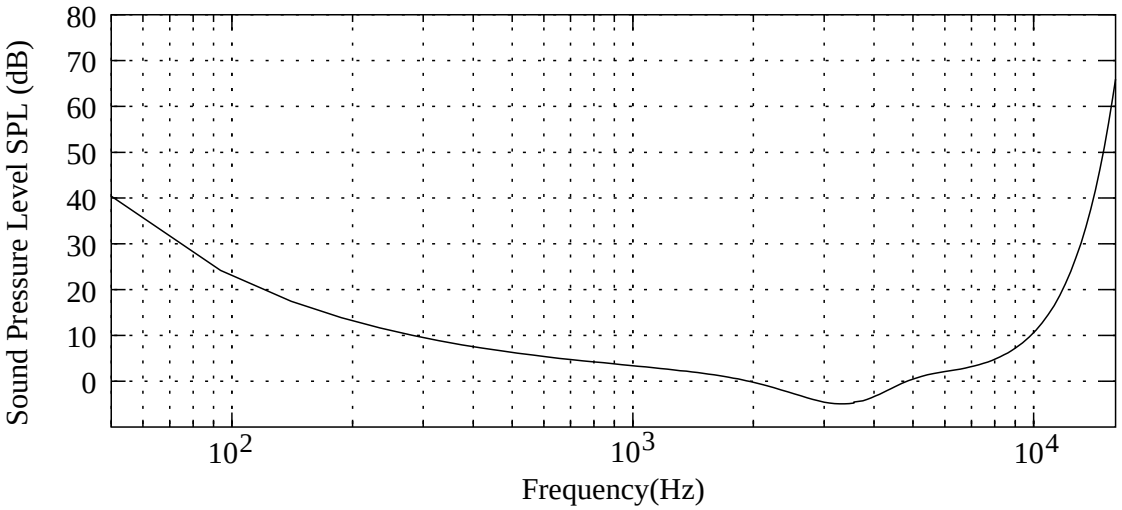


Figure 2.2.: Absolute Threshold of Hearing

The problem of applying the absolute threshold of hearing to audio coding is that the actual playback level is not known beforehand, so in coding systems the lowest point (near 4kHz) is referenced to the energy in ±1 bit of signal amplitude.

2.3.2. Critical Bands

Normally, audio signals are spectrally complex and time variant, so the actual detection threshold is a time-varying function of the input signal. In order to estimate this threshold, we have to understand how the auditory system performs spectral analysis. A frequency-to-place transformation occurs along the basilar membrane (fig. 2.3) in the inner ear (cochlea).

Each point at the basilar membrane is associated with a certain frequency, its characteristic frequency, so that a travelling wave coming from the oval window with that frequency reaches its maximum amplitude at this point of the membrane. As a result of this frequency-to-place transformation the cochlea can be seen as a bank of highly overlapping bandpass filters. The bandwidth of one of this filters is called the *critical bandwidth* and is a nonlinear function of the frequency. It can be assumed that for the

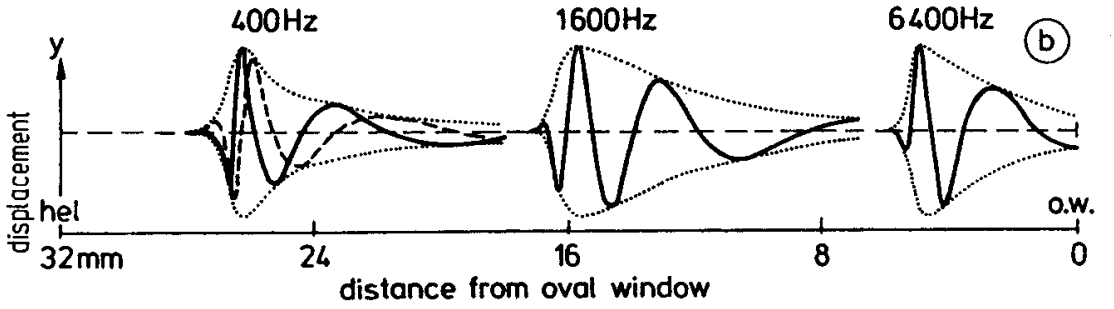


Figure 2.3.: The frequency-to-place transformation along the basilar membrane (from[3])

perception of an audio event mainly stimuli within a critical band are taken into consideration. A typical example and also a method to measure the critical bandwidth is the threshold of hearing of a narrowband noise between two tones of equal level (fig. 2.4).

As long as the two masking tones are within the critical band, the threshold is independent of the frequency separation of the two tones and starts decreasing beyond critical bandwith.

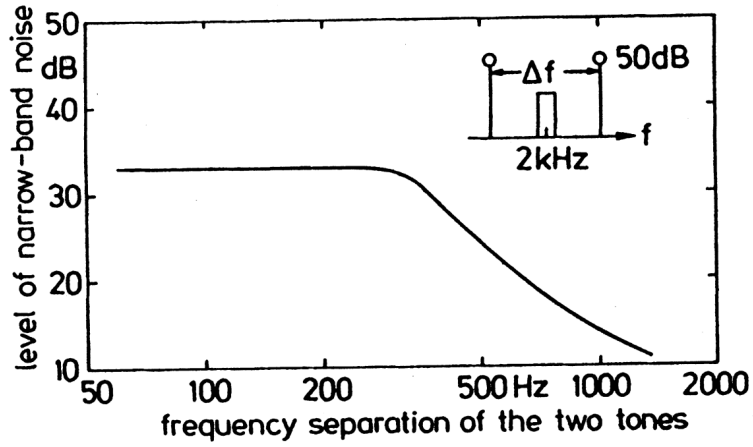


Figure 2.4.: The threshold of a narrow-band noise centred between two masking tone of equal level as a function of the frequency separation of the two tones (from[3])

The critical bandwidth tends to remain constant for frequencies under 500Hz and increases to approximately 20% the center frequency above 500Hz, a useful analytical expression is

$$\Delta f_G/Hz = 25 + 75(1 + 1.4(f/kHz)^2)^{0.69} \tag{2.3}$$

It is also useful to divide the audible frequency range up to 16kHz into 24 critical bands (table 2.2), for this filter bank the so called critical-band rate scale is defined so that the

distance of one critical band is one "Bark" and can be expressed as

$$z/Bark = 13arctan(0.76f/kHz) + 3.5arctan(f/7.5kHz)^2. \quad (2.4)$$

Critical Band Number	Lower Freq. Hz	Center Freq. Hz	Upper Freq. Hz	Bandwith Hz
1	0	50	100	100
2	100	150	200	100
3	200	250	300	100
4	300	350	400	100
5	400	450	510	110
6	510	570	630	120
7	630	700	770	140
8	770	840	920	150
9	920	1000	1080	160
10	1080	1170	1270	190
11	1270	1370	1480	210
12	1480	1600	1720	240
13	1720	1850	2000	280
14	2000	2150	2320	320
15	2320	2500	2700	380
16	2700	2900	3150	450
17	3150	3400	3700	550
18	3700	4000	4400	700
19	4400	4800	5300	900
20	5300	5800	6400	1100
21	6400	7000	7700	1300
22	7700	8500	9500	1800
23	9500	10500	12000	2500
24	12000	13500	15500	3500
25	15500	19500		

Table 2.2.: List of critical bands (from Zwicker[3]).

Another possible measure of the perceptual frequency of the ear is the *equivalent rectangular bandwidth* (ERB), which comes from research on auditory filter shapes. The ERB of a filter corresponds to the the bandwith of a rectangular filter which has the same peak transmission and passes the same power given a white noise input as the corresponding auditory filter. The ERB scale can be expressed as

$$ERB(f) = 24.7(4370f + 1) \quad (2.5)$$

When the Bark and the ERB scale are compared (fig. 2.5) it can be seen that the ERB bandwith decreases below 500Hz, while the critical bandwith remains flat, which has implications on the optimal filter bank design and perceptual bit allocation strategies.

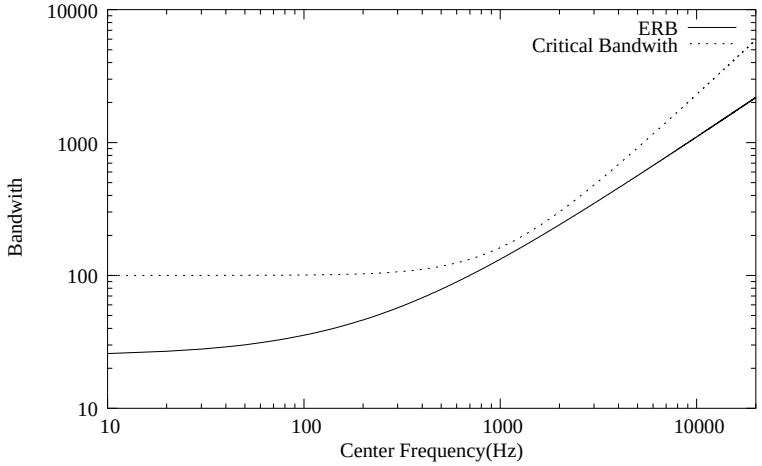


Figure 2.5.: ERB versus critical bandwith as a function of the center frequency

The frequency resolution of the auditory filter bank largely determines which portions of a audio signal are perceptually irrelevant. The time-frequency analysis in the auditory system results in simultaneous and nonsimultaneous masking effects that are used by modern audio coders to shape the quantization noise spectrum.

2.3.3. Masking

Masking is a process where one sound cannot be heard because of the presence of another sound, that means the threshold of hearing is increased in comparison to the threshold of hearing in quiet. The stronger signal normally is called the *masker* whereas the inaudible signal is called *maskee*. Masking effects can either be categorized as simultaneous, when both masker and maskee are presented at the same time or nonsimultaneous, when a time offset occurs between the two stimuli.

2.3.3.1. Simultaneous Masking

Simultaneous masking is the more important phenomenon, since it produces the largest amount of masking. In a frequency domain view, the spectral shape of the stimuli determine the masking threshold, in a time-domain view phase relationships are also significant.

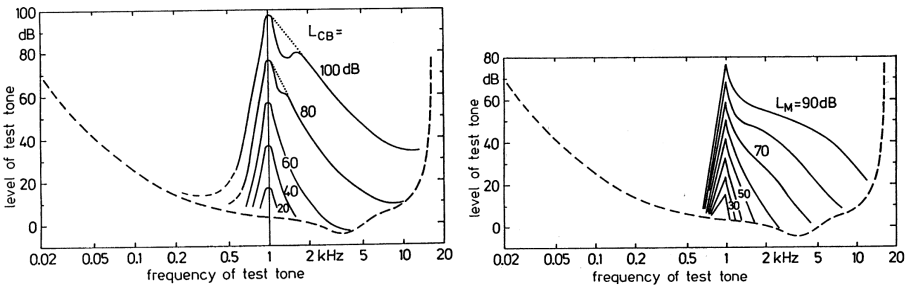
An explanation for the masking effect is that a strong signal creates such an excitation on the basilar membrane in a specific critical band that weaker signals are suppressed.

Normally spectrally complex masking patterns occur in a real audio signal, but for the shaping of the coding distortions it is convenient to look at three basic masking types, *noise masking tone*, *tone masking noise*, and *noise masking noise*.

Noise masking Tone In the NMT scenario a narrow-band noise (i.e. with critical bandwidth), masks a tone within the same critical band. The hearing threshold for the

tone is related to the intensity and, to a lesser extent, to the center frequency of the masking noise. The minimum *signal-to-mask-ratio* SMR, which is the level difference between the masking noise and the threshold, occurs when the probe tone frequency equals the center frequency of the masking tone. For the example in fig 2.7 (a), the SMR is 4dB. Masking power decreases, and SMR increases, when the probe tone frequency is above or below the center frequency.

Tone masking Noise In the case of TMN a pure tone occurring at the center of a critical band masks noise of any subcritical bandwith or shape, as long as the noise spectrum is below a threshold determined by the level and the frequency of the masking tone. The minimum SMR lies between 21 and 28 dB (see fig. 2.7 (b)). As with the NMT, the TNM masking power decreases for critical bandwith noises centered below or above the frequency of the masking tone.



(a) Noise masking Tone (b) Tone masking Noise

Figure 2.6.: Masking patterns of noise masking tone and tone masking tone (from[3])

Asymmetry of Masking Fig. 2.7 clearly shows that the SMR of NMT and TMN differ greatly. Tonal maskers yield a greater SMR than noise maskers, that means noise masking is more effective than tonal masking.

Masking Patterns Fig. 2.6 shows examples of masking patterns for tone masking tone and noise masking tone for different masker levels at a specific frequency. For tone masking tone the slope towards lower frequencies becomes less steep for lower masker intensities, while the slope towards higher frequencies becomes less steep for *higher* masker intensities.

With a small band noise as masker, the slope towards lower frequencies remains nearly independent of the masker level, while the slope towards higher frequencies shows a similar behavior as with tonal maskers.

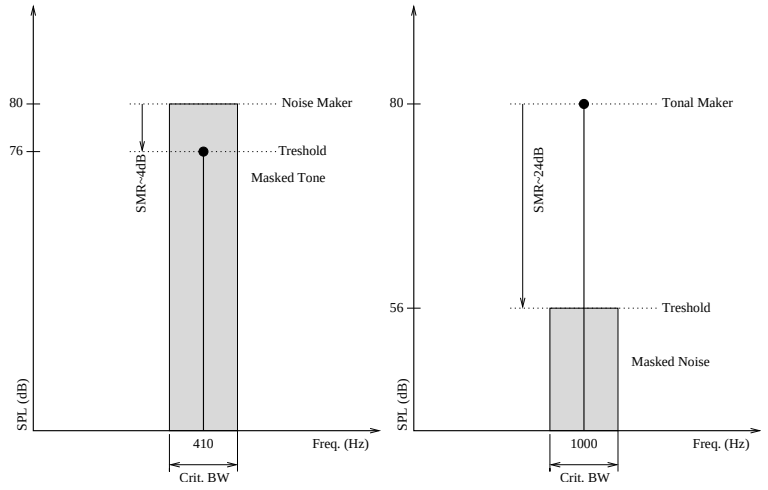


Figure 2.7.: Example to illustrate the asymmetry of masking

2.3.3.2. Temporal Masking

Masking not only happens when masker and maskee are presented simultaneously, but also extends in time. As it can be seen in fig. 2.8, sounds are masked after the masker is removed, and, which is at first surprising, even before the onset of the masker. The first case, called postmasking, corresponds to a decay in the effect of the masker and is more or less expected. The second case, premasking, does not mean that the ear can hear into the future, but can be explained by different build up times for sensations in the auditory system, where louder stimuli have a faster build up time than fainter sounds.

The duration for premasking is about 20ms, for postmasking it is dependent on the level of the masker and can excess 100ms.

Tonal masking is not incorporated into psychacoustic models of actual coding standards, but only in experimental coders and in the psychoacoustic model of the PEAQ standard for objective assessment of audio quality (see section 2.8.2).

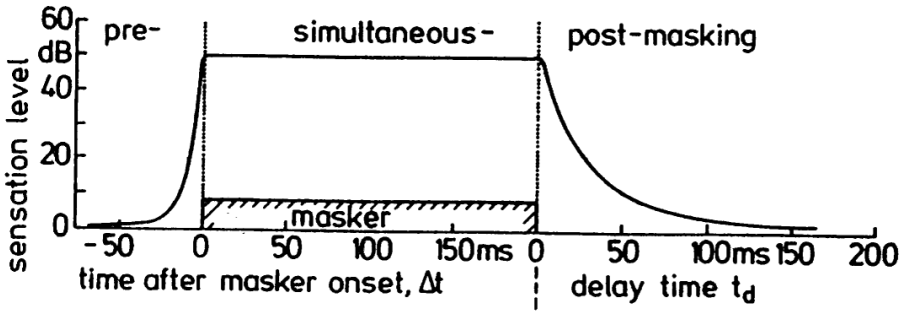


Figure 2.8.: Temporal Masking (from[3])

2.3.4. Example of an Psychoacoustic Model (MPEG-1 Psychoacoustical model 1)

The calculation of the signal-to-mask-ratio needed for the bit allocation is computed based on the following steps [4]:

1. Calculation of the FFT for time to frequency conversion
2. Determination of the sound pressure level in each subband
3. Determination of the threshold in quiet (absolute threshold)
4. Finding of the tonal (more sinusoid-like) and non-tonal (more noise-like) components of the audio signal
5. Decimation of the maskers, to obtain only the relevant maskers
6. Calculation of the individual masking thresholds
7. Determination of the global masking threshold
8. Determination of the minimal masking threshold in each subband
9. Calculation of the signal-to-mask-ratio in each subband

2.4. Time-Frequency Transformation

The basic idea in time-frequency transformation is to reduce the amount of data needed to reproduce the audio signal. Consider the simple example of a sine wave. Its representation in the frequency domain is fully described by three parameters, frequency, phase and amplitude while for the same sine wave a huge amount of data in the time domain is needed.

While real audio signals are not that simple, they can be considered as quasi-stationary and they can be modelled by using short-time spectrum analysis.

Once the signal is represented in the time-frequency domain, the number of bits used to encode each frequency component can be adjusted by removing redundant information.

Using the psychoacoustic principles from the preceding section, the data in the frequency domain can be further compressed by removing irrelevant information (i.e. information lost in the auditory system).

Although for early coders the distinction between transform coders using unitary transforms like DFT and DCT and subband coders using e.g. tree structured QMF filter banks was appropriate, modern coders using either PQMF filer banks with a relatively low number of bands (e.g. 32) or MDCT filter banks with high resolution (up to 2048 bands). Confusion comes from the fact that the MDCT is normally implemented with algorithms that use fast transforms, but are in fact mathematically equal to subband coders, as was developed by Malvar[5].

2.4.1. Subband Filter Banks

Fig. 2.9 shows the structure of a filterbank containing L subbands. In each subband, the input signal is first filtered with the analysis filter F and then decimated by the factor M, that means only every M-th sample is taken from the filter output. If the decimation factor M equals the number of subbands, the filter bank is called critically sampled or maximally decimated, meaning that the number of subband samples equals the number of input samples.

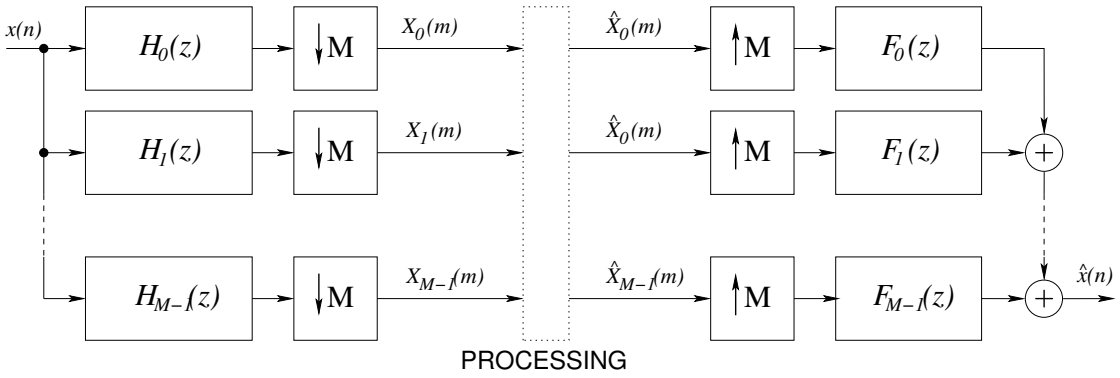


Figure 2.9.: Critically sampled M-channel filter bank

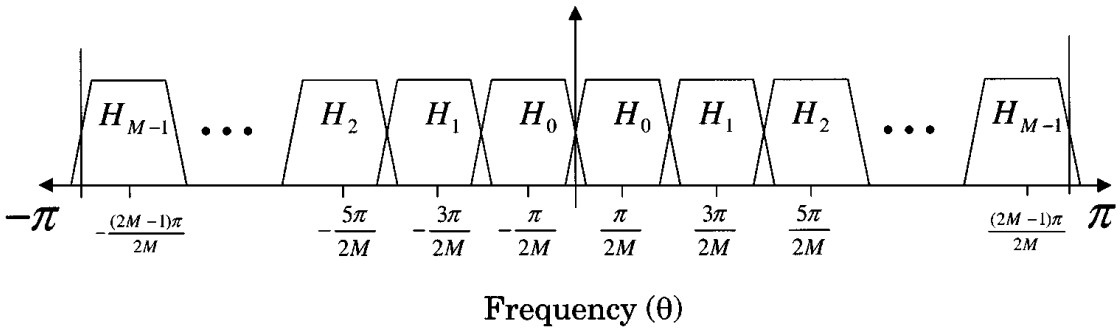


Figure 2.10.: Frequency response of the oddly stacked M-channel filter bank (from [2])

One goal in designing the filter bank is to obtain good signal reconstruction, so that if the subband samples are not modified, i.e. $\widehat{X}_k(m) = X_k(m)$, then the output signal $\widehat{x}(n)$ should approximate $x(n - D)$, where D is the processing delay. To obtain the necessary condition on the filter responses for signal reconstruction, lets start by noting that the output of the k-th analysis filter is given by [5]

$$X_k(m) = \sum_{n=-\infty}^{\infty} x(n) h_k(mM - n) \tag{2.6}$$

where $h_k(n)$ is the impulse response of the kth analysis filter. The reconstructed signal

can be written as a function of the processed subband signals as

$$\hat{x}(n) = \sum_{k=0}^{M-1} \sum_{m=-\infty}^{\infty} \hat{X}_k(m) f_k(n - mM) \tag{2.7}$$

where $f_k(n)$ is the impulse response of the kth synthesis filter. When we do not modify the subband signals, we have $\hat{X}_k(m) = X_k(m), \forall k$. Then 2.6 can be substituted into 2.7, with the result

$$\hat{x}(n) = \sum_{l=-\infty}^{\infty} x(l) h_T(n, l) \tag{2.8}$$

where the time-varying impulse response of the total system is given by

$$h_T(n, l) = \sum_{k=0}^{M-1} \sum_{m=-\infty}^{\infty} f_k(n - mM) h_k(mM - l) \tag{2.9}$$

We can obtain perfect reconstruction (PR), if and only if $h_T(n, l) = \delta(n - l - D)$, that is

$$\sum_{k=0}^{M-1} \sum_{m=-\infty}^{\infty} f_k(n - mM) h_k(mM - l) = \delta(n - l - D) \tag{2.10}$$

where D is just a delay, which must be included if we want the analysis and synthesis filters to be causal. One approach to design such filters is described in the following.

For $M = 2$, the z-Transform of the output of the synthesis filters is [6]

$$\hat{X}(z) = \frac{1}{2} \left[H_0(z) F_0(z) + H_1(z) F_1(z) \right] X(z) + \frac{1}{2} \left[H_0(-z) F_0(z) + H_1(-z) F_1(z) \right] X(-z) \tag{2.11}$$

The aliasing component $X(-z)$ can be cancelled with the following choice for the synthesis filter

$$H_1(z) = -H_0(-z), F_0(z) = -H_1(-z), F_1(z) = H_0(-z) \tag{2.12}$$

Filters satisfying 2.12 are called *quadrature mirror filters*.

With the restriction in 2.12, and the output signal being a delayed copy of the input signal, we get

$$H_0^2(z) - H_0^2(-z) = 2z^{-D} \tag{2.13}$$

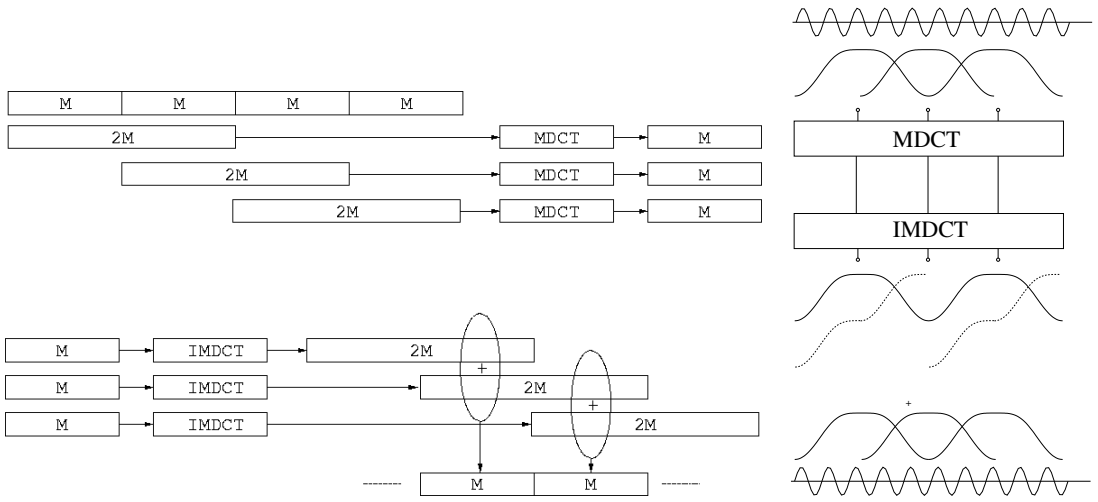
However, no practicable filters satisfy the perfect reconstruction constraints, but it is possible to derive filters that reasonably well approximate the QMF PR requirement.

In general, in audio coding we are interested in filter banks with a number of subbands $M >> 2$. For this case a class of filters called *pseudo-quadrature mirror filters* (PQMF) exists. A structure used in the MPEG-1 audio coding scheme [4] uses subband filters which are a modulated version of a single low-pass filter with bandwidth fs/N

$$h_k(n) = h(n) cos[\frac{\pi}{M} \left[(k + \frac{1}{2})(n - \frac{L-1}{2}) \frac{\pi}{M} + \phi_k \right] \tag{2.14}$$

where N is the number of frequency channels and L is the length of the filters h_k. The reconstruction filters can be derived from the analysis filter as follows

$$h_k(n) = g_k(L - 1 - n). \tag{2.15}$$



(a) (b)

Figure 2.11.: MDCT

2.4.2. MDCT

The MDCT is based on time domain aliasing cancellation (TDAC) and was first developed by Princen and Bradley[7] independently from the development of PQFM filter banks, but Malvar[5] later unified both approaches in the frame of the lapped orthogonal transform LOT.

For the MDCT (or modulated lapped transform MLT), the length L of the filters is constrained to be equal to twice the number of subbands, i.e. $L = 2M$.The filter responses can be put in the modulated form of 2.14. For the MDCT, every M samples a part of the signal with length $2M$ is taken and transformed, giving M subband samples, meaning that the MDCT is critically sampled. For the inverse MDCT the $2M$ time samples are added in an overlap-add process (see Fig. 2.11(a)) to reconstruct the signal. The analysis window (impulse response of the prototype filter) and the synthesis window are identical, so the constraints for perfect reconstruction are [8]

$$h(k) = h(2N - 1 - k) \tag{2.16}$$

$$h^2(k) + h^2(k + N) = 1 \tag{2.17}$$

A window fullfilling this constraints is the sine window

$$h(n) = \sin\left[(n + \frac{1}{2})\frac{\pi}{2M}\right] \tag{2.18}$$

which is widely used in audio coding. The forward MDCT is defined as

$$X(k) = \sum_{n=0}^{M-1} x(n)p_{n,k} \tag{2.19}$$

for $k = 0, 1, \cdots, M-1$ where

$$p_{n,k} = h(n)\sqrt{\frac{2}{M}}\cos\left[\frac{(2n+M+1)(2k+1)\pi}{4M}\right]. \tag{2.20}$$

To recover $x(n)$, one requires not only $X(k)$ for the current block, but also the previous block $X^P(k)$ (see. Fig. 2.11). Then

$$x(n) = \sum_{k=0}^{M-1}\left[X(k)p_{n,k} + X^P(k)p_{n+M,k}\right]. \tag{2.21}$$

For M a power of 2, fast implementations for computing the direct and inverse MDCT utilizing fast block transforms exist.

A summary of the main properties of the MDCT is [8]

- MDCT is not an orthogonal transform, perfect reconstruction can only be achieved in the overlap-add process
- If the frequency component of a signal and the basis function $p_{n,k}$ of the MDCT with the same frequency are 90 out of phase, the resulting MDCT transfer component is zero. That means that the MDCT does not fulfill Parseval's theorem, i.e. the time domain energy is not equal to the frequency domain energy.
- Nevertheless, on average, MDCT, similar to such orthogonal transforms as DFT, DCT, DST, etc. possesses energy compaction capability and acceptable Fourier spectrum analysis.

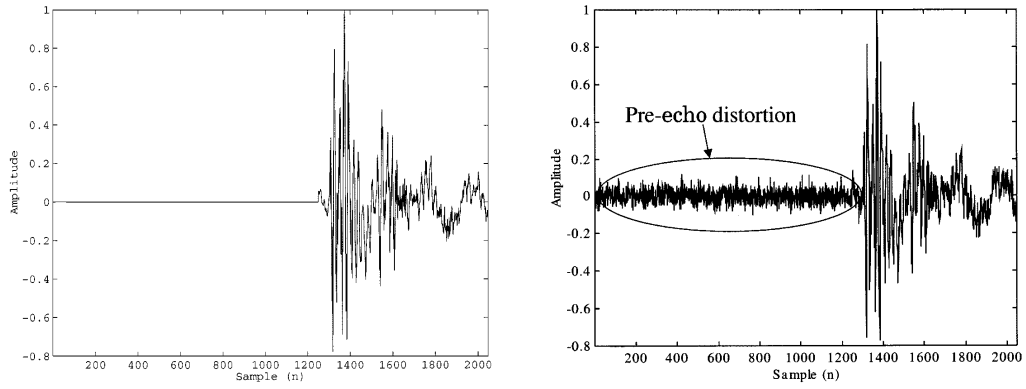
Perfect reconstruction is lost when the subband signals are altered, so the prototype filter (window) should be designed such that there is low frequency aliasing between the subbands, i.e. the subband filters have good stopband attenuation.

2.4.2.1. Pre-Echo Distortion

When coders use perceptual coding rules, an artifact known as pre-echo distortion can arise. Pre-echoes occur when a signal with a sharp attack begins near the end of a transform block following a region of low energy. This situation can arise when coding recordings of percussive instrument, for example the castanets (see Fig. 2.12). For a block-based algorithm, when quantization is performed to satisfy the masking thresholds from the psychoacoustic model, time-frequency uncertainty causes the inverse transform to spread the quantization noise evenly troughout the reconstructed block (see Fig. 2.12(b)).

This results in unmasked distortion throughout the low-energy region preceding the signal attack at the decoder. Different strategies exist to reduce this pre-echo distortions.

Bit Reservoir Although most coders have a fixed bit rate, the instantenous bit rate required to satisfy masked thresholds on each frame are different. So bits not needed for frames with low demand are added to a bit reservoir which can be used for frames with higher demand.



(a) (b)

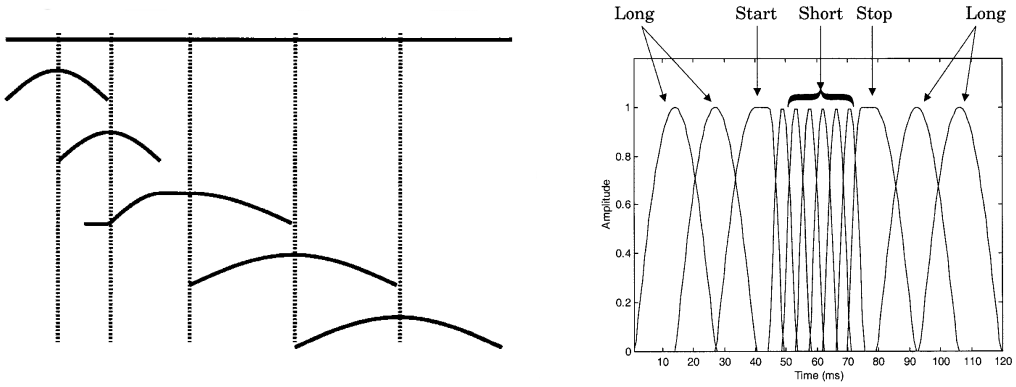
Figure 2.12.: Pre-echo example: (a) uncoded castanets and (b) transform coded castanets, 2048 block size

Window Switching A sufficiently short window length reduces the pre-echo distortion, so coders use a long window for stationary segments while switching to a shorter window for transients. This minimizes the spread of quantization noise in time, so that temporal pre-masking effects may make it inaudible. To use window switching in MDCT-based coders, transition windows (see Fig. 2.13 (a)) have to be introduced. Shlien [9] has shown that the constraints for PR windows can be relaxed to allow for transient windows with PR reconstruction on the trade-off of poor time and frequency localization properties.

Hybrid, Switched Filter Banks In contrast to window switching schemes, the hybrid and switched filter banks build upon distinct filter bank modes. In hybrid filter banks, compatible filter banks are cascaded to achieve the time-frequency tiling best suited to the current input signal. In switched filter banks, hard switching decisions are made to select a single monolithic filter bank.

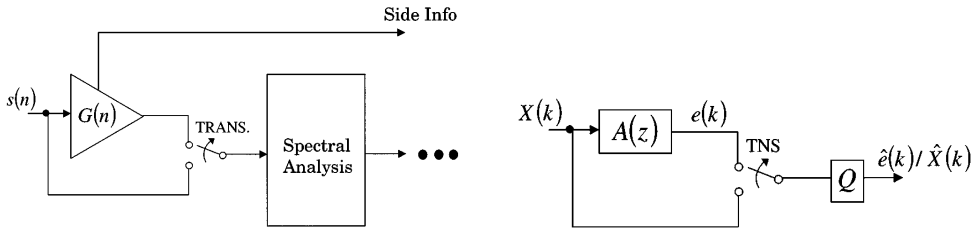
Gain Modification The gain modification smoothes transient peaks in the time-domain prior to spectral analysis. The time-varying gain and the modification time interval are transmitted as side information, and inverse operation are performed at the decoder to recover the original signal. This method has also caveats, because gain modification distorts the spectral analysis window which can lead to broadening of the filter banks responses at low frequencies beyond critical bandwidth.

Temporal Noise shaping (TNS) Temporal Noise shaping [10] is a frequency-domain method that operates on the spectral coefficients $X(k)$ generated by the analysis filter bank. The idea is to apply linear prediction LP (see section 2.4.4) across frequency (rather than time), since for an impulsive time signal, frequency-domain coding is max-



(a) (b)

Figure 2.13.: Window switching: (a) Introduction of a transient window (center) to switch between two window lengths (from [9])(b) example window switching scheme (MPEG-1, Layer III) (from [9])

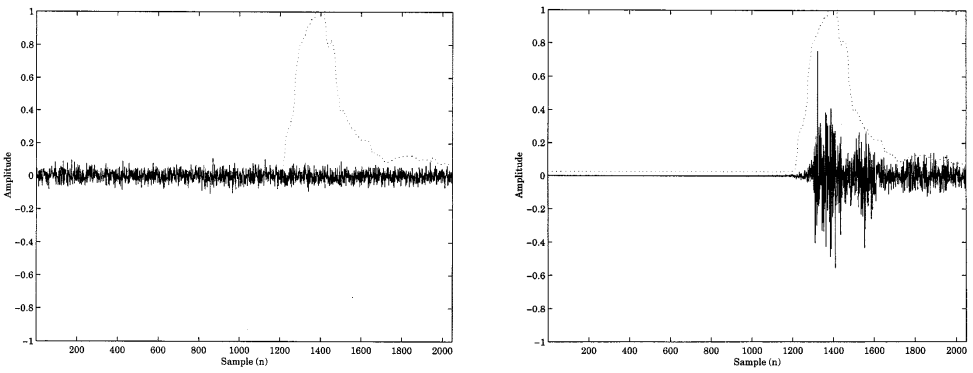


(a) (b)

Figure 2.14.: (a) gain modification and (b) TNS scheme (from [2]

imized using prediction techniques. The parameters of a spectral LP synthesis filter are estimated via application of standard minimum MSE estimation methods.

The resulting prediction residual $e(k)$ is quantized and encoded using standard perceptual encoding, the prediction coefficients are transmitted as side information. The convolution operation associated with spectral domain prediction is associated with multiplication in time. Analogous to the source-system separation realized by LP analysis in the time-domain, TNS separates the time-domain waveform into an envelope and temporally flat "excitation". Then, because quantization noise is added to the flattened residual, the time-domain multiplicative envelope corresponding to $A(z)$ shapes the quantization noise such that it follows the original signal envelope (see Fig. 2.15).



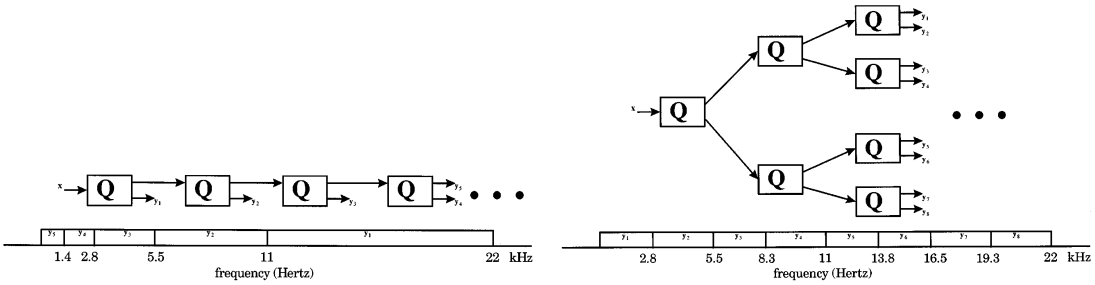
(a) (b)

Figure 2.15.: TNS example showing quantization noise and the input signal energy envelope for castanets:(a) without TNS (b) with TNS (from [2]

2.4.3. Wavelet Filter Banks [11]

Other than the previous described subband filter banks, which have a fixed bandwith of the subband filters, in wavelet decomposition the relative bandwith stays constant for the subband filters. This offers a flexible time-frequency tiling so that it is possible, for example, to approximate the critical bands. As Figure 2.16(a) shows, the output of a wavelet transform corresponds to the frequency subbands realized in 2:1 decimated output sequences from a QMF bank. Therefore, recursive DWT applications effectively pass input data trough a tree structured cascade of low-pass and high-pass filters followed by a 2:1 decimation at every node. The usual wavelet decomposition implements an octave-band filter bank structure shown in Figure 2.16(a).

Wavelet Packet (WP) or DWPT representations, in the other hand, decompose both the detail and approximation coefficients at each stage of the tree, as shown in Fig. 2.16(b).



(a) Subband decomposition associated with a discreet wavelet transform

(b) Subband decomposition associated with discrete wavelet package transform (DWPT). Note,that other , nonuniform decomposition trees are also possible

Figure 2.16.: DWT and DWPT

2.4.4. Linear Prediction

Although linear prediction is widely used in speech coding, its application to wideband audio coding has not been widely explored, because the LP analysis-synthesis framework is not well suited to model the nearly sinusoidal components present in steady-state audio.

A recent trend is to utilize LP coding in hybrid coding schemes for very low bit-rate audio coding with bit-rates below 16kb/s. Listening tests have shown that LP coders outperform sinusoidal coders for speech, while its the other way round for music signals.

In a p-th order forward linear predictor[12] the present sample is predicted from a linear combination of past samples

$$\hat{x}(n) = \sum_{k=1}^{p} a(k)x(n-k) \tag{2.22}$$

or in the z-domain

$$\hat{X}(z) = \left[\sum_{k=1}^{p} a(k)z^{-k}\right] X(z) \tag{2.23}$$

The prediction coefficients are computed so that the error between the predicted sample and the actual value

$$e(n) = y(n) - \hat{y}(n) \tag{2.24}$$

is minimized in least squares sense, e.g. with the Levinson-Durbin algorithm. The LP can be modified by substituting the unit delay filter z^{-1} by a first-order all pass filter [13]

$$D(z) = \frac{z^{-1} - \lambda}{1 - \lambda z^{-1}} \tag{2.25}$$

to obtain

$$\hat{X}(z) = \left[\sum_{k=1}^{p} a(k)D(z)^k\right] X(z). \tag{2.26}$$

This system is called frequency warped linear prediction WLP. For a choice $\lambda = 0.723$ the frequency warp represents the Bark frequency scale. The inherent Bark frequency resolution of the WLP produces a perceptually shaped quantization noise without an explicit psychoacoustic model.

2.5. Bit Allocation

Normally, when a perceptual audio coder is applied to a signal, a desired target bit rate is specified. To meet this target the information gathered in the time-frequency transform and other side information has to be coded.

2.5.1. Quantization

This is the stage where the lossy compression happens. According to the obtained masking threshold from the psychoacoustic model the frequency domain samples are quantized in such a way that the quantization noise stays below the masking threshold. Also frequency components below the masking threshold are removed.

The quantization itself can be uniform or or non-uniform, it may either be performed on scalar or vector data (VQ).

2.5.2. Entropy Coding

The quantized data still contains statistical redundancies, which can be further removed trough noiseless run length (RL) or entropy coding, e.g. Huffman, Arithmetic or Liv, Zempel and Welch (LZW) coding designs.

2.6. Coding Standards

2.6.1. MPEG 1[4]

The International Standards Organization/Moving Pictures Expert Group (ISO/MPEG) audio coding standard for stereo CD-quality audio was adopted in 1992 after four years of extensive collaborative research by audio coding experts worldwide. ISO 11172-3 consists of a flexible hybrid coding technique, which incorporates several methods including sub-band filter banks, transform coding, entropy coding, dynamic bit allocation, nonuniform quantizers, adaptive segmentation and psychoacoustic analysis. MPEG coders accept 16-bit PCM input data at sample rates of 32, 44.1 and 48 kHz, while available bit rates range from 32-192 kbit/s per channel for mono and stereo material.

The MPEG-1 architecture contains three layers of increasing complexity, delay and output quality.

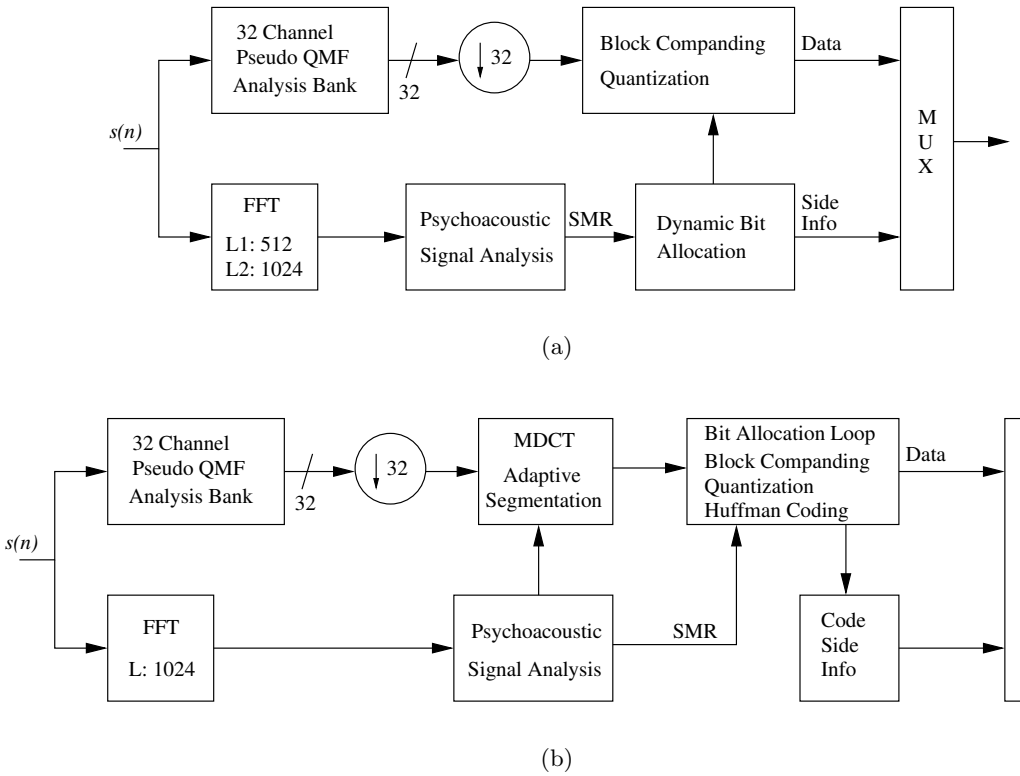


Figure 2.17.: ISO/IEC 11172-3 (MPEG-1): (a) Layer I/II encoder (b) Layer III encoder

2.6.1.1. Layer I and II

Filter Bank The analysis is done by a 32 band PQMF filter bank, the prototype filter is of order 511.

Psychoacoustic Model 1 For the model, a 512-point FFT for Layer 1 and a 1024-point FFT for layer 2 are used, for a brief description see section 2.3.4. The FFT is computed in parallel with the subband decomposition for each decimated block of 12 input samples.

Block Companding Quantization The subbands are block companded (normalized by a scalefactor) such that the maximum sample amplitude in each block is unity, then an iterative bit allocation procedure applies JND thresholds to select an optimal quantizer for each subband, simultaneously satisfying bit rate and masking thresholds. Scalefactor and quantizer choice are both coded and transmitted as side information.

Layer II enhances Layer I in three portions in order to realize reduced bit rates and enhanced audio quality. The perceptual model works on a higher resolution FFT, the maximum subband quantizer resolution is increased and scale-factor side information is reduced.

2.6.1.2. Layer III

Layer III adds a hybrid filter bank, each subband filter is followed by an adaptive MDCT for higher frequency resolution and pre-echo control. The MDCT switches between 6 points for pre echo control and 18 points for steady-state periods.

Bit allocation and quantization of the spectral lines is realized in a nested loop procedure that uses both nonuniform quantizers and Huffman coding. The inner loop adjusts the nonuniform quantizer step sizes for each block until the number of bits required to encode the transform components fall within the desired bit rate. The outer loop evaluates the quality of the coded signal (analysis-by-synthesis) in terms of quantization noise relative to the masking thresholds provided by the perceptual model.

2.6.2. MPEG 2[14]

The original MPEG 2 Audio standard finalized in 1994 just consists of 2 extensions to MPEG-1:

- Backwards compatible multichannel coding adds the option of forward and backwards compatible coding of multichannel signals including the 5.1 channel configuration known from cinema sound.
- coding at lower sampling frequencies adds sampling frequencies of 16 kHz, 22.05 kHz and 24 kHz to the sampling frequencies supported by MPEG-1.

Otherwise no new coding algorithms are introduced over MPEG-1 audio.

2.6.2.1. MPEG 2 Advanced Audio Coding[15]

Verification tests showed that giving up backwards compatibility and introducing new coding algorithms can improve the coding efficiency. As a result, the definition of a new work item led to the MPEG-2 Advanced Audio Coder finalized in 1997.

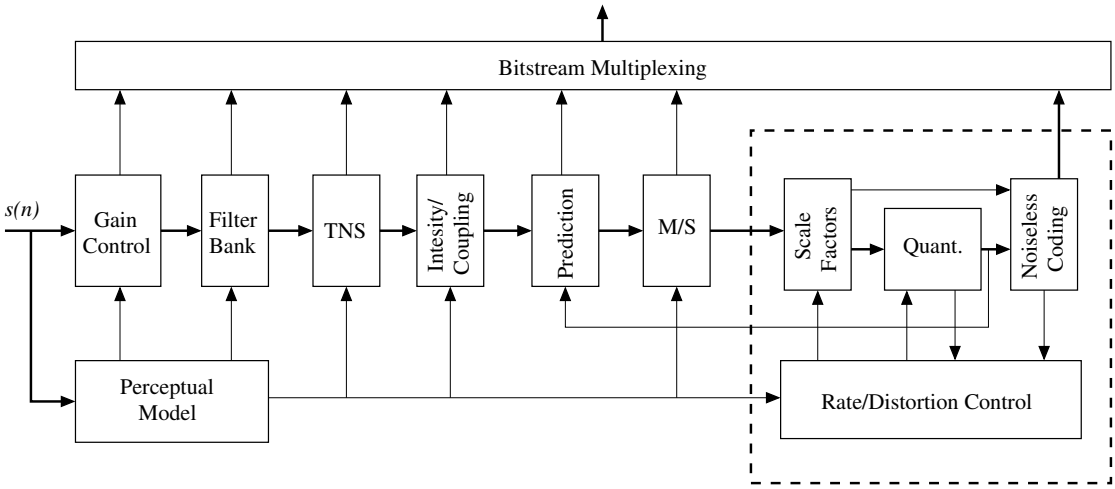


Figure 2.18.: MPEG-2/MPEG-4 Basic AAC Encoder Block Diagram

Encoding Structure (see Fig. 2.18):

Preprocessing In the sampling rate scalability profile, the preprocessing block is added in the input stage of the encoder. The preprocessing module consists of a polyphase quadrature filter (PQF), gain detectors and gain modifiers.

Filter Bank The conversion is done by MDCT with window switching, with windows length of 2048 for long and 256 for short windows (256 and 32 for SRS). For the 2048-length window the window shape is switched between a Kaiser-Bessel Derived (KBD) and a sine window depending on the input signal. To maintain block alignment, short windows only appear in sequences of eight consecutive short blocks.

Perceptual model AAC employs a perceptual model similar to MPEG-1 model 2.

TNS MPEG-2 AAC employs the TNS scheme described in Section 2.4.2.1.

Intensity/Coupling The M/S and intensity coding are improved over MPEG-1 Layer 3

Prediction To exploit correlations between the spectral components of succesive frames, backward adaptive predictors are employed. One backwards predictor works on each spectral component. Since short windows indicate signal changes, i.e. non-stationary signal characteristics, prediction is only used for long windows.

Quantization and coding The spectral components are grouped into bands whose bandwith resemble the critical bands closely. For each band one scalefactor is computed by which all spectral components within the band are scaled. The scaled spectral components are non-uniformly quantized and Huffman coded. The rate-distortion control assures that the target bit rate and the masking thresholds from the psychoacoustic model are met.

Bitstream Multiplexing The coded spectral components, scale factors, and side information are multiplexed to form the bit stream.

To allow tradeoffs between quality and memory/processing power requirements, the AAC offers three profiles:

Main Profile Uses all tools.

Low Complexity (LC) Profile The prediction tool is not utilized and the TNS order and bandwith are limited.

Sample Rate Scaleable (SRS) Profile Window length are a quarter of that for the other profiles, and it can provide a frequency scaleable bitstream.

2.6.3. MPEG 4[16]

The most recent MPEG standard ISO/IEC 14496 or MPEG-4 was adopted in 1998 after many proposed algorithms where tested. A second version was completed in 2000, enhancing the coding tools further.

MPEG-4 offers a great deal more than the perceptual audio coding tools described below, it also contains tool for coding speech (CELP and HVCX) and a text-to-speech coding tool.

2.6.3.1. General Audio Coder

The MPEG-4 General Audio Coder is based on the MPEG-2 AAC, but has been enhanced to provide better quality for low bit rates and fine grain scalability.

The additional tools available for MPEG-4 GA are[17]:

TwinVQ Transform Domain Weighted Interleave Vector Quantization replaces the AAC quantization and coding block for bit rates below 16 kb/s mono, where it is superior to AAC.

BSAC Bit sliced arithmetic coding replaces the noiseless AAC coding. To allow for small step scalability, the quantized values are grouped into frequency bands. Each of these groups contains quantized spectral values in their binary representation. Then the bits of a group are processed in in slices according to their significance (i.e. first all MSB bits in a group are processed etc.). These bit slices are encoded using arithmetic coding providing scalability steps of 1 kbit/s per audio channel.

Long Term Prediction The LTP replaces the backwards predictor from MPEG-2 AAC to avoid its complexity, while providing similar coding gain. It works like a speech coder, calculating the LTP in the time domain.

Perceptual Noise Substitution Noiselike frequency bands often don't require coding of the wave form in the band. Instead a noise detection module decides for each coder band whether noise substitution is possible or not, and if possible calculates the correct noise energy. The quantization and coding block is fed with a zeroed signal and the noise energy transmitted as side information to the decoder.

Error Resilience Tools Two classes of error resilience tools are defined, the first class contains algorithms to improve the error robustness of the source coding itself. The Virtual Code Books tool (VCB11) permits to detect serious error within the spectral data of an MPEG-4 AAC bitstream. The Reversible Variable Length Coding tool (RVLC) replaces the Huffman and DPCM coding of the scalefactors in an AAC bitstream. It uses symmetric codewords that can be decoded forward and backward. The Huffman Codeword Reordering tool (HCR) extends the Huffman coding of spectral data by placing some Huffman codewords at known positions in the stream, so that error propagation into these so called "priority codewords" can be avoided.

The second class consists of general tools for error protection The Error Protection Tool (EP) provides Unequal Error Protection (UEP) for MPEG-4 audio by ordering the bits into different error sensitivity classes and applies error correction to the parts (Both Cyclic Redundancy Check CRC and Forward Error Protection FEC can be applied).

2.6.3.2. Harmonic, Individual Lines plus Noise(HILN)[18]

The MPEG-4 parametric audio coding tools HILN permit coding of general audio signals at bit rates of 4 kbit/s and above using parametric representations of the audio signal. The basic idea of this technique is to decompose the input signal into components which are described by appropriate source models and represented by model parameters. Fig. 2.19 shows the block diagram of the HILN parametric audio coder. First the input signal is decomposed into different components and then the model parameters for the components' source models are estimated:

- An individual sinusoid is described by its frequency and amplitude.
- A harmonic tone is described by its fundamental frequency, amplitude, and the spectral envelope of its partials.
- A noise signal is described by its amplitude and spectral envelope.

The modeling of transients is improved by optional parameters describing their amplitude envelope.

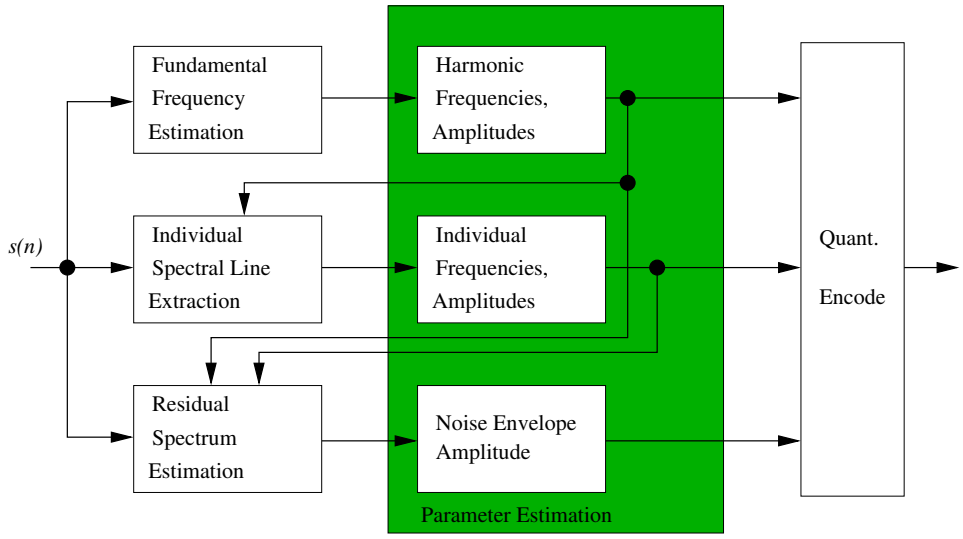


Figure 2.19.: MPEG-4 HILN Encoder Block Diagram

Due to the very low target bit rates only the parameters for a small number of components can be transmitted. Therefore a perceptual model is employed to select those components that are most important for the perceptual quality of the signal. The components parameters are finally quantized, coded, and multiplexed to form a bitstream.

2.6.3.3. Structured Audio

MPEG-4 Structured Audio is the first standard to allow the direct application of algorithmic structured audio techniques to the transmission of sound in a multimedia context. Algorithmic structured audio is the idea of using a general purpose software-synthesis language, and parameters to programs written in that language to represent sound for transmission.

The heart of the SA standard is a sound-synthesis language called SAOL, for "Structured Audio Orchestra Language". A program in SAOL describes a sound-processing or sound-synthesis algorithm. SAOL resembles C syntactically, but variables in SAOL contain audio signals, and built-in processing functions allow signals to be generated, mixed, filtered, processed and otherwise manipulated.

The bitstream header of a SA stream contains one or more algorithms written in SAOL, and the streaming data consists of Access Units containing parametric events written in SASL ("'Structured Audio Score Language").

At session startup, the SAOL algorithms are communicated to a reconfigurable synthesis engine. this engine configures itself accordingly to the SAOL programs.

During the streaming of the session, the Access Units are decoded into events, which are stored in a time-sorted list. The run-time scheduler, also specified in the standard, keeps track of this events and dispatches them when their time arrives.

Scheirer and Kim [19] developed the Generalzed Audio Coding method, where the

model is not fixed, but itself transmitted in the bitstream, and have shown, that SA is in fact such a generalized audio coder. That means, when new coding techniques arise, no standards have to be created for encoding and decoding, but SA can be employed to describe this models. As an example they implemented MPEG-1 Layer 1 and LPC decoding as SAOL programs.

2.6.4. AC3[20]

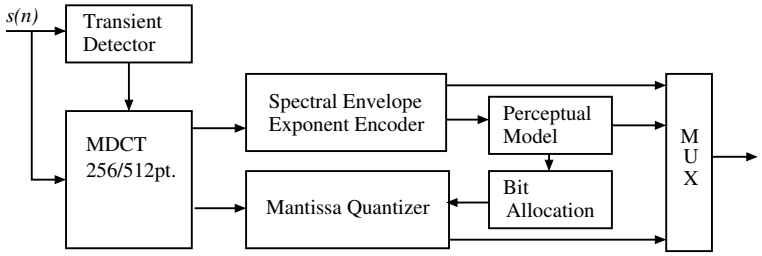


Figure 2.20.: AC3 encoder block diagram

The AC3 audio coder was developed by Dolby, who licenses it under the Dolby Digital trademark for cinema sound, and was adopted as audio part for the US HDTV system. It uses a 256/512 point adaptive MDCT filter bank. Up to four frequency-adjacent spectral components are grouped into a block and lumped together in groups spanning one to six transform blocks in time. For each group the maximum is identified and quantized as an exponent in terms of the number of left shifts required until overflow occurs, approximating the spectral envelope, then the spectral components of the group are normalized by the exponent to generate the mantissas. The perceptual model uses the spectral envelope to calculate the masking thresholds applied to the mantissa quantizer, where a bit allocation process similar to MPEG-1 takes place. A difference between AC3 and other coders is, that the information about the bit allocation (i.e. quantizers used in the mantissa quantization) is not transmitted to the decoder, but the decoder has the same perceptual model as the encoder and calculates the information itself.

2.6.5. PAC[21]

The Lucent Technologies PAC (Fig. 2.21(a)) system uses a signal-adaptive MDCT filter bank, with a long window of 2048 points for steady state segments, and a short window with 256 points for segments with transients or sharp attacks. Masking thresholds are used to select one of 128 exponentially distributed quantization step sizes in each of 49 or 14 bands. The coder bands are quantized using an iterative rate control loop in which thresholds are adjusted to satisfy bit-rate constraints and an equal loudness criterion that attempts to shape quantization noise such that its absolute loudness is constant relative to the masking threshold.

In an effort to enhance PAC at low bit rates, a switched MDCT/WP filter bank scheme (Fig. 2.21(b)) was introduced in EPAC.

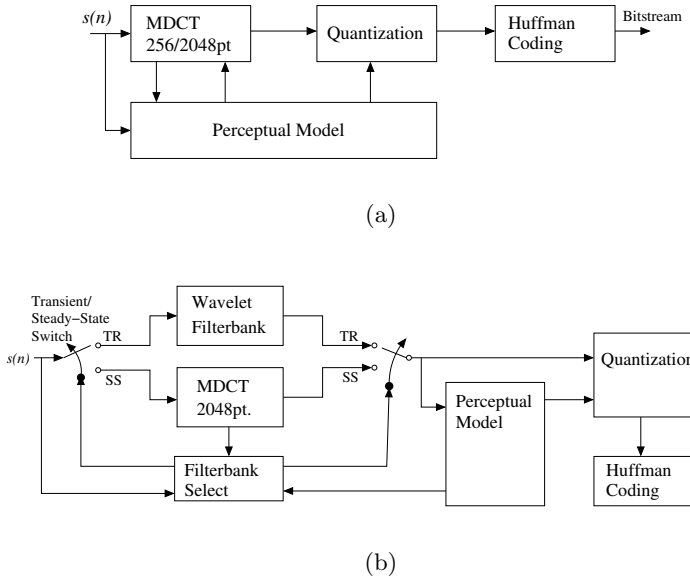


Figure 2.21.: Lucent Technologies PAC: (a) PAC and (b) EPAC

2.6.6. ATRAC[22]

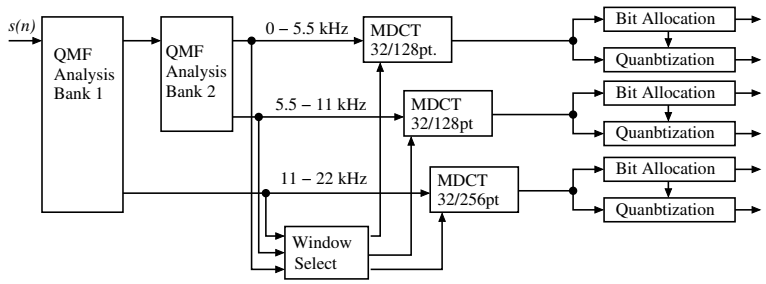


Figure 2.22.: ATRAC encoder block diagram

ATRAC was developed by Sony and is used for MiniDisc and in the Sony Dynamic Digital Sound (SDDS) cinema sound system. ATRAC uses a hybrid filter bank (see Fig. 2.22). The input signal is first split into three subbands using a tree structured QMF filter bank. Each of the three subbands is then transformed into the frequency domain using a MDCT with adaptive window length, the long window has a duration of 11.6ms (44.1 kHz) while the short window is 1.45ms for the highest frequency band and 2.9ms for the others two, providing a nonuniform time-frequency tiling. The MDCT spectral components are grouped into so called block floating units, which are separately scaled by a scalefactor and then quantized. The bit allocation algorithm is not specified, allowing for simple algorithms for low-cost, low-power devices like portable recorders and complex ones for other applications.

2.6.7. Ogg Vorbis

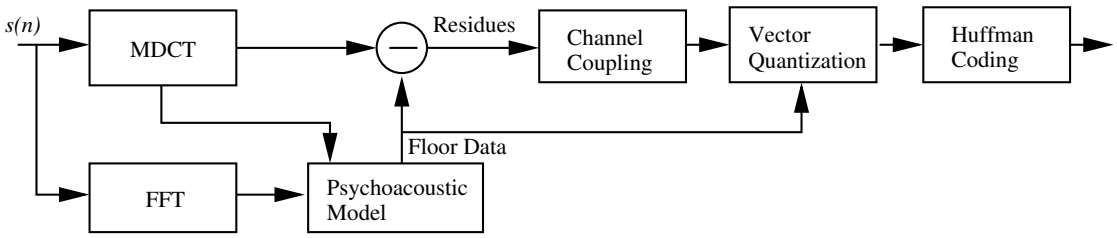


Figure 2.23.: Ogg Vorbis encoder blockdiagram

Ogg Vorbis is an open source perceptual audio coder [1] released under a BSD style license (see appendix B), meaning that no license fees have to be paid and everybody can contribute to the development of the format. Figure 2.23 shows the block diagram of the encoder, which uses following techniques:

MDCT The MDCT filter bank can use two different window lengths, window length can be powers of 2 ranging from 64 to 8192 points. The window used is

$$h(n) = sin\left(sin\left(\frac{n\pi}{2M}\right)^2 \frac{\pi}{2}\right). \tag{2.27}$$

Psychoacoustic model The perceptual model computes the global masking threshold curve which is either coded as LPC curve (deprecated) or approximated by a piecewise linear curve, the so called floor. This floor is subtracted from the MDCT spectrum components, leaving the residues for further encoding.

Channel Coupling The residues can be channel coupled, exploiting interchannel correlation, channel coupling is only available for stereo sources, although Vorbis is able to code up to 99 channels.

Vector quantization, Huffman coding The residues and floor parameters are vector quantized and Huffman coded.

One big difference between Vorbis and other codecs is that in Vorbis the codebooks for VQ and Huffman coding are not fixed in the specification, but transmitted in the header of the stream. While providing flexibility for signal adapted codebooks and optimization of codebooks at the encoder side, this also means that a vorbis stream can only be decoded correctly when the header information is received error free by the decoder, which causes some problem when streaming Vorbis streams over channels where information can be lost (e.g. streaming via RTP/UDP).

2.7. Low Latency Coding

Although all mentioned coding standards reach high coding gains, they are not suitable for applications where low latencies are required, because of their system inherent delay

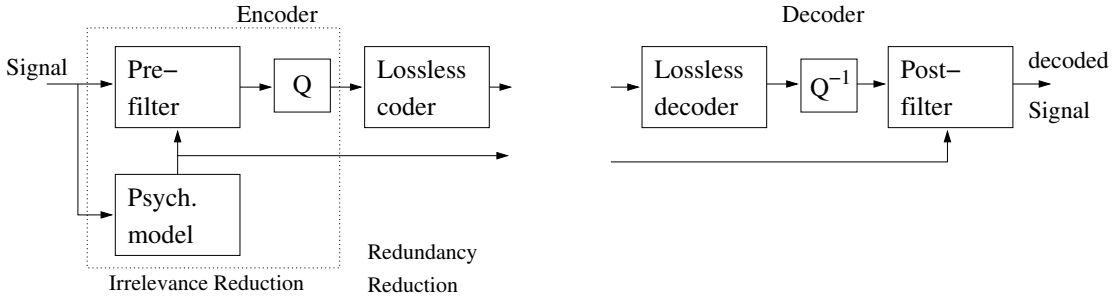


Figure 2.24.: Low latency coding scheme [23]

due to block processing. So Schuller et al[23] chose to use predictive coding for lossless coding, which has no inherent system delay, and adaptive pre- and post-filtering for irrelevance removal. Which this choice they also made the irrelevance and redundancy reduction independent of each other.

2.7.1. Pre- and Post-Filter

The irrelevance reduction unit consists of a psychoacoustically controlled time-varying pre-filter followed by a quantizer. The psycho-acoustic part is block-based, with very short blocks of 128 samples to reduce delay. The pre-filter has a frequency response inverse to the masking threshold, meaning the signal is normalized to its masking threshold. Its filter coefficients are computed with techniques from LPC analysis, using the masked threshold as short term power spectrum. To reduce audible artifacts introduced by hard switching of the filter coefficients between adjacent blocks, the filter coefficients of the lattice-structured pre-filter are obtained by linear interpolation. The pre-filter is followed by a simple uniform constant step size quantizer, in the proposed system a simple rounding operation to the nearest integer. The parameters of the pre-filter are transmitted as side information to the decoder, where the signal is post-filtered with the inverse frequency response of the pre-filter, i.e. the masked threshold, simply normalizing the quantization noise to the threshold. Bit rate control of the signal-dependent bit-stream after the quantizer is achieved by a simple attenuation of the pre-filtered signal, this increases the effective step size of the quantizer, leading to audible quantization noise but a reduced bit-rate.

2.7.2. Lossless Coding

The lossless coding section uses a new method, called Weighted Cascaded LMS Predictor (WCLMS), consisting of three ingredients: 1)normalized LMS, 2) cascading of the normalized LMS predictors, and 3) PMDL weighting of the cascaded predictors.

Normalized LMS Prediction LMS is an efficient and long used algorithm that minimizes adaptively the least square error, with a complexity linear in the order of the predictor.

Cascading the LMS predictors Three predictors with different orders are cascaded, i.e. the prediction error of the preceding predictor is used as input for the following predictor. The normally real outputs of the predictors are reduced to 8-bit integers.

PMDL (Predictive Minimum Description Length) Weighting The output of all three predictors are combined to form a final predictor. The weights for the individual predictors are based in how well a predictor has predicted the signal in the past, using the PMDL principle, which has a close connection to Bayesian statistics.

2.7.3. Entropy Coding of Prediction Errors

An adaptive Huffman coding scheme with a delay of only 17 samples was chosen, after it surprisingly showed that it achieves comparable bit-rates to a block-based Huffman coder with block lengths of 4096. This leads to an overall delay of 128 + 17 samples for the combination of pre- and post-filter with the WCLMS lossless unit the adaptive Huffman coding, which are both in the indentended order of 200 samples, leading to a encoding/decoding delay of 6ms at a sampling rate of 32kHz.

2.7.4. Experimental Results and Conclusion

The combination of order 200, 80, and 40 leads to an average bit rate of 2 bits/sample. Listening tests in comparison with a PAC coder set to the same average bit rate showed that the perceived quality is comparable at a much lower delay than the PAC, which has a delay of about 2600 samples.

2.8. Quality Assessment

In many situations, and particularly in the context of standardization activities, performance measures are needed to evaluate whether one of the established or emerging techniques in perceptual audio coding is in some sense superior to available methods. Perceptual audio coders are most often evaluated in terms of bit rate, complexity, delay, and output quality. Of these all but output quality can be quantified in straightforward objective terms. Reliable and repeatable output quality assessment on the other hand, presents a significant challenge, since classical objective measures of signal fidelity such as SNR or THD are inadequate, because its possible to achieve transparent quality even for a unweighted SNR of down to 13dB.

As a result, time consuming and expensive subjective listening tests have been required to measure the small impairments that mostly characterize the high-quality perceptual coding algorithms.

To minimize the need for this subjective tests, an objective quality assessment method has been standardized by the ITU.

2.8.1. Subjective Quality Tests

2.8.1.1. Small Impairments of High Quality Audio

The commonly used methodology for conducting formal listening tests is the ITU-R Recommendation BS.1116 [24]. The test uses critical audio test material, presented to trained expert listeners in a double-blind triple-stimulus with hidden reference method. Three stimuli ("A", "B", "C") are presented to the listener, A is always the reference, while B and C are randomly assigned to the hidden reference and the test signal. The listener has to grade the impairment of B and C relative to the reference signal, based on the continuous five grade impairment scale in table 2.3. A subjective difference grade is computed by subtracting the score assigned to the hidden reference from the score assigned to the test signal.

The listening panel for subjective test should consist of expert listeners, and the test should be preceded by a training phase to familiarize the test subjects to the grading system and to the artefacts under study.

A result of such an subjective listening test is given in table 2.4.

Impairment	Grade
Imperceptible	5.0
Perceptible, but not annoying	4.0
Slightly annoying	3.0
Annoying	2.0
Very annoying	1.0

Table 2.3.: ITU-R BS.1116-1 Grading Scale

2.8.1.2. Intermediate Quality Level

ITU-R BS.1116-1 is poor at discriminating small differences in quality at the bottom end of the scale, so it is not entirely suitable for evaluating lower quality audio systems which have emerged in the last years due to new applications with restricted data rates.

So a new subjective test methodology was proposed by the ITU in the recommendation BS 1534-1[26]. The methodology is called "Multi Stimulus test with Hidden Reference and Anchor (MUSHRA)". The method uses the original unprocessed programme material with full bandwidth as the reference signal (which is also used as a hidden reference) as well as at least one hidden anchor, which is a low-pass filtered version of the unprocessed signal with a bandwidth of 3.5 kHz. Additional anchors showing other impairments (bandwidth limitation to 7 or 10 kHz, reduced stereo image, additional noise, drop outs, packet losses or others) can be used. In each trial, the subject is presented with the reference version, all versions of the test signal processed by the systems under test, the hidden reference and the anchors.

The grading process uses a graphical continuous quality scale (CQS), which is divided into five equal intervals with the adjectives as given in table 2.5 from top to bottom, adopted from evaluation of picture quality.

Group	Algorithm	Rate	Mean Diff. Grade	Transparent Items	Items Below -1.0
1	AAC	128	-0.47	1	0
	AC-3	192	-0.52	1	1
2	PAC	160	-0.82	1	3
3	PAC	128	-1.03	1	4
	AC-3	160	-1.04	0	4
	AAC	96	-1.15	0	5
	MP1-L2	192	-1.18	0	5
4	ITIS	192	-1.38	0	6
5	MP1-L3	128	-1.73	0	6
	MP1-L2	160	-1.75	0	7
	PAC	96	-1.83	0	6
	ITIS	160	-1.84	0	6
6	AC-3	128	-2.11	0	8
	MP1-L2	128	-2.14	0	8
	ITIS	128	-2.21	0	8
7	PAC	64	-3.09	0	8
8	ITIS	96	-3.32	0	8

Table 2.4.: Comparison of Standardized Two-Channel Algorithms (after [25])

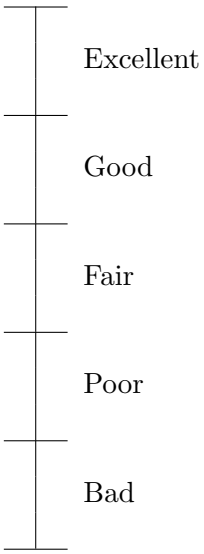


Table 2.5.: ITU-R BS.1534-1 Continuous Quality Scale.

2.8.2. Objective Quality Assessment with PEAQ

To reduce the necessity for subjective listening tests, schemes for objective quality measurements have been developed since the late 70's. More recently, the ITU started a standardization process which led to the ITU-R 1234.7 Recommendation "Perceptual Assessment of Audio Quality (PEAQ)" [27], an overview is given in [28] and a detailed examination of the model was conducted by Kabal[29]. Figure 2.27 shows the high-level representation of the model. In general it compares a signal that has been processed in some way with the corresponding time-aligned original signal. Concurrent frames of the original and processed signal are each transformed to a basilar membrane representation, and differences are analyzed further as a function of frequency and time by a cognitive model. The latter extracts perceptually relevant features which are used to compute a measure of quality. As indicated in the figure, a number of intermediate model output variables (MOV) are available. A selected set of these MOVs is mapped to an objective grade, which is comparable to the subjective difference grade (SDG) of subjective listening tests. The mapping was established by minimizing the difference between the distribution of objective measurements and the corresponding distribution of mean subjective qualities for an available data set.

2.8.2.1. Perceptual Measurement Concepts

Distance from Masked Threshold The error signal, which is the difference between the original and the processed signal, is compared to the masking threshold of the original signal. An error at a certain time and frequency is considered inaudible if its magnitude is below the masked threshold.

Comparison of Internal Representation The excitation patterns on the basilar membrane are modelled by simulating the signal transformations performed in the ear. The excitation patterns of both the original and processed signal are compared to derive the quality measurement. Since this approach is much closer to the function of the auditory system than the masked threshold concept, it is suited better for modeling more complex auditory phenomena.

Spectral Analysis of Errors Some effects, such as the perception of fundamental frequency and associated harmonic structure, are easier to model using linear spectra instead of basilar membrane excitation patterns.

2.8.2.2. Ear Model

The peripheral ear model contains all steps that transform the incoming sound into a basilar membrane representation (excitation patterns).

Absolute Threshold see Sect. 2.3.1 *Absolute Threshold of hearing.*

Perceptual Frequency Scale see Sect. 2.3.2 *Critical Bands*.

Excitation The hair cells in the cochlea generate neural activity in response to the vibration of the basilar membrane caused by incoming sound events. This patterns are approximately triangular in shape, and can be deduced from the masking threshold of a noise masking a tone (see fig. 2.6).

Alternatively, the excitation patterns can be seen as the output of the auditory filter bank, which models the frequency response of the basilar membrane.

Detection The excitation patterns are processed and stored in the brain. Three different kinds of memories are distinguished: echoic memory, short-term memory and long-term memory. Subjective listening test rely on both echoic and short-term memory to detect small level differences between signal. The detection threshold for differences in signal levels, called *just noticeable level difference (JNLD)*, depends on the sound pressure level of the input signal.

Masking see Sect. 2.3.3 *Masking*.

Loudness and Partial Loudness The perceived loudness of an audio signal depends not only on its sound pressure level but also on its duration and its temporal and spectral structure. The partial loudness of a signal is the perceived loudness after it has been reduced by a masker. This is important in perceptual measurements, as partial loudness takes into account the reduction in perceived loudness of an audible distortion due to the masker.

Implemented Ear Models PEAQ utilizes two different ear models, one FFT-based with a blocklength of 2048(see fig. 2.25), and the other filter bank based (see fig. 2.26 consisting of 40 pairs of linear phase filters corresponding to auditory filter widths.

2.8.2.3. Cognitive Model

The perceptual representation provided by the ear is mapped to a cognitive representation that is more difficult to describe. It depends heavily on the listeners world knowledge about certain sounds. Nevertheless, some reasonable assumptions can be made about the cognitive process underlying a quality judgement. Since computer programs typically have little world knowledge, the original signal has to be used as a reference for the cognitive model, however only a part of the missing world knowledge is compensated by using the original signal as reference. For example, for more noiselike input signals, and aesthetically pleasing processing, the quality rating of the resulting signal may be higher than a judgement strictly based on the comparison of input and output. This behavior is difficult to model without knowledge of the ideal audio signal that is in the mind of the listener. Other aspects concern the different perception of adding and removing certain time-frequency components, linear and nonlinear distortions, learning, and the fact that

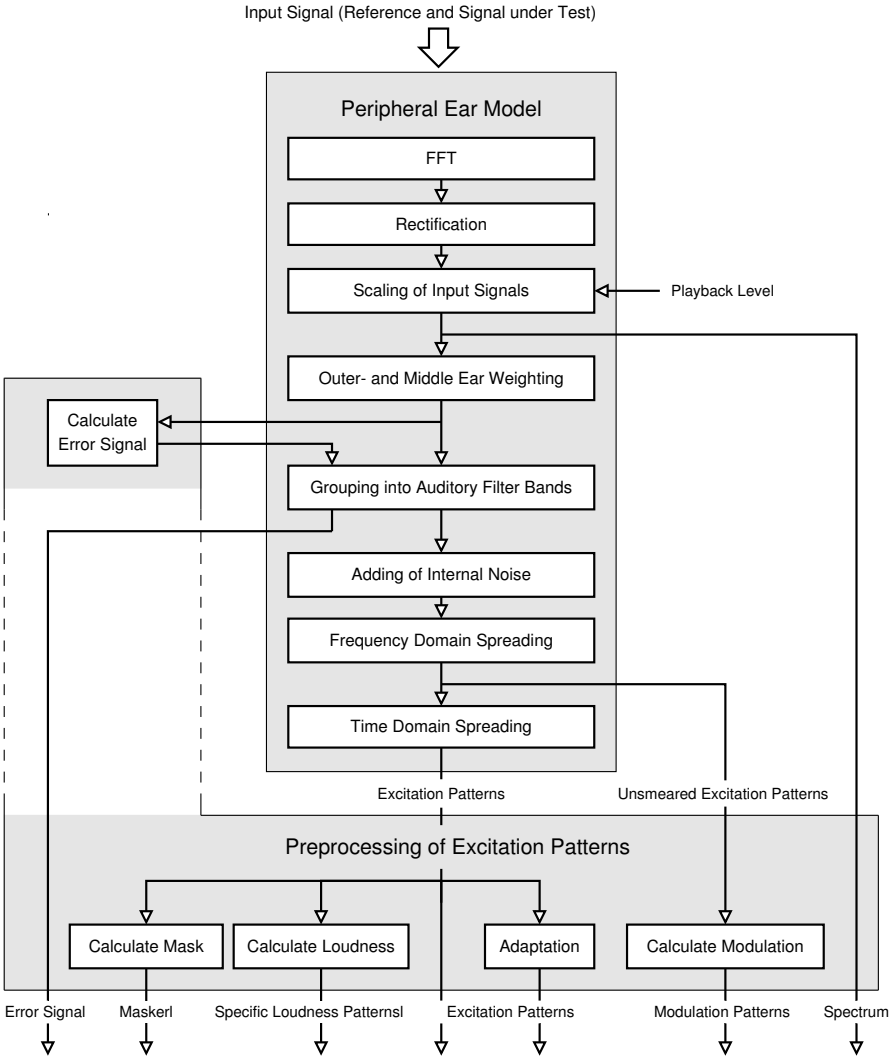


Figure 2.25.: FFT based ear model and preprocessing of excitation patterns (after [28])

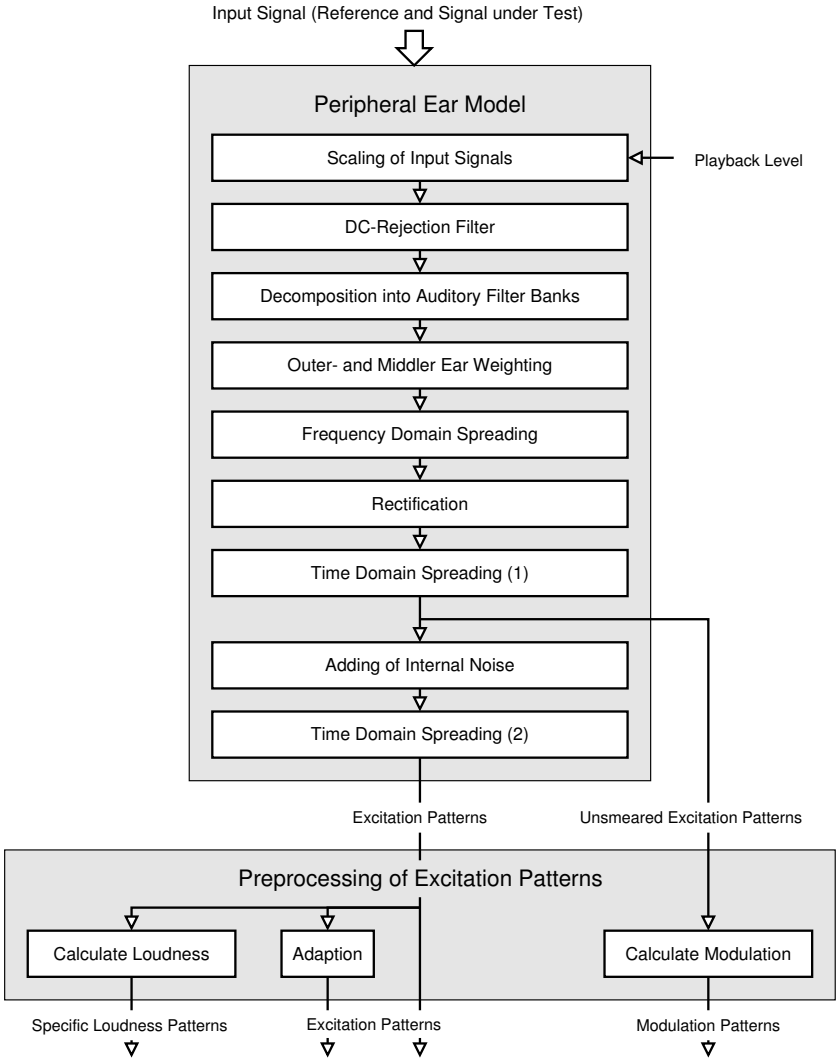


Figure 2.26.: Filter bank based ear model and preprocessing of excitation patterns(after [28]

certain regions in the audio signal carry more information and therefore may be more important than others when assessing distortions.

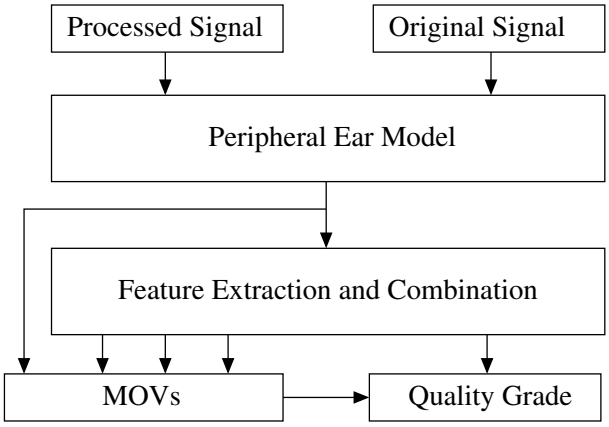


Figure 2.27.: High-level representation of the PEAQ model

Two versions of the model exist, one for high processing speed for real time applications, called the basic version, and one for applications requiring the highest achievable accuracy, called the advanced version.

2.8.2.4. Basic Version

The basic version uses only the FFT-based ear model, and employs both the concept of comparing internal representations and the concept of masking threshold. The restrictions arising form the poor temporal resolution of the FFT-based ear model are partly compensated by a higher number of MOVs and an increased spectral resolution (as compared to the advanced version). The variables derived from the ear model measure the loudness of distortions, the amount of linear distortions, the relative frequency of audible distortions, changes in the temporal envelope, a noise-to-mask ratio, noise detection probability and harmonic structure in the error signal.

2.8.2.5. Advanced Version

The advanced version of PEAQ uses the FFT-based ear model as well as the filter bank based ear model (see Fig. 2.28). The masking threshold concept is applied using the FFT-based ear model, whereas the concept of comparing internal representations is applied using the filter bank based ear model. The variables based on the FFT include a noise-to-mask ratio and a cepstrum like measure of harmonic structure in the error signal.

2.8.2.6. Implementations

A current overview of systems implementing PEAQ can be found on the PEAQ information web site[30]. The software used in this thesis for objective quality assessment is

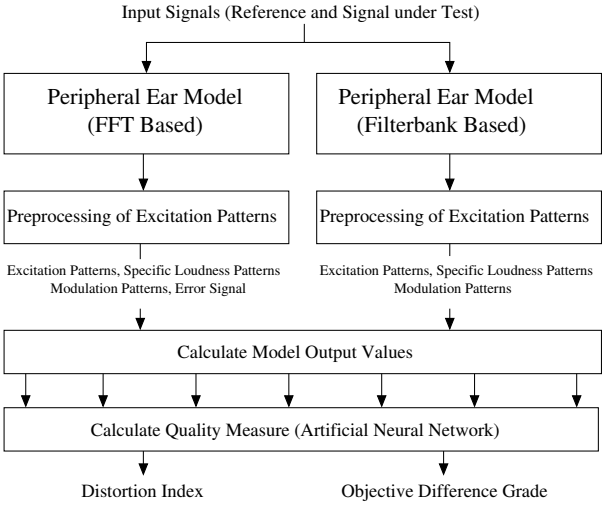


Figure 2.28.: Block diagram of the PEAQ advanced version

EAQUAL[31], which implements the basic model.

3. Mixing Two MDCT-Based Audio Streams

3.1. Basic Concepts of the MDCT

The Modulated Discrete Cosine Transform (MDCT), also called Time Domain Aliasing Cancellation (TDAC) filter bank or Modulated Lapped Transform (MLT), is defined as

$$X(k) = \sum_{n=0}^{2M-1} x(n) p_{k,n} \tag{3.1}$$

where

$$p_{n,k} = h(n)\sqrt{\frac{2}{M}} \cos\left[\frac{(2n+M+1)(2k+1)\pi}{4M}\right]. \tag{3.2}$$

This can also be written as a linear transformation of the form

$$\mathbf{X} = \mathbf{P}^T \mathbf{x} \tag{3.3}$$

where $\mathbf{P}$ is the lapped transform matrix, with the MLT basis functions $p_{n,k}$ as its columns and $\mathbf{x}$ is an extended signal block having $2M$ samples

$$\mathbf{x} = [x(mM - 2M + 1) \quad x(mM - 2M + 2) \quad \dots \quad x(mM - 1) \quad x(mM)]^T \tag{3.4}$$

with m being the block index. Since the MLT is an orthogonal transformation, the inverse MLT for a single block is simply

$$\tilde{\mathbf{x}} = \mathbf{P}\mathbf{X}. \tag{3.5}$$

The signal is reconstructed by overlapping and adding adjacent blocks.

3.2. Superposition

Lets assume a linear combination of two independent signals

$$x = a x_a + b x_b. \tag{3.6}$$

Then the transformed signal, assuming the block boundaries for both signals are aligned and block lengths are equal, is

$$\mathbf{X} = \mathbf{P}^T \mathbf{x} = \mathbf{P}^T (a\mathbf{x}_a + b\mathbf{x}_b) = a\mathbf{P}^T \mathbf{x}_a + b\mathbf{P}^T \mathbf{x}_b = a\mathbf{X}_a + b\mathbf{X}_b. \tag{3.7}$$

The inverse is

$$\tilde{\mathbf{x}} = \mathbf{P}\mathbf{X} = \mathbf{P}\left(a\mathbf{X}_a + b\mathbf{X}_b\right) = a\mathbf{P}\mathbf{X}_a + b\mathbf{P}\mathbf{X}_b. \tag{3.8}$$

Thus two audio streams with equal window lengths can simply be mixed by adding the two scaled transforms block-wise.

3.3. Changing the Block Length

When two MDCT coded audio streams with different window lengths are mixed, at least for one of the streams the window sequence has to be altered to get equal window sequences, either to a shorter block length (called interpolation in MDCT according to interpolation in the the time domain since the number of blocks increases) or to a longer block length, called decimation in MDCT.

Figure 3.1 shows a sequence of shorter and longer blocks. Throughout the following sections we assume always that the starting blocks of the sequences are aligned and block lengths are a power of 2, since most existing MDCT based coders have such block lengths.

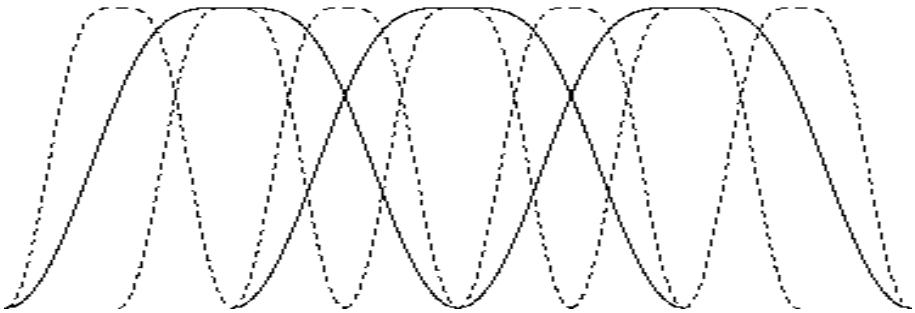


Figure 3.1.: Block sequence for changing the block length in MDCT. In this example the shorter block length is half the longer block length.

3.3.1. Decimation in the MDCT Domain

First, we just examine one long block. Figure 3.2 shows an example where the short blocks are half as long as the long block. We note that the first and last short block extend over the boundaries of the long block. The number of short blocks per long block is

$$s = (\frac{2M_l}{M_s} - 1) + 2. \tag{3.9}$$

. To get the contribution of a single short window with index $o = 0 \dots s-1$ to the long window, we compute the inverse transform of the short window, and pad the resulting time domain data with zeros to get an extended long block with length $2M_l + 2M_s$

$$\hat{\mathbf{x}} = [\mathbf{0} \quad \mathbf{P}\mathbf{X}_l \quad \mathbf{0}]^T \tag{3.10}$$

where the length of the zero padding at the beginning is oM_s and the length of the zero padding at the end is $(s-o-2)M_s$. An other way to get this extended transform of the short window is to compute the inverse transform with an extended transform matrix where $\mathbf{P}_s$ is padded at the top and bottom with zero row vectors according to

the aformentioned padding of the the audio data

$$\hat{\mathbf{P}}_{s,o} = \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_s \\ \mathbf{0} \end{bmatrix} \tag{3.11}$$

Then we compute the long window transformation of $\hat{\mathbf{x}}$ starting at $\hat{\mathbf{x}}(M_s)$. This can also be achieved by computing the transformation for the whole extended block with an extended transform matrix where $\mathbf{P}_l$ is padded with M_s zero rows at the top an the bottom:

$$\hat{\mathbf{P}}_l = \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_l \\ \mathbf{0} \end{bmatrix}. \tag{3.12}$$

The complete reverse and forward transformation to get the contribution of the short block o to the long block is then

$$\mathbf{X}_{l,o} = \hat{\mathbf{P}}_l^T \hat{\mathbf{P}}_{s,o} \mathbf{X}_{s,o}. \tag{3.13}$$

Since the matrix multiplication is zero for all row vectors of $\mathbf{P}_{s,0}$ this can also be computed by only taking the rows p_l^i from the extendend long transform, where i is the row index of $\mathbf{P}_{s,o}$ with non-zero entries:

$$\mathbf{X}_{l,o} = \hat{\mathbf{P}}_{l,o}^T \mathbf{P}_s \mathbf{X}_{s,o} \tag{3.14}$$

Or with the decimation matrix $\mathbf{D}_o$ defined as

$$\mathbf{D}_o = \hat{\mathbf{P}}_{l,o}^T \mathbf{P}_s \tag{3.15}$$

$$\mathbf{X}_{l,o} = \mathbf{D}_o \mathbf{X}_{s,o}. \tag{3.16}$$

To get the complete long transform coefficients, the contributions of all short blocks are simply added up

$$\mathbf{X}_l = \sum_{o=0}^{s-1} \mathbf{X}_{l,o}. \tag{3.17}$$

3.3.2. Interpolation in the MDCT Domain

Like in the previous section, we first look at the interpolation of one long block into a sequence of short blocks like in figure 3.2. To compute the transform of the short blocks, we just perform the reverse transformation of the long block and pad the resulting vector with M_s zeros at both tails:

$$\hat{\mathbf{x}} = [\mathbf{0} \quad \mathbf{P}_l \mathbf{X}_l \quad \mathbf{0}]^T \tag{3.18}$$

$\hat{\mathbf{x}}$ can also be computed by transforming $\mathbf{X}_l$ with an extended transform matrix, where $\mathbf{P}_l$ is padded with M_s zero rows at the top an the bottom:

$$\hat{\mathbf{P}}_l = \begin{bmatrix} \mathbf{0} \\ \mathbf{P}_l \\ \mathbf{0} \end{bmatrix} \tag{3.19}$$

One single short block, o being the short block index, is then computed by

$$\mathbf{X}_{s,o} = \mathbf{P}_s^T \hat{\mathbf{x}}_o \tag{3.20}$$

where $\hat{\mathbf{x}}_o$ is the part of the long block signal needed for transformation

$$\mathbf{x}_o = [\hat{\mathbf{x}}[oM_s - 2M_s + 1] \quad \hat{\mathbf{x}}[oM_s - 2M_s + 2] \quad \dots \quad \hat{\mathbf{x}}[oM_s - 1]]^T. \tag{3.21}$$

By taking a submatrix $\hat{\mathbf{P}}_{l,o}$, containing only the row vectors

$$\hat{\mathbf{p}}_l^q, \quad q = (o-1)M_s + 1 \dots oM_s$$

of $\hat{\mathbf{P}}_l$, a single short block can be computed from the long block by

$$\mathbf{X}_{s,o} = \mathbf{P}_s^T \hat{\mathbf{P}}_{l,o} \mathbf{X}_l. \tag{3.22}$$

We can now define a series of direct transformation matrices

$$\mathbf{U}_o = \mathbf{P}_s^T \hat{\mathbf{P}}_{l,o} \tag{3.23}$$

so that a single short block is

$$\mathbf{X}_{s,o} = \mathbf{U}_o \mathbf{X}_l. \tag{3.24}$$

To extend this single frame derivation to a sequence of long windows, we just assume, since superposition applies to the MDCT, the stream as a combination of streams, every stream consisting only of one long windowed segment of the sequence. As figure 3.1 shows, one short block contains data of two, or at frame boundaries, of three consecutive long blocks. To get the final sequence of short blocks, the contributions of the different long blocks are simply just added up. As long as both windows provide perfect reconstruction, the interpolated sequence also does.

If we compare the definitions of $\mathbf{U}_o$ and $\mathbf{D}_o$ it can be seen that

$$\mathbf{U} = \mathbf{D}^T \tag{3.25}$$

if long and short window lengths of the decimation and interpolation are equal.

3.4. Arbitrary Window Lengths and Positions

The approach for interpolating and decimating in the MDCT domain can of course be generalized to get direct transform matrices for arbitrary window sizes and positions (fig. 3.3).

To get the direct transformation matrix, we simply pad both transform matrices according to the non-overlapping parts, so that both have an extended length spanning from the begin of the front window to the end of the back window. The direct transform matrix is then:

$$\mathbf{C} = \hat{\mathbf{P}}_t^T \hat{\mathbf{P}}_s \tag{3.26}$$

where $\hat{\mathbf{P}}_t$ is the padded transform matrix of the target window and $\hat{\mathbf{P}}_s$ is the padded transform matrix of the source window. The direct transformation is then

$$\mathbf{X}_t = \mathbf{C}\mathbf{X}_s. \tag{3.27}$$

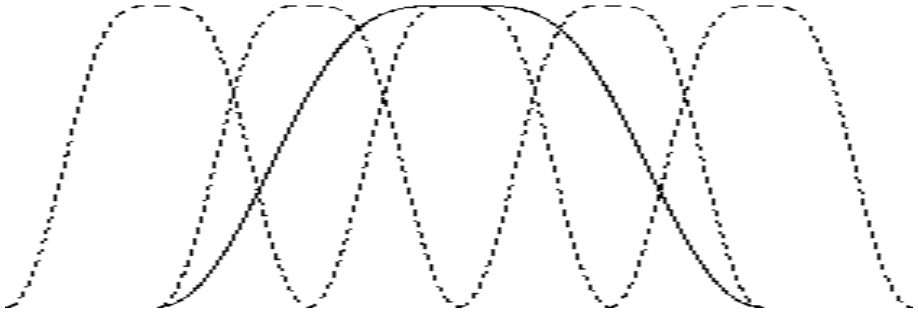


Figure 3.2.: Single long window with short window sequence of all blocks that contain data of the long window.

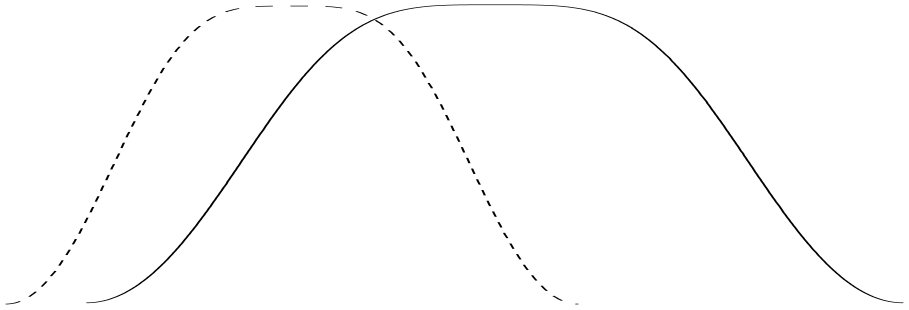


Figure 3.3.: Two arbitrary sized and positioned windows.

3.5. Computational Complexity

The direct transform matrices have of course the size $M_s \times M_l$. If no fast algorithms for calculating the MDCT would exist, this would be a advantage in complexity, since the reverse and forward transformation matrices have the size $2M_s \times M_s$ and $2M_l \times M_l$ respectively, the second one alone having more elements than the direct transformation matrix.

But since fast algorithms for the MDCT exist, which have a computational complexity of about one order of magnitude less than the matrix approach, direct transformation with fully occupied direct transform matrices would yield no computational advance over inverse and forward MDCT. Table 3.1 shows a comparison of the approximated average number of multiplications needed per resulting block when interpolating or decimating a MDCT stream. The fast MDCT numbers were determined using the MDCT algorithm implemented in the Ogg Vorbis source code.

As this table shows, the computational complexity for direct transformation is much higher than a inverse/forward transformation using a fast algorithm. To reduce the complexity, two different approaches were considered, one reducing the number of short windows computed per long window, i.e. ignoring the short windows that extend over the block boundaries of the long window, and the second setting all elements in the direct transformation matrices to zero which are below a certain threshold.

from	to	Vorbis	Direct
256	128	1,564	20,477
512	128	1,778	36,859
512	256	4,220	81,908
1024	128	1,949	69,624
1024	256	4,562	147,438
1024	512	1,0043	327,631
2048	128	2,098	135,154
2048	256	4,861	278,497
2048	512	10,641	589,750
2048	1024	22,714	1,310,523

(a) Interpolation

from	to	Vorbis	Direct
128	2048	2,158	2,162,688
128	1024	2,005	557,056
128	512	1,826	147,456
128	256	1,596	40,960
256	2048	4,917	2,228,224
256	1024	4,610	589,824
256	512	,4252	163,840
512	2048	10,689	2,359,296
512	1024	10,075	655,360
1024	2048	22,746	2,621,440

(b) Decimation

Table 3.1.: Comparison of the number of multiplications needed per target window.

To explore the signal degradation due to reducing the computational complexities, eight different test signals were interpolated and decimated using different settings for the threshold and the inclusion of boundary blocks. Tables 3.2 and 3.3 show the test results. The threshold indicates that elements in the direct transformation matrices, which levels are below the threshold are zeroed, the reference level being the level of the largest absolute value in all transformation matrices. If "Full" is ticked, the boundary blocks are included, otherwise excluded. The objective difference grades (ODG) were determined using EAQUAL with standard settings.

Figure 3.4 shows the ODG's for some window lengths changes.

The results show, that to achieve a complexity comparable to forward/inverse transform, the threshold for zeroing direct transform coefficients has to be quite high. The ODGs show that the effect of excluding the boundary blocks is independent of the threshold until a threshold of -20dB, a logical cause of the fact that the direct transform coefficients for the boundary blocks are relatively small, the influence being smaller for window changes with larger differences of the block length. This could be also expected, since a bigger difference means that the energy of the large window is spread over more small windows.

Especially when the window lengths only differ by a factor 2 the exclusion of the boundary box leads to clearly audible artifacts for certain signals. Figure 3.5 shows the error signal for an interpolated male speech signal. It can be seen, that the error signal is periodic with the length of the shorter window.

A good compromise between signal degradation and complexity would be a threshold of -40dB and including the boundary blocks. For this settings, the mean object difference grade is higher than -0.15 for all window length combinations, still being near indistinguishable audio quality.

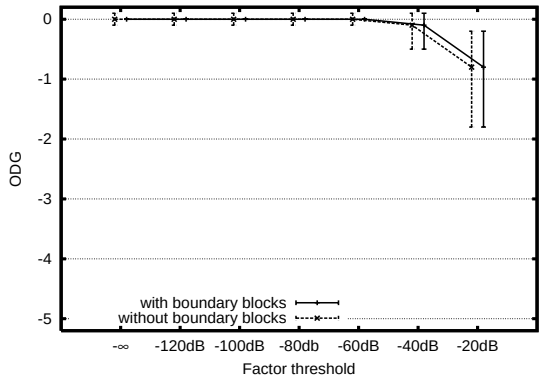
Direct transformation has at least one advantage over inverse/forward transformation.

Table 3.2.: Result table for interpolation.

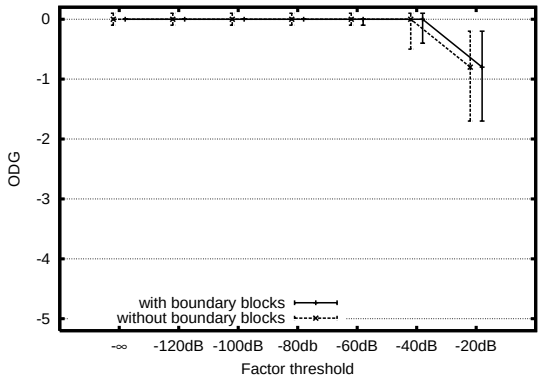
from/ to	Treshold	$-\infty$		-120 dB		-100 dB		-80 dB		-60 dB		-40 dB		-20 dB	
	Full	√		√		√		√		√		√		√	
256/ 128	MSE·10^{-6}	0.00	152.16	0.00	152.16	0.00	152.16	0.00	152.16	0.06	152.20	2.55	153.48	247.19	286.73
	max. Error	0.00	0.13	0.00	0.13	0.00	0.13	0.00	0.13	0.00	0.13	0.01	0.13	0.13	0.14
	ODG	0.01	-1.01	0.01	-1.01	-0.01	-1.01	-0.01	-1.01	-0.01	-1.01	-0.14	-1.02	-1.47	-1.52
	Multiplications	20,476	12,287	19,667	12,223	10,566	5,940	5,362	2,966	2,600	1,473	1,504	877	415	413
512/ 128	MSE·10^{-6}	0.00	6.60	0.00	6.60	0.00	6.60	0.00	6.60	0.06	6.65	3.29	8.47	291.97	291.97
	max. Error	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.02	0.03	0.10	0.10
	ODG	0.01	-0.09	0.01	-0.09	-0.01	-0.09	-0.02	-0.09	-0.02	-0.11	-0.15	-0.17	-1.43	-1.43
	Multiplications	36,859	28,669	32,508	26,700	19,044	15,931	9,580	8,090	4,631	3,848	2,398	2,209	949	949
512/ 256	MSE·10^{-6}	0.00	181.21	0.00	181.21	0.00	181.21	0.00	181.21	0.06	181.23	2.39	182.28	298.04	348.25
	max. Error	0.00	0.12	0.00	0.12	0.00	0.12	0.00	0.12	0.00	0.12	0.01	0.12	0.13	0.14
	ODG	0.01	-0.65	0.01	-0.65	-0.01	-0.65	0.00	-0.65	0.00	-0.66	-0.04	-0.67	-1.08	-1.11
	Multiplications	81,907	49,149	44,687	25,017	22,534	12,274	11,089	6,070	5,256	2,977	3,040	1,773	831	829
1024/ 128	MSE·10^{-6}	0.00	0.20	0.00	0.20	0.00	0.20	0.00	0.20	0.06	0.26	2.48	2.48	259.25	259.25
	max. Error	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.10	0.10
	ODG	0.01	-0.01	0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.02	-0.03	-0.13	-0.13	-1.41	-1.41
	Multiplications	69,624	61,434	57,940	53,877	34,355	32,286	17,482	16,460	8,501	7,998	4,526	4,527	1,702	1,702
1024/ 256	MSE·10^{-6}	0.00	6.26	0.00	6.26	0.00	6.26	0.00	6.27	0.06	6.31	3.27	7.85	231.87	231.87
	max. Error	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.01	0.03	0.09	0.09
	ODG	0.01	0.03	0.01	0.03	-0.01	0.03	0.00	0.03	0.01	0.03	-0.05	-0.05	-0.82	-0.82
	Multiplications	147,437	114,679	80,805	67,613	41,237	34,702	19,847	16,805	9,414	7,832	4,846	4,465	1,909	1,909
1024/ 512	MSE·10^{-6}	0.00	193.97	0.00	193.97	0.00	193.97	0.00	193.97	0.08	194.02	2.33	194.99	314.36	368.74
	max. Error	0.00	0.12	0.00	0.12	0.00	0.12	0.00	0.12	0.00	0.12	0.01	0.12	0.11	0.13
	ODG	0.01	-0.24	0.01	-0.24	0.00	-0.24	-0.00	-0.24	0.00	-0.25	0.02	-0.26	-0.55	-0.55
	Multiplications	327,630	196,598	94,992	51,185	46,569	24,943	22,543	12,278	10,567	5,985	6,112	3,565	1,663	1,661
2048/ 128	MSE·10^{-6}	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.06	0.07	2.60	2.60	279.75	279.75
	max. Error	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.10	0.10
	ODG	0.01	-0.01	0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.03	-0.03	-0.12	-0.12	-1.49	-1.49
	Multiplications	135,153	126,964	107,829	105,129	63,554	62,232	32,367	31,651	15,800	15,796	8,554	8,554	3,289	3,289
2048/ 256	MSE·10^{-6}	0.00	0.19	0.00	0.19	0.00	0.19	0.00	0.20	0.06	0.25	2.45	2.45	257.97	257.97
	max. Error	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.11	0.11
	ODG	0.01	-0.01	0.01	-0.01	-0.01	-0.01	-0.01	-0.01	0.00	-0.02	-0.05	-0.05	-0.96	-0.96
	Multiplications	278,496	245,738	146,025	137,234	74,775	70,483	36,330	34,253	17,301	16,287	9,150	9,150	3,422	3,422
2048/ 512	MSE·10^{-6}	0.00	6.16	0.00	6.16	0.00	6.16	0.00	6.16	0.08	6.22	3.73	8.24	221.79	221.79
	max. Error	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.01	0.03	0.09	0.09
	ODG	0.01	-0.05	0.01	-0.05	0.00	-0.05	-0.00	-0.05	-0.01	-0.05	-0.01	-0.02	-0.52	-0.52
	Multiplications	589,750	458,717	176,739	148,658	85,672	72,294	40,416	34,245	18,981	15,799	9,741	8,977	3,829	3,829
2048/ 1024	MSE·10^{-6}	0.00	172.98	0.00	172.98	0.00	172.98	0.00	172.98	0.08	173.01	2.67	173.75	344.62	365.75
	max. Error	0.00	0.11	0.00	0.11	0.00	0.11	0.00	0.11	0.00	0.11	0.01	0.11	0.12	0.13
	ODG	0.01	-0.18	0.01	-0.18	-0.01	-0.18	-0.00	-0.18	-0.00	-0.18	-0.00	-0.19	-0.54	-0.54
	Multiplications	1,310,523	786,392	195,980	103,781	94,693	50,286	45,452	24,693	21,189	12,001	12,255	7,149	3,327	3,325

Table 3.3.: Result table for decimation.

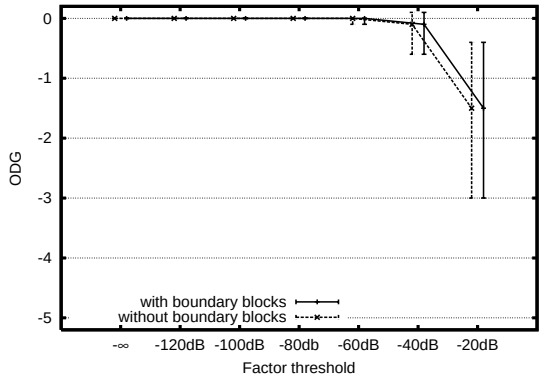
from/ to	Treshold	$-\infty$		-120 dB		-100 dB		-80 dB		-60 dB		-40 dB		-20 dB	
	Full	√		√		√		√		√		√		√	
128/ 256	MSE·10^{-6}	0.00	174.34	0.00	174.34	0.00	174.34	0.00	174.34	0.07	174.38	2.74	174.83	222.02	291.94
	max. Error	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.01	0.10	0.10	0.11
	ODG	0.01	-0.99	0.01	-0.99	-0.01	-0.99	-0.01	-1.00	-0.03	-0.99	-0.13	-1.00	-1.41	-1.45
	Multiplications	40,960	24,576	39,340	24,448	21,136	11,882	10,726	5,934	5,202	2,948	3,010	1,756	832	828
128/ 512	MSE·10^{-6}	0.00	6.30	0.00	6.30	0.00	6.30	0.00	6.30	0.07	6.36	3.22	8.20	211.88	211.88
	max. Error	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.01	0.03	0.10	0.10
	ODG	0.01	-0.10	0.01	-0.10	-0.01	-0.10	-0.01	-0.10	-0.03	-0.11	-0.15	-0.19	-1.30	-1.30
	Multiplications	147,456	114,688	130,050	106,812	76,188	63,730	38,328	32,364	18,528	15,396	9,596	8,840	3,800	3,800
128/ 1024	MSE·10^{-6}	0.00	0.20	0.00	0.20	0.00	0.20	0.00	0.20	0.06	0.26	2.73	2.73	236.88	236.88
	max. Error	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.09	0.09
	ODG	0.01	-0.01	0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.02	-0.02	-0.13	-0.13	-1.39	-1.39
	Multiplications	557,056	491,520	463,574	431,056	274,872	258,318	139,874	131,694	68,016	63,996	36,220	36,220	13,624	13,624
128/ 2048	MSE·10^{-6}	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.06	0.07	2.73	2.73	226.51	226.51
	max. Error	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.10	0.10
	ODG	0.01	-0.00	0.01	-0.00	-0.01	-0.00	-0.01	-0.01	-0.02	-0.02	-0.13	-0.13	-1.42	-1.42
	Multiplications	2,162,688	2,031,616	1,725,452	1,682,228	1,016,986	995,814	517,932	506,470	252,832	252,770	136,888	136,888	52,634	52,634
256/ 512	MSE·10^{-6}	0.00	173.64	0.00	173.64	0.00	173.64	0.00	173.64	0.07	173.67	2.61	173.98	259.69	318.98
	max. Error	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.01	0.10	0.12	0.12
	ODG	0.01	-0.67	0.01	-0.67	-0.01	-0.67	-0.01	-0.67	-0.01	-0.68	-0.06	-0.69	-1.05	-1.06
	Multiplications	163,840	98,304	89,388	50,038	45,076	24,550	22,182	12,142	10,514	5,956	6,082	3,548	1,664	1,660
256/ 1024	MSE·10^{-6}	0.00	6.17	0.00	6.17	0.00	6.17	0.00	6.17	0.06	6.23	3.22	8.24	230.24	230.24
	max. Error	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.00	0.03	0.01	0.03	0.09	0.09
	ODG	0.01	0.01	0.01	0.01	-0.01	0.01	-0.01	0.01	-0.01	0.01	-0.05	-0.05	-0.81	-0.81
	Multiplications	589,824	458,752	323,264	270,474	164,970	138,820	79,400	67,228	37,664	31,332	19,388	17,864	7,640	7,640
256/ 2048	MSE·10^{-6}	0.00	0.20	0.00	0.20	0.00	0.20	0.00	0.20	0.06	0.25	2.50	2.50	253.38	253.38
	max. Error	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.01	0.10	0.10
	ODG	0.01	-0.00	0.01	-0.00	-0.01	-0.00	-0.01	-0.00	-0.01	-0.03	-0.05	-0.05	-0.97	-0.97
	Multiplications	2,228,224	1,966,080	1,168,332	1,097,976	598,268	563,918	290,678	274,050	138,424	130,308	73,212	73,212	27,384	27,384
512/ 1024	MSE·10^{-6}	0.00	169.84	0.00	169.84	0.00	169.84	0.00	169.84	0.08	169.88	2.98	171.16	319.25	352.45
	max. Error	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.01	0.10	0.13	0.13
	ODG	0.01	-0.23	0.01	-0.23	0.00	-0.23	-0.00	-0.23	-0.01	-0.24	0.01	-0.25	-0.59	-0.59
	Multiplications	655,360	393,216	190,014	102,376	93,152	49,890	45,094	24,558	21,138	11,972	12,226	7,132	3,328	3,324
512/ 2048	MSE·10^{-6}	0.00	5.98	0.00	5.98	0.00	5.98	0.00	5.98	0.08	6.04	3.67	8.04	233.69	233.69
	max. Error	0.00	0.02	0.00	0.02	0.00	0.02	0.00	0.02	0.00	0.03	0.01	0.02	0.09	0.09
	ODG	0.01	-0.03	0.01	-0.03	0.00	-0.03	-0.00	-0.03	-0.01	-0.03	-0.07	-0.01	-0.56	-0.56
	Multiplications	2,359,296	1,835,008	707,048	594,680	342,734	289,198	161,688	136,992	75,936	63,204	38,972	35,912	15,320	15,320
1024/ 2048	MSE·10^{-6}	0.00	177.28	0.00	177.28	0.00	177.28	0.00	177.28	0.08	177.33	2.52	178.62	366.35	370.37
	max. Error	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.00	0.10	0.01	0.10	0.12	0.12
	ODG	0.01	-0.17	0.01	-0.17	-0.00	-0.17	-0.00	-0.17	-0.00	-0.17	-0.00	-0.17	-0.52	-0.51
	Multiplications	2,621,440	1,572,864	392,020	207,574	189,416	100,578	90,918	49,390	42,386	24,004	24,514	14,300	6,656	6,652



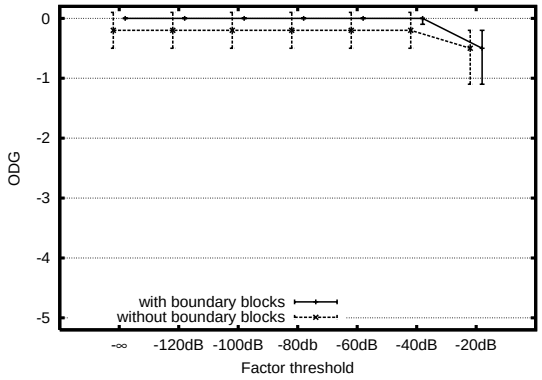
(a) Interpolating from 1024 to 256



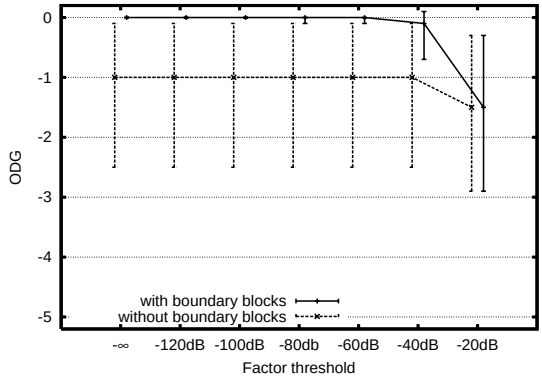
(b) Decimating from 256 to 1024



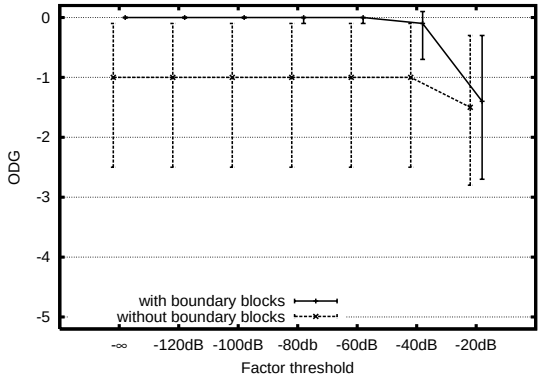
(c) Interpolating from 2048 to 128



(d) Decimating from 128 to 2048



(e) Interpolating from 256 to 128



(f) Decimating from 128 to 256

Figure 3.4.: ODGs for different window length changes.

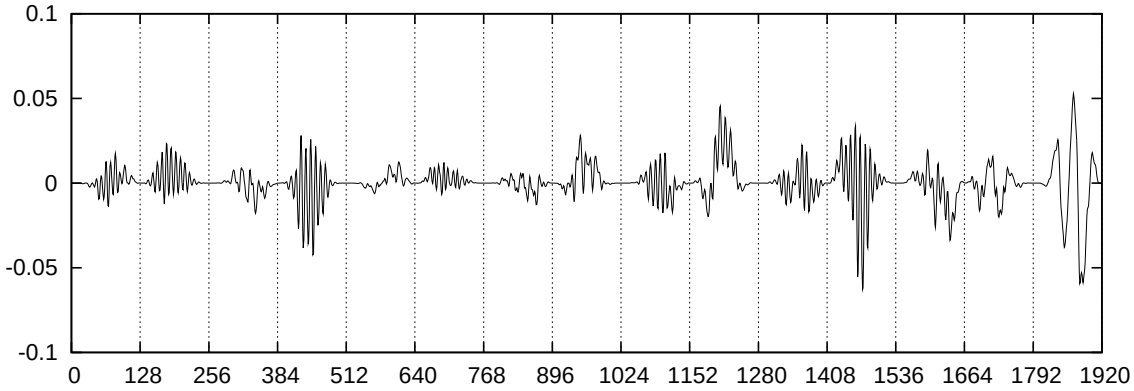


Figure 3.5.: Error signal for interpolation from 256 to 128 without boundary blocks for a male speech signal.

When combining two streams, in the normal case the block length of only one stream must be altered to allow addition with the other stream, in the ideal scenario of identical window lengths on both streams, no direct transformation has to take place.

Another aspect of the computational complexity is that the MDCT transform is only a part of the encoding/decoding process. A profiling of the reference Ogg Vorbis encoder showed that the forward MDCT takes up 10% of the processing time, while the psychoacoustic model is the most complex part of the encoding process, consuming about 40% of the processing time.

3.6. Latency

The minimum encoding latency for a MDCT stream with fixed window length is $2M$ samples, because enough data for one window has to be buffered before analyzing the first window. Decoding adds a minimal further delay of M samples, because before the first block can be completely decoded, the next block is needed for overlap and add. Thus the minimum latency for one coding/decoding cycle is $3M$ samples, not including time needed for computation and the delay introduced by the transport channel.

3.6.1. Comparison of Latency Between Direct Processing and Inverse/Forward MDCT

3.6.1.1. No Change of the Window Length

When the window length of a stream is not changed in processing, ideally no further latency is introduced, since every block can be processed independently. Inverse/forward transformation adds the above mentioned decoding delay of M samples. Direct processing thus gives an advantage of M samples, without consideration of further delay due to processing time. Due to the fact that the block boundaries at the encoders would not be exactly aligned, one stream has to be delayed a further variable delay of up to $M - 1$

samples.

3.6.1.2. Window Length Change

Direct processing with window length change yields no smaller latency than the inverse/forward transformation, since all source windows contributing to a target window have to be buffered anyway.

3.7. Effects on Psychoacoustic Processing

For irrelevance removal, perceptual coders use psychoacoustic models to determine the masking threshold. At least the estimation for tonal maskers is carried out with an FFT in most encoders, because the MDCT transform is not well suited for tonal estimation. The MDCT is a real transformation, so a strong tone at a frequency equal to one of the basis functions still results in a zero MDCT coefficient when tone and basis function have a phase difference of $\frac{\pi}{2}$.

Nevertheless, schemes for tonal estimation in the MLT exist, for example [32].

The estimation of noise maskers can be carried out in the MLT domain, like it is implemented in Ogg Vorbis.

On possible solution for calculating the masking threshold lies in the fact that masking curves are transmitted with the coded stream, either implicitly, like in the scalefactor and quantizer choices in MPEG audio coding, or explicitly, like the floor curves in Ogg Vorbis coded material. Estimating of the new masking curves should be less complicated with this information.

3.8. MDCT Stream Mixing Strategies

Depending on the stream structures of the two MDCT streams, different mixing strategies can be applied.

3.8.1. Identical Window lengths, no Window Switching

As shown in section 3.2, two audio streams with identical windows can be simply mixed by adding the scaled transform coefficients block-wise.

3.8.2. Different Window Lengths, no Window Switching

One can either decide to interpolate the stream with longer blocks to get a stream with block lengths equal to the stream with shorter blocks, or decimate the other stream to the longer block lengths, or change the window lengths of both streams to an intermediate length.

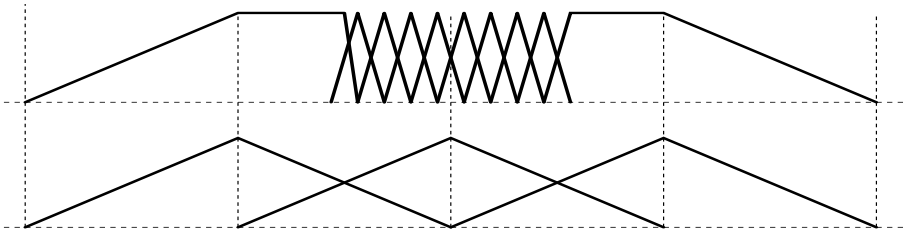


Figure 3.6.: Example of the built in synchronicity of AAC audio streams with short blocks restricted to groups of eight.

3.8.3. Window Switching on one Stream

The simplest way in this case would be to change both streams to the minimum window length in both streams, or if the fixed window length is smaller or equal to the short switched window, adapt the fixed window length stream to match the switched stream.

3.8.4. Window Switching on both Streams

When both streams consist of arbitrary sequences of long and short blocks, it only makes sense to change both streams to the minimum window lengths. But if we guarantee that both stream's windows sequences fullfill certain constraints, an other strategy is possible. One example of such a constraint stream is a MPEG 2 Advanced Audio Coding stream. Fig. 3.6 shows that short windows can only appear in groups of eight. This guarantees that for two streams the long windows are always exactly aligned. For the case that a long window in one stream meets a short window sequence in the other stream, the mixing algorithm has to decide if it either decimates the short sequence into a long window or interpolates a short sequence from the long window and then mixes the streams.

Such streams with always aligned long frames are provided if the short frames only appear in sequences containing $\frac{M_l}{M_s}$ short frames.

4. Implementing a Vorbis File Mixer in MATLAB

The examination of mixing MDCT coded streams in the previous chapter was carried out with uncoded data. To get insight in the behavior of the developed scheme for coded material , a mixer for Ogg Vorbis files was programmed in MATLAB.

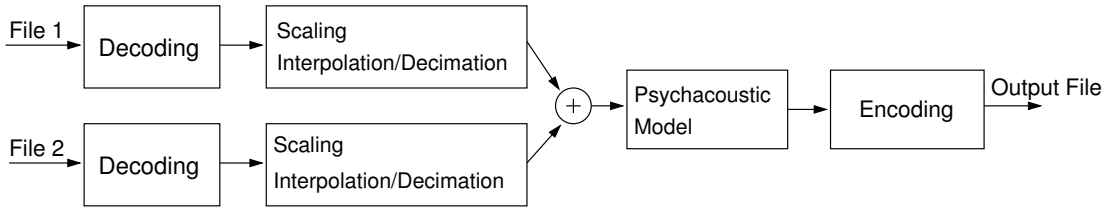


Figure 4.1.: Block diagram for the MATLAB Vorbis file mixer.

Figure 4.1 shows a block diagram of the Vorbis file mixer.

4.1. File Mixer Parts

4.1.1. Decoder Part

The decoder part was programmed according to the Vorbis I stream specifications [33]. Unfortunately this document contains some errors prohibiting correct decoding, so that for some parts of the decoder the C source code of the Vorbis libraries was taken as reference. The implementation follows the specifications very straightforward with only a view optimizations. Some aspects of the specifications where ignored, e.g. the granule positions at the start and end of the streams, indicating the number of samples padded or truncated.

The steps of the high-level decode process are:

1. Decode Setup

    a) Decode the identification header, which contains substantial information about the stream, like sample rate, bit rate, number of channels.

    b) Decode the comment header, it contains user tags, e.g. for title, album or artist.

    c) Decode the Setup header, containing the decode setup information. One specialty of Ogg Vorbis is the fact that there are no fixed codebooks for the

lossless compression but the inclusion of the codebooks needed for decoding within the setup header.

2. Audio block decoding

   a) Decode packet type
   b) Decode mode number
   c) Decode window shape
   d) Decode floor
   e) Decode residues into residue vectors (this part of the Vorbis I specifications contain some serious errors)
   f) Inverse channel coupling of residue vectors
   g) Generate floor curve from decoded floor data
   h) Compute dot product of floor and residue, producing the MDCT spectrum vector
   i) Inverse monolithic transform of the MDCT spectrum vector.
   j) Overlap/add left hand output of transform with right-hand output of previous frame
   k) Store right hand-data from transform of current frame for future lapping
   l) If not first frame, return result of overlap/add as audio result of current frame

For the file mixer, only steps a) to h) of the audio block decoding where computed, saving the MDCT and floor vectors and additional information about the block as well as general informations about the stream for later processing.

4.1.2. Scaling and Interpolation/Decimation

Scaling is carried out by simply multiplying the MDCT spectrum with the scaling factor, and when a window length adjustment is needed, it is followed by the interpolation/decimation with specified parameters for the coefficient threshold and inclusion of boundary blocks.

4.1.3. Addition

The two block sequences' MDCT spectra are simply added, and the additional block information adjusted.

4.1.4. Psychoacoustic model

For the psychoacoustic model the noise masking and absolute threshold of hearing application blocks were taken directly from the Ogg Vorbis reference encoder implementation. The tone masking section was omitted, due to the reason that it would have needed FFT

spectra to work on. A examination of the masking curves produced by the reference encoder implementation showed that although a reasonable part of the processing power is spent on tone masking, tone masking components hardly show up in the resulting masking curve, which are to a great extend only defined by the noise masking and an ATH curve which is floated according to the loudest spectrum line. Unfortunately the main part of the noise masking algorithm in Ogg Vorbis, the function `bark_noise_hybridmp` is completely undocumented and an attempt to get more information via the Ogg Vorbis development mailing list proved fruitless, since it is neigh to impossible to get answers for questions regarding deeper insight into the encoding process. So the precise algorithm of the noise masking process could only be guessed by me.

4.1.5. Encoding

For the encoding, the reference Vorbis library was adapted to accommodate for encoding of already transformed and psychoacoustically processed audio blocks and is accessed via an MATLAB MEX C access function.

4.1.6. User Interface

Figure 4.2 shows the user interface for the Vorbis file mixer. It is roughly divided into two parts, the input files section on the left side and the output file and settings section on the right side.

4.1.6.1. Input file section

The input file controls for both possible input files are identical. A input file can be chosen via a "File open..." dialog accessed by the "Browse..." button or by directly typing the file name into the edit box. When the file is a valid Ogg Vorbis file, the basic information about the contained audio stream are shown below the audio file. With the slider on the right side of the input file boxes, the wanted level adjustment can be set.

4.1.6.2. Output file and settings section

The output file can be chosen similarly like the input files.

The encoding setting section sets the wanted quality level and blocksize for the output file.

The Interpolation/Decimation setting section sets the wanted threshold for the calculation and whether boundary blocks should be included or not.

4.1.6.3. Mixing

After at least one input and the output file have been chosen, the "Mix->" button gets enabled, and the file(s) can be processed by clicking it.

At the moment the file mixer allows only for input and output files with a fixed block length.

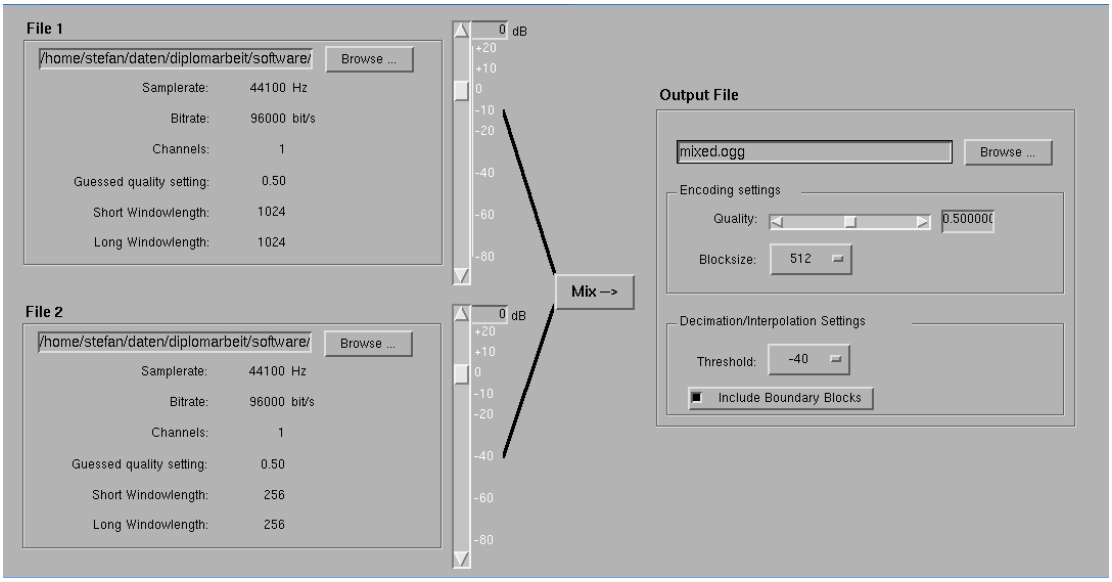


Figure 4.2.: GUI for MATLAB Vorbis file mixer.

4.2. Evaluation

To evaluate the performance of the vorbis file mixer, Ogg Vorbis files with fixed block lengths were generated. For this, the reference Vorbis libraries were modified, since normally Ogg Vorbis encodes only window switched streams. This modifications are highly unoptimized, resulting in higher bitrates than the reference encoder.

For the evaluation only interpolation and decimation of single Vorbis files without mixing and scaling were carried out, since the processing time within MATLAB was very high (about 11 Minutes for mixing two files with window changes and scaling on both files on a AMD Athlon 64 3000+).

The settings for decimation and interpolation were kept at -40dB for the threshold and the inclusion of boundary blocks as a tradeoff chosen between complexity and signal degradation.

Settings for the quality were kept constant over one set, i.e. the quality settings for the input file and the output file were equal.

4.2.1. Vorbis Interpolation

The results for interpolating over all test signals are shown in tables 4.1 to 4.2 and figures 4.3 to 4.4 for different quality settings.

The ODGs of the direct interpolation are nearly the same as the ODGs of files reencoded with a complete decoding/encoding cycle, while the resulting bit rates are about 15% higher. This would come from the simpler psychacoustic model implemented for direct interpolation/decimation, generating lower floor curves due to the absence of tone masking and the lower floating of the absolute threshold of hearing. A adjustment of the

psychoacoustic model to get comparable bit rates would most probably lower the ODGs for direct interpolation a bit.

As a second result it can be seen that every reencoding cycle, independent of the used scheme, adds further signal degradation, as the ODGs for direct encoded reference files for the respective target block lengths show in the result tables.

Thirdly the results also show that longer block lengths provide higher coding gain and lower bitrates.

Table 4.1.: Results for Vorbis interpolation with quality 0.2, threshold -40dB, boundary blocks included

from	2048				1024			512		256
to	1023	511	255	127	511	255	127	255	127	127
ODG	-1.13	-0.85	-0.85	-1.00	-0.74	-0.77	-0.95	-0.82	-1.02	-1.05
ref. ODG[1]	-0.41	-0.50	-0.53	-0.83	-0.50	-0.53	-0.83	-0.53	-0.83	-0.83
ref. ODG[2]	-1.03	-1.05	-1.00	-1.17	-0.97	-0.94	-1.11	-0.98	-1.16	-1.20
bitrate [kbit/s]	95	108	118	158	107	118	157	118	157	157
ref. bitrate [kbit/s][1]	75	83	91	126	83	91	126	91	126	126
ref. bitrate [kbit/s][2]	75	83	93	128	84	93	128	93	128	128

[1] for direct encoded file [2] for file encoded with inverse/forward coding

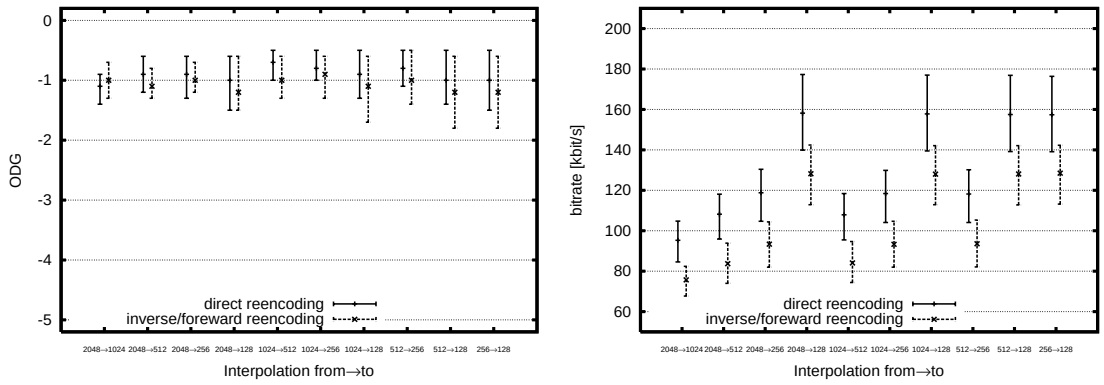


Figure 4.3.: Results for Vorbis interpolation with settings quality 0.2, threshold -40dB, include boundary blocks.

4.2.2. Vorbis Decimation

The results for decimating over all test signals are shown in tables 4.3 to 4.4 and figures 4.5 to 4.6 for different quality settings. The results reflect those for interpolating quite as expected, the only difference being the more serious signal degradation for the cases 128 to 2048 and 256 to 2048.

Table 4.2.: Results for Vorbis interpolation with quality 5.0, threshold -40dB, boundary blocks included

from	2048				1024			512		256
to	1023	511	255	127	511	255	127	255	127	127
ODG	-0.60	-0.36	-0.41	-0.49	-0.28	-0.34	-0.43	-0.41	-0.51	-0.54
ref. ODG[1]	-0.11	-0.16	-0.26	-0.39	-0.16	-0.26	-0.39	-0.26	-0.39	-0.39
ref. ODG[2]	-0.43	-0.49	-0.52	-0.59	-0.44	-0.48	-0.54	-0.53	-0.60	-0.64
bitrate [kbit/s]	113	126	141	178	126	141	178	141	177	178
ref. bitrate [kbit/s][1]	94	99	114	145	99	114	145	114	145	145
ref. bitrate [kbit/s][2]	94	99	114	146	100	114	146	116	147	147

[1] for direct encoded file [2] for file encoded with inverse/forward coding

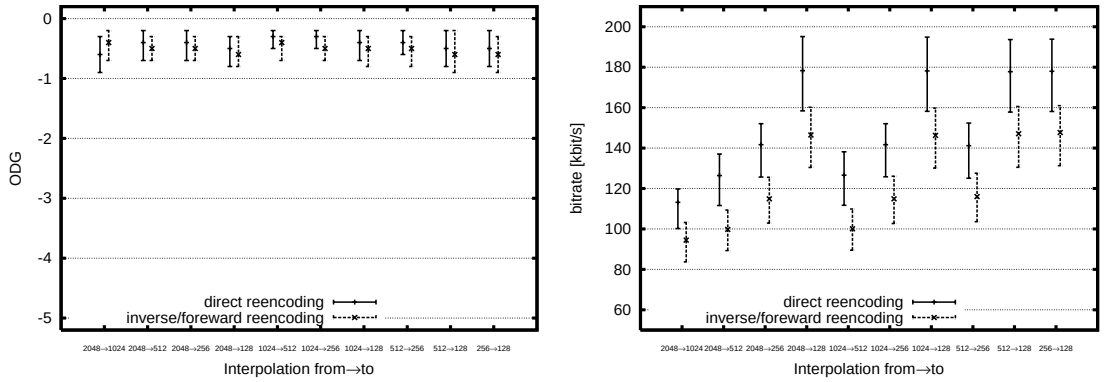


Figure 4.4.: Results for Vorbis interpolation with settings quality 0.5, threshold -40dB, include boundary blocks.

Table 4.3.: Results for Vorbis decimation with quality 0.2, threshold -40dB, boundary blocks included.

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-1.22	-1.25	-1.32	-2.50	-0.92	-0.97	-2.54	-0.92	-0.96	-0.91
ref. ODG[1]	-0.53	-0.50	-0.41	-0.53	-0.50	-0.41	-0.53	-0.41	-0.53	-0.53
ref. ODG[2]	-1.30	-1.38	-1.36	-1.44	-1.08	-1.06	-1.14	-1.07	-1.15	-1.11
bitrate [kbit/s]	120	109	97	91	109	97	91	96	91	90
ref. bitrate [kbit/s][1]	92	83	76	72	83	76	72	76	72	72
ref. bitrate [kbit/s][2]	94	84	76	73	84	76	73	75	72	72

[1] for direct encoded file [2] for file encoded with inverse/forward coding

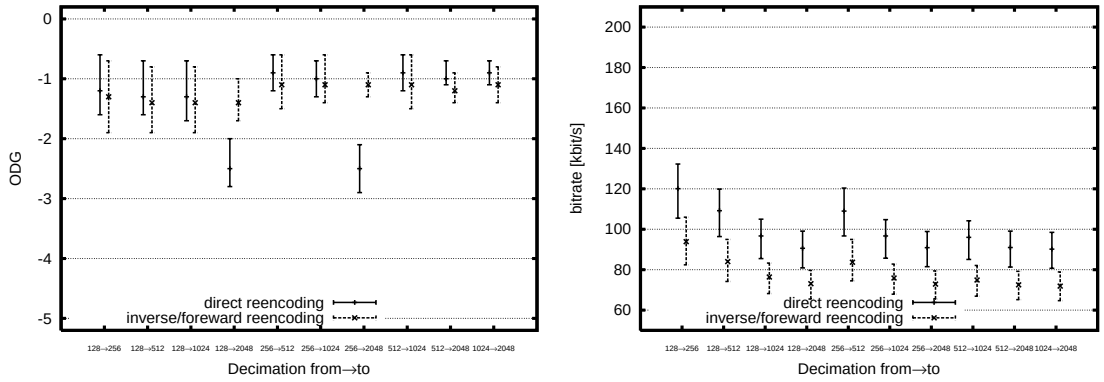


Figure 4.5.: Results for Vorbis decimation with settings quality 0.2, threshold -40dB, include boundary blocks.

Table 4.4.: Results for Vorbis decimation with quality 0.5, threshold -40dB, boundary blocks included.

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.63	-0.64	-0.66	-1.64	-0.51	-0.52	-1.79	-0.37	-0.40	-0.35
ref. ODG[1]	-0.26	-0.16	-0.11	-0.20	-0.16	-0.11	-0.20	-0.11	-0.20	-0.20
ref. ODG[2]	-0.66	-0.66	-0.61	-0.69	-0.54	-0.52	-0.58	-0.47	-0.53	-0.47
bitrate [kbit/s]	141	125	112	105	126	112	105	112	106	107
ref. bitrate [kbit/s][1]	114	99	94	89	99	94	89	94	89	89
ref. bitrate [kbit/s][2]	115	99	93	88	100	93	88	93	88	89

[1] for direct encoded file [2] for file encoded with inverse/forward coding

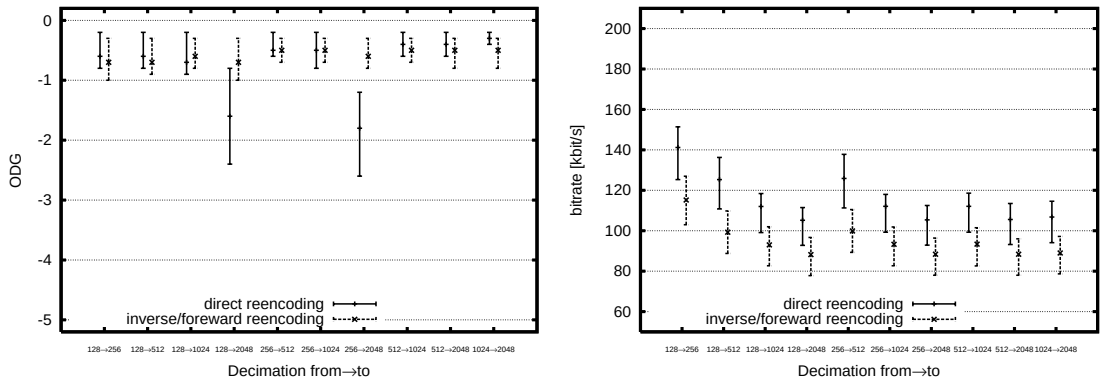


Figure 4.6.: Results for Vorbis decimation with settings quality 0.5, threshold -40dB, include boundary blocks.

5. Conclusions

The target of this thesis was to investigate possibilities to mix perceptual coded audio streams based on the MDCT. For this, a algorithm for changing the window lengths of the streams was developed. The direct MDCT interpolation/decimation proved to be equal to a inverse/forward transformation, but was computationally more complex than existing fast algorithms for the MDCT. To reduce the complexity small coefficients and/or source blocks contributing little were ignored in the direct window length change, resulting in a signal degradation.

Staying in the MDCT domain makes the psychoacoustic analysis for reencoding more difficult, since a reliable tonal masking estimation is more complicated on MDCT spectra.

As a conclusion it has to be admitted, although a direct window length change within the MDCT domain is possible, it yields no practicable advantages over a inverse/direct transform, especially since to get in the vicinity to fast MDCT transform algorithms regarding computational complexity, signal degradation has to be accepted, and a favor in latency is only achieved in one special case (see below). Further diminution on the topic of computational complexity arises from the fact that the MDCT transform itself normally only takes up a relatively small amount of processing time at the encoding process in perceptual audio coders (10% in the reference Vorbis encoder implementation, where the psychoacoustic model alone uses about 40% of the processing time).

On the other hand, due to the validity of superposition for MDCTs with equal block lengths, when mixing two streams together it is only necessary to adjust only the parts of one stream where window lengths don't match in the best case, while all other parts can be processed within the MDCT domain.

5.1. Low Latency Applications

The latency examination showed that a small favor in terms of latency can only be realized when no window length change has to be computed for the streams. But since for a low latency application it would anyway be wise to choose the shortest possible block length, and to do without window length switching, since that would require extensive buffering of the input signal for the switching decision (up to 2 seconds in the actual Vorbis encoder), this would certainly be the case.

5.2. Low Bandwidth Applications

For low bandwidth and low complexity applications (e.g. mixing together several surveillance audio channels) using Ogg Vorbis streams one could use a very simple approach

and use the same fixed window length on all channels and omit the psychoacoustic model at the reencoding stage by simply scaling and adding the floors from the partly decoded streams just like the MDCT spectra.

5.3. Future Work

- Extend the developed algorithms to accommodate for streams with window switching.
- Incorporate tonal masking into the psychoacoustic model.
- Extend the the processing possibilities in the MDCT domain beyond simple scaling, e.g. filtering.
- Implement the scheme into a realtime mixer working at realworld audio streams.

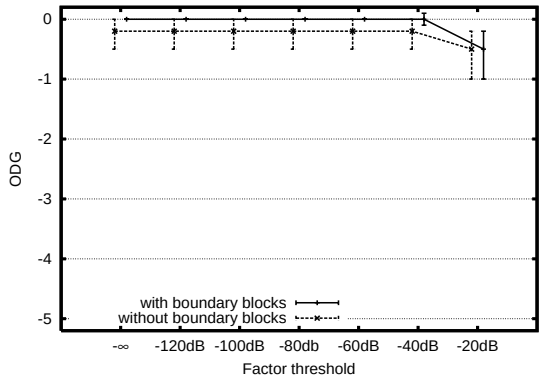
A. Test results for all test files

A.1. Test files

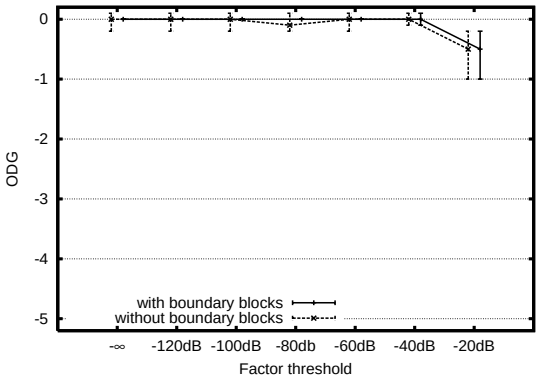
Eight different test signal were used. Each signal is of a length of approximately 10 seconds. All test signals were downmixed from stereo sources to mono.

Filename	Description
`monoBeethovenRef.wav`	Piano Concert, big classical orchestra piece
`monoGouldRef.wav`	Bach Goldberg Variations, single piano
`monoStrawinskiRef.wav`	Le Sacre du Printemps, Igor Strawinsky, again big orchestra
`monoTrovesiRef.wav`	only percussion, kettledrum
`monoProdigyRef.wav`	electronic pop piece, lots of noise
`monoMalespeechRef.wav`	two male speakers talking
`monoPinknoiseRef.wav`	pink noise
`monoWhitenoiseRef.wav`	white noise

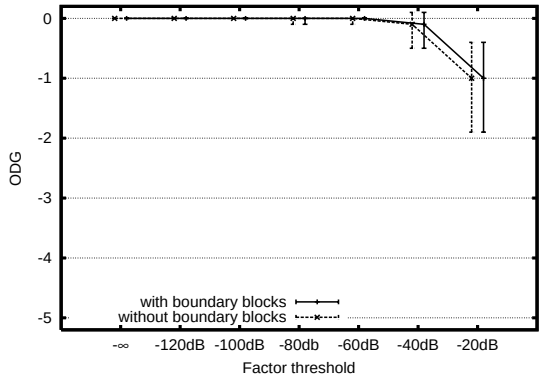
Table A.1.: Test signals



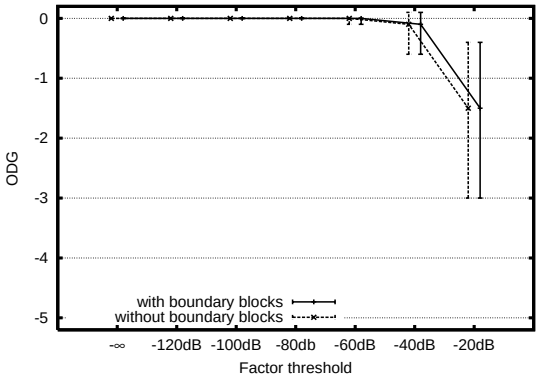
(a) Interpolating from 2048 to 1024



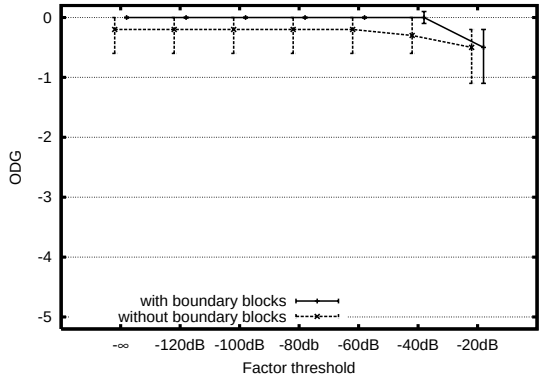
(b) Interpolating from 2048 to 512



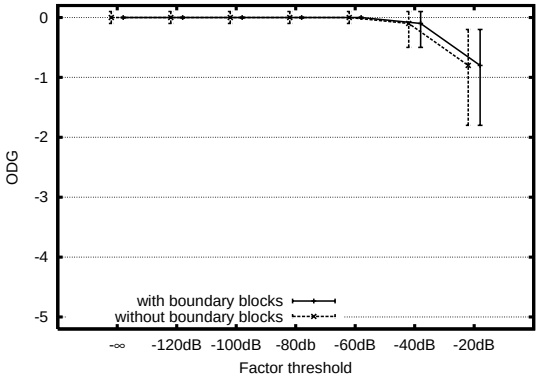
(c) Interpolating from 2048 to 256



(d) Interpolating from 2048 to 128

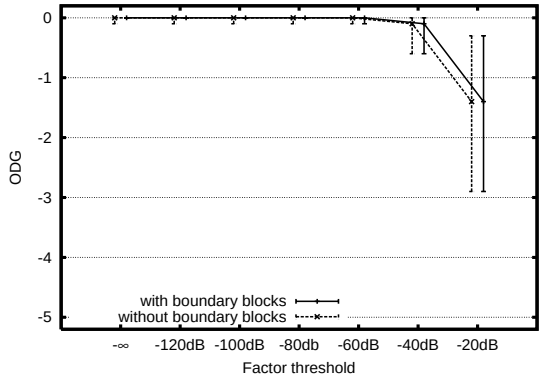


(e) Interpolating from 1024 to 512

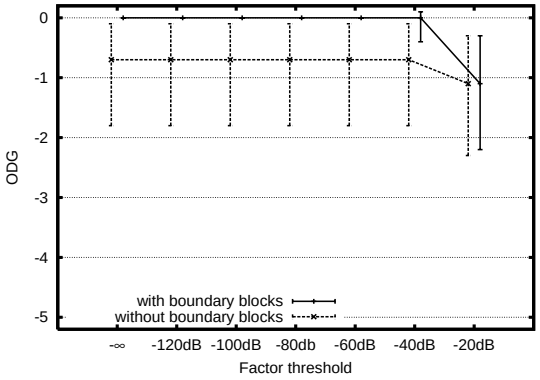


(f) Interpolating from 1024 to 256

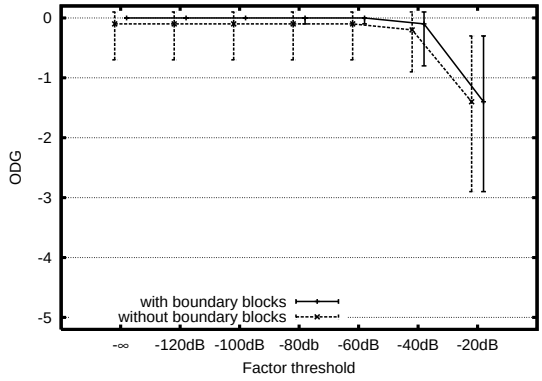
Figure A.1.: Test results for MDCT interpolation



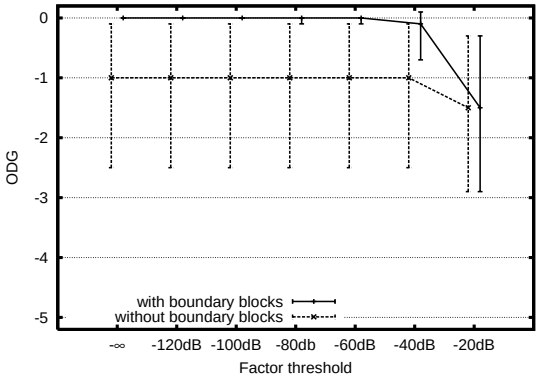
(g) Interpolating from 1024 to 128



(h) Interpolating from 512 to 256

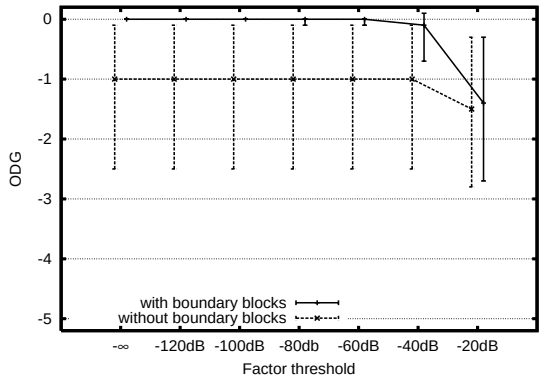


(i) Interpolating from 512 to 128

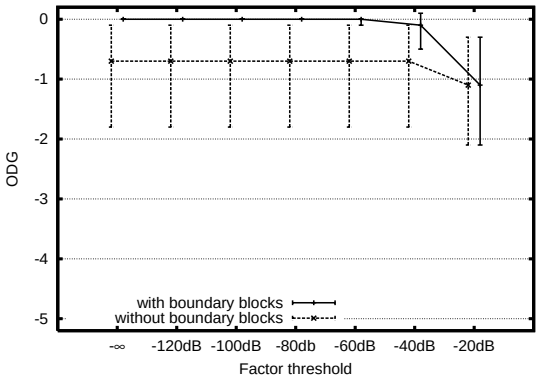


(j) Interpolating from 256 to 128

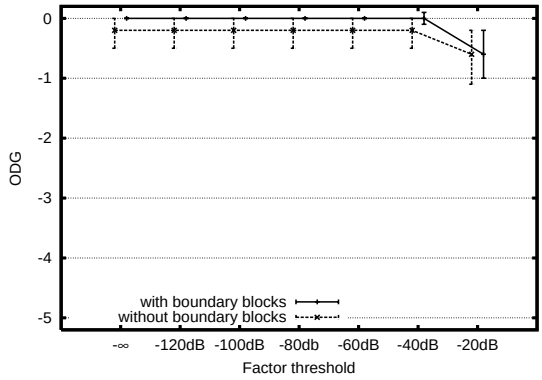
Figure A.1.: Test results for MDCT interpolation (cont.).



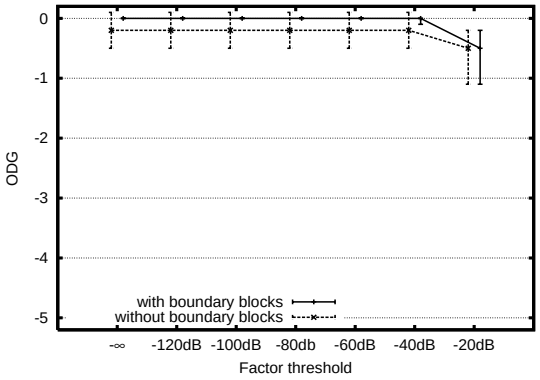
(a) Decimating from 128 to 256



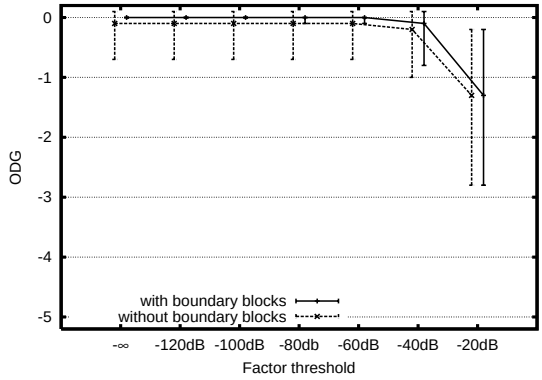
(b) Decimating from 128 to 512



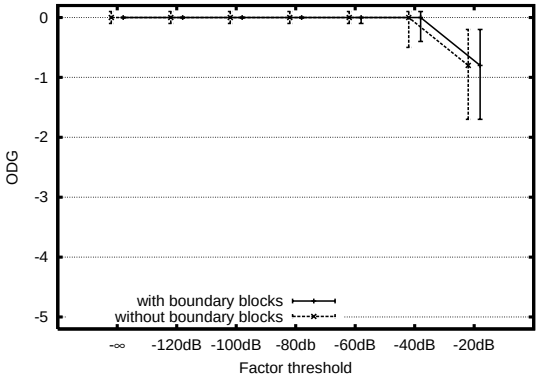
(c) Decimating from 128 to 1024



(d) Decimating from 128 to 2048

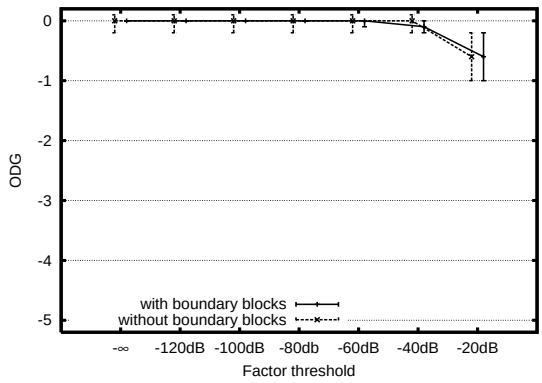


(e) Decimating from 256 to 512

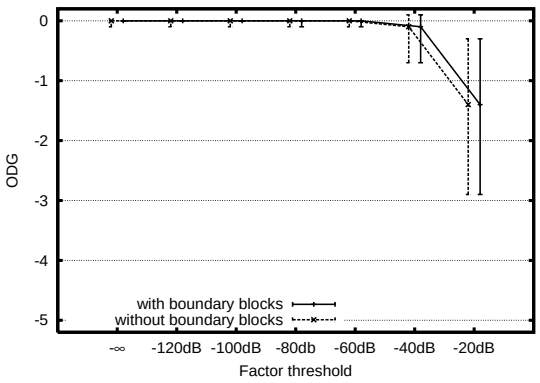


(f) Decimating from 256 to 1024

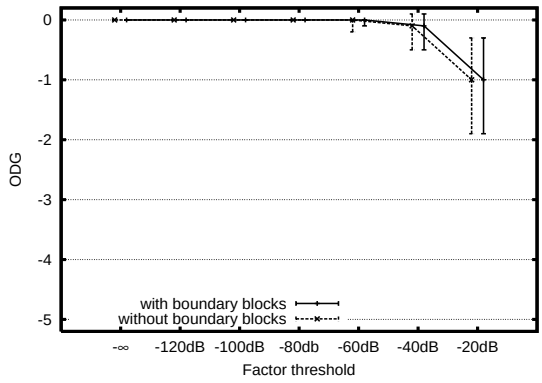
Figure A.2.: Test results for MDCT decimation.



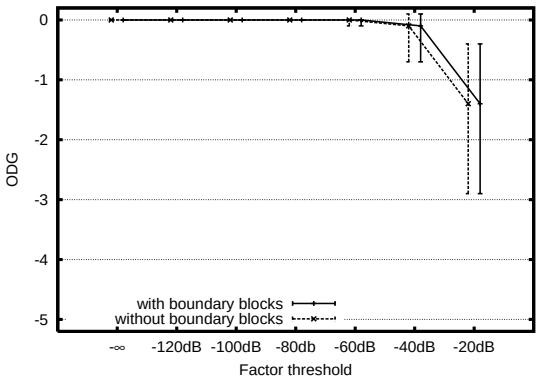
(g) Decimating from 256 to 2048



(h) Decimating from 512 to 1024



(i) Decimating from 512 to 2048



(j) Decimating from 1024 to 2048

Figure A.2.: Test results for MDCT decimation (cont.).

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-1.26	-0.79	-0.79	-0.84	-0.85	-0.83	-0.86	-0.92	-0.93	-1.07
ref. ODG[1]	-0.63	-0.70	-0.76	-0.99	-0.70	-0.76	-0.99	-0.76	-0.99	-0.99
ref. ODG[2]	-1.14	-1.07	-1.05	-1.15	-1.15	-1.08	-1.23	-1.15	-1.26	-1.41
bitrate [kbit/s]	97	108	118	151	107	117	150	117	150	150
ref. bitrate [kbit/s][1]	76	82	89	120	82	89	120	89	120	120
ref. bitrate [kbit/s][2]	76	83	92	123	84	92	123	92	123	123

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(a) Test file `monoBeethovenRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-1.26	-0.55	-0.59	-0.72	-0.67	-0.66	-0.77	-0.74	-0.85	-0.85
ref. ODG[1]	-0.27	-0.43	-0.53	-0.75	-0.43	-0.53	-0.75	-0.53	-0.75	-0.75
ref. ODG[2]	-0.71	-0.77	-0.76	-1.01	-0.86	-0.88	-1.03	-0.92	-1.08	-1.13
bitrate [kbit/s]	87	99	108	149	99	108	149	107	149	149
ref. bitrate [kbit/s][1]	70	78	86	123	78	86	123	86	123	123
ref. bitrate [kbit/s][2]	70	78	86	123	78	86	123	86	123	123

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(b) Test file `monoGouldRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-1.03	-0.93	-0.90	-1.15	-0.98	-0.97	-1.22	-0.99	-1.33	-1.27
ref. ODG[1]	-0.56	-0.67	-0.59	-0.95	-0.67	-0.59	-0.95	-0.59	-0.95	-0.95
ref. ODG[2]	-1.15	-1.15	-1.09	-1.33	-1.26	-1.14	-1.37	-1.22	-1.47	-1.42
bitrate [kbit/s]	104	117	128	177	116	128	176	127	176	176
ref. bitrate [kbit/s][1]	81	87	98	141	87	98	141	98	141	141
ref. bitrate [kbit/s][2]	81	88	100	142	88	99	142	99	142	142

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(c) Test file `monoGouldRef.wav`

Table A.2.: Vorbis Interpolation for quality 0.2, threshold -40dB, boundary blocks included

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-1.35	-1.24	-1.30	-1.54	-0.49	-0.66	-1.03	-0.62	-1.16	-1.18
ref. ODG[1]	-0.24	-0.18	-0.25	-0.71	-0.18	-0.25	-0.71	-0.25	-0.71	-0.71
ref. ODG[2]	-1.32	-1.30	-1.19	-1.37	-0.59	-0.58	-0.80	-0.54	-0.83	-0.85
bitrate [kbit/s]	101	118	130	169	118	129	168	130	168	168
ref. bitrate [kbit/s][1]	83	95	104	136	95	104	136	104	136	136
ref. bitrate [kbit/s][2]	82	93	104	136	94	104	136	105	137	138

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(d) Test file `monoTrovesiRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.92	-0.74	-0.65	-0.55	-0.55	-0.51	-0.46	-0.54	-0.49	-0.45
ref. ODG[1]	-0.33	-0.39	-0.29	-0.37	-0.39	-0.29	-0.37	-0.29	-0.37	-0.37
ref. ODG[2]	-0.90	-0.86	-0.70	-0.61	-0.73	-0.61	-0.57	-0.64	-0.56	-0.55
bitrate [kbit/s]	95	109	122	161	109	121	160	122	160	160
ref. bitrate [kbit/s][1]	74	81	90	125	81	90	125	90	125	125
ref. bitrate [kbit/s][2]	75	83	95	129	83	94	129	95	129	130

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(e) Test file `monoTrovesiRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.95	-0.87	-0.88	-1.23	-0.88	-0.96	-1.33	-1.10	-1.38	-1.46
ref. ODG[1]	-0.46	-0.64	-0.78	-1.21	-0.64	-0.78	-1.21	-0.78	-1.21	-1.21
ref. ODG[2]	-0.99	-1.16	-1.19	-1.53	-1.24	-1.34	-1.68	-1.38	-1.78	-1.81
bitrate [kbit/s]	84	95	104	139	95	104	139	104	139	139
ref. bitrate [kbit/s][1]	67	73	80	111	73	80	111	80	111	111
ref. bitrate [kbit/s][2]	67	74	82	112	74	81	112	82	112	113

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(f) Test file `monoTrovesiRef.wav`

Table A.2.: Vorbis Interpolation for quality 0.2, threshold -40dB, boundary blocks included (cont.).

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.61	-0.26	-0.35	-0.34	-0.30	-0.35	-0.35	-0.44	-0.41	-0.50
ref. ODG[1]	-0.11	-0.21	-0.36	-0.43	-0.21	-0.36	-0.43	-0.36	-0.43	-0.43
ref. ODG[2]	-0.33	-0.39	-0.45	-0.49	-0.40	-0.47	-0.49	-0.54	-0.55	-0.63
bitrate [kbit/s]	119	130	143	173	130	143	173	143	172	172
ref. bitrate [kbit/s][1]	100	101	114	140	101	114	140	114	140	140
ref. bitrate [kbit/s][2]	100	102	115	142	103	115	141	116	143	143

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(a) Test file `monoBeethovenRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.91	-0.18	-0.29	-0.37	-0.25	-0.35	-0.41	-0.43	-0.49	-0.54
ref. ODG[1]	-0.01	-0.07	-0.27	-0.40	-0.07	-0.27	-0.40	-0.27	-0.40	-0.40
ref. ODG[2]	-0.24	-0.30	-0.43	-0.54	-0.40	-0.51	-0.61	-0.55	-0.62	-0.74
bitrate [kbit/s]	105	116	130	167	115	130	167	130	166	166
ref. bitrate [kbit/s][1]	87	94	108	138	94	108	138	108	138	138
ref. bitrate [kbit/s][2]	87	94	109	140	94	108	140	109	140	140

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(b) Test file `monoGouldRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.51	-0.44	-0.46	-0.52	-0.47	-0.48	-0.51	-0.57	-0.64	-0.62
ref. ODG[1]	-0.27	-0.32	-0.35	-0.40	-0.32	-0.35	-0.40	-0.35	-0.40	-0.40
ref. ODG[2]	-0.57	-0.63	-0.65	-0.67	-0.66	-0.67	-0.70	-0.72	-0.79	-0.78
bitrate [kbit/s]	119	133	152	195	133	152	194	150	193	193
ref. bitrate [kbit/s][1]	97	102	120	158	102	120	158	120	158	158
ref. bitrate [kbit/s][2]	97	103	121	160	103	120	159	121	160	161

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(c) Test file `monoStrawinskiRef.wav`

Table A.3.: Vorbis Interpolation for quality 0.5, threshold -40dB, boundary blocks included.

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.78	-0.69	-0.68	-0.85	-0.18	-0.23	-0.47	-0.24	-0.53	-0.51
ref. ODG[1]	-0.08	-0.08	-0.13	-0.40	-0.08	-0.13	-0.40	-0.13	-0.40	-0.40
ref. ODG[2]	-0.70	-0.74	-0.71	-0.79	-0.32	-0.33	-0.43	-0.34	-0.47	-0.51
bitrate [kbit/s]	119	137	151	191	138	151	191	152	191	192
ref. bitrate [kbit/s][1]	103	109	126	157	109	126	157	126	157	157
ref. bitrate [kbit/s][2]	103	109	125	157	109	126	158	127	159	160

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(d) Test file `monoTrovesiRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.35	-0.25	-0.25	-0.27	-0.16	-0.20	-0.21	-0.20	-0.22	-0.19
ref. ODG[1]	-0.09	-0.09	-0.09	-0.11	-0.09	-0.09	-0.11	-0.09	-0.11	-0.11
ref. ODG[2]	-0.34	-0.33	-0.30	-0.30	-0.28	-0.26	-0.26	-0.26	-0.25	-0.26
bitrate [kbit/s]	115	129	146	184	130	146	184	146	183	184
ref. bitrate [kbit/s][1]	93	98	114	145	98	114	145	114	145	145
ref. bitrate [kbit/s][2]	93	99	115	147	99	115	147	117	148	149

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(e) Test file `monoProdigyRef.wav`

from	2048				1024			512		256
to	1024	512	256	128	512	256	128	256	128	128
ODG	-0.42	-0.34	-0.42	-0.62	-0.35	-0.45	-0.65	-0.57	-0.79	-0.85
ref. ODG[1]	-0.12	-0.21	-0.35	-0.59	-0.21	-0.35	-0.59	-0.35	-0.59	-0.59
ref. ODG[2]	-0.43	-0.56	-0.60	-0.74	-0.57	-0.64	-0.78	-0.75	-0.91	-0.91
bitrate [kbit/s]	100	111	125	158	111	125	158	125	157	158
ref. bitrate [kbit/s][1]	83	89	102	129	89	102	129	102	129	129
ref. bitrate [kbit/s][2]	83	89	102	130	89	102	130	103	130	131

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(f) Test file `monoMalespeechRef.wav`

Table A.3.: Vorbis Interpolation for quality 0.5, threshold -40dB, boundary blocks included (cont.).

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-1.38	-1.42	-1.46	-2.75	-1.11	-1.15	-2.85	-1.21	-1.08	-1.10
ref. ODG[1]	-0.76	-0.70	-0.63	-0.47	-0.70	-0.63	-0.47	-0.63	-0.47	-0.47
ref. ODG[2]	-1.58	-1.67	-1.62	-1.60	-1.43	-1.41	-1.29	-1.50	-1.39	-1.45
bitrate [kbit/s]	119	109	98	93	109	98	94	98	95	94
ref. bitrate [kbit/s][1]	90	82	76	74	82	76	74	76	74	74
ref. bitrate [kbit/s][2]	93	83	77	75	83	77	75	76	74	74

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(a) Test file `monoBeethovenRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-1.01	-1.06	-1.10	-2.64	-0.86	-0.89	-2.69	-0.81	-0.90	-0.99
ref. ODG[1]	-0.53	-0.43	-0.27	-0.17	-0.43	-0.27	-0.17	-0.27	-0.17	-0.17
ref. ODG[2]	-1.19	-1.21	-1.23	-1.17	-1.02	-0.99	-0.98	-0.97	-0.89	-0.85
bitrate [kbit/s]	110	102	91	84	101	90	84	89	84	83
ref. bitrate [kbit/s][1]	86	78	71	66	78	71	66	71	66	66
ref. bitrate [kbit/s][2]	87	78	72	67	78	71	67	70	66	65

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(b) Test file `monoGouldRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-1.48	-1.49	-1.62	-2.65	-1.04	-1.19	-2.38	-1.10	-1.08	-0.98
ref. ODG[1]	-0.59	-0.67	-0.56	-0.50	-0.67	-0.56	-0.50	-0.56	-0.50	-0.50
ref. ODG[2]	-1.45	-1.65	-1.60	-1.54	-1.30	-1.23	-1.19	-1.32	-1.22	-1.19
bitrate [kbit/s]	129	117	105	99	117	105	99	104	99	98
ref. bitrate [kbit/s][1]	99	88	82	77	88	82	77	82	77	77
ref. bitrate [kbit/s][2]	100	89	82	78	88	81	77	80	77	77

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(c) Test file `monoStrawinskiRef.wav`

Table A.4.: Vorbis Decimation for quality 0.2, threshold -40dB, boundary blocks included.

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-1.26	-1.24	-1.28	-2.19	-0.70	-0.66	-2.08	-0.56	-0.93	-0.86
ref. ODG[1]	-0.25	-0.18	-0.24	-1.07	-0.18	-0.24	-1.07	-0.24	-1.07	-1.07
ref. ODG[2]	-0.96	-1.03	-1.04	-1.63	-0.59	-0.62	-1.31	-0.61	-1.36	-1.33
bitrate [kbit/s]	132	120	104	95	120	104	96	104	96	95
ref. bitrate [kbit/s][1]	105	95	84	79	95	84	79	84	79	79
ref. bitrate [kbit/s][2]	106	95	83	80	95	83	79	82	79	79

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(d) Test file `monoTrovesiRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.59	-0.68	-0.74	-1.98	-0.59	-0.65	-2.43	-0.68	-0.67	-0.68
ref. ODG[1]	-0.29	-0.39	-0.33	-0.54	-0.39	-0.33	-0.54	-0.33	-0.54	-0.54
ref. ODG[2]	-0.68	-0.81	-0.84	-0.97	-0.71	-0.76	-0.86	-0.78	-0.95	-0.90
bitrate [kbit/s]	124	111	97	91	110	97	91	96	91	90
ref. bitrate [kbit/s][1]	91	82	75	72	82	75	72	75	72	72
ref. bitrate [kbit/s][2]	96	85	77	74	84	76	73	75	72	72

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(e) Test file `monoProdigyRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-1.62	-1.62	-1.74	-2.82	-1.22	-1.26	-2.78	-1.13	-1.11	-0.84
ref. ODG[1]	-0.78	-0.64	-0.46	-0.40	-0.64	-0.46	-0.40	-0.46	-0.40	-0.40
ref. ODG[2]	-1.91	-1.91	-1.86	-1.71	-1.46	-1.38	-1.22	-1.27	-1.11	-0.93
bitrate [kbit/s]	106	96	86	81	97	86	82	85	81	81
ref. bitrate [kbit/s][1]	81	74	68	65	74	68	65	68	65	65
ref. bitrate [kbit/s][2]	82	74	68	65	74	68	66	67	65	65

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(f) Test file `monoMalespeechRef.wav`

Table A.4.: Vorbis Interpolation for quality 0.2, threshold -40dB, boundary blocks included (cont.).

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.60	-0.61	-0.60	-1.68	-0.53	-0.53	-1.95	-0.41	-0.34	-0.30
ref. ODG[1]	-0.36	-0.21	-0.11	-0.08	-0.21	-0.11	-0.08	-0.11	-0.08	-0.08
ref. ODG[2]	-0.70	-0.69	-0.60	-0.62	-0.60	-0.55	-0.55	-0.50	-0.49	-0.41
bitrate [kbit/s]	142	128	117	111	129	118	113	119	114	115
ref. bitrate [kbit/s][1]	114	102	100	97	102	100	97	100	97	97
ref. bitrate [kbit/s][2]	115	100	97	93	102	99	95	100	95	96

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(a) Test file `monoBeethovenRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.67	-0.64	-0.65	-2.35	-0.63	-0.57	-2.59	-0.31	-0.46	-0.41
ref. ODG[1]	-0.27	-0.07	-0.01	0.06	-0.07	-0.01	0.06	-0.01	0.06	0.06
ref. ODG[2]	-0.74	-0.69	-0.65	-0.62	-0.59	-0.55	-0.49	-0.42	-0.34	-0.29
bitrate [kbit/s]	130	115	105	98	116	105	98	105	98	98
ref. bitrate [kbit/s][1]	108	94	88	82	94	88	82	88	82	82
ref. bitrate [kbit/s][2]	109	94	86	81	94	87	81	87	81	81

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(b) Test file `monoGouldRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.74	-0.80	-0.91	-1.91	-0.63	-0.76	-1.88	-0.59	-0.56	-0.42
ref. ODG[1]	-0.35	-0.32	-0.27	-0.21	-0.32	-0.27	-0.21	-0.27	-0.21	-0.21
ref. ODG[2]	-0.74	-0.84	-0.76	-0.77	-0.72	-0.73	-0.69	-0.71	-0.68	-0.57
bitrate [kbit/s]	151	132	118	112	132	118	111	118	111	113
ref. bitrate [kbit/s][1]	120	103	97	92	103	97	92	97	92	92
ref. bitrate [kbit/s][2]	121	103	96	91	103	96	91	96	91	91

[1]for direct encoded file [2]for file encoded with inverse/forward coding

(c) Test file `monoStrawinskiRef.wav`

Table A.5.: Vorbis Decimation for quality 0.5, threshold -40dB, boundary blocks included.

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.78	-0.77	-0.73	-1.37	-0.49	-0.44	-1.24	-0.28	-0.45	-0.44
ref. ODG[1]	-0.13	-0.08	-0.08	-0.62	-0.08	-0.08	-0.62	-0.08	-0.62	-0.62
ref. ODG[2]	-0.53	-0.55	-0.52	-1.01	-0.36	-0.35	-0.81	-0.34	-0.80	-0.80
bitrate [kbit/s]	151	136	118	110	138	118	110	117	109	112
ref. bitrate [kbit/s][1]	126	110	103	98	110	103	98	103	98	98
ref. bitrate [kbit/s][2]	127	110	102	97	110	102	96	102	96	97

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(d) Test file `monoTrovesiRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.19	-0.18	-0.21	-0.81	-0.18	-0.20	-1.16	-0.17	-0.17	-0.19
ref. ODG[1]	-0.09	-0.09	-0.09	-0.21	-0.09	-0.09	-0.21	-0.09	-0.21	-0.21
ref. ODG[2]	-0.26	-0.27	-0.28	-0.34	-0.27	-0.29	-0.34	-0.29	-0.34	-0.32
bitrate [kbit/s]	147	130	115	108	130	115	108	114	108	109
ref. bitrate [kbit/s][1]	115	99	93	89	99	93	89	93	89	89
ref. bitrate [kbit/s][2]	117	100	94	90	100	94	90	93	89	89

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(e) Test file `monoProdigyRef.wav`

from	128				256			512		1024
to	256	512	1024	2048	512	1024	2048	1024	2048	2048
ODG	-0.82	-0.83	-0.85	-1.70	-0.60	-0.65	-1.95	-0.46	-0.41	-0.32
ref. ODG[1]	-0.35	-0.21	-0.12	-0.15	-0.21	-0.12	-0.15	-0.12	-0.15	-0.15
ref. ODG[2]	-0.96	-0.95	-0.85	-0.77	-0.73	-0.67	-0.58	-0.59	-0.51	-0.42
bitrate [kbit/s]	125	111	99	93	111	99	93	99	93	94
ref. bitrate [kbit/s][1]	102	89	84	79	89	84	79	84	79	79
ref. bitrate [kbit/s][2]	103	89	83	78	89	83	78	83	78	79

[1] for direct encoded file [2] for file encoded with inverse/forward coding

(f) Test file `monoMalespeechRef.wav`

Table A.5.: Vorbis Decimation for quality 0.5, threshold -40dB, boundary blocks included (cont.).

B. Ogg Vorbis license

Bibliography

[1] Ogg Vorbis Homepage http://www.vorbis.com.

[2] Ted Painter and Spanias Andreas. Perceptual Coding of Digital Audio. *Proceedings of the IEEE*, 88(4):451–512, April 2000.

[3] E. Zwicker and H. Fastl. *Psychoacoustics: Facts and Models*. Number 22 in Springer series in information sciences. Springer, 1999.

[4] ISO/IEC 11172-3 Information Technology – Coding of Moving Pictures and Associated audio for Digital Storage Media at up to about 1,5 Mbit/s – Part 3: Audio, 1993.

[5] Henrique S. Malvar. *Signal Processing with Lapped Transforms*. Artech House, 1992.

[6] Marina Bosi. Filter Banks in Perceptual Audio Coding. In von Recklinghausen [34], pages 125–136.

[7] John P. Princen and Alan Bernard Bradley. Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation. *IEEE Transactions on Speech and Audio Processing*, ASSP-34(5):1153–1161, October 1986.

[8] Ye Wnag, Leonid Yaroslavsky, Miikka Vilermo, and Väänänen. Some Peculiar Properties of the MDCT. In *5th International Conference on Signal Processing Proceedings, 2000. WCCC-ICSP 2000, Beijing, China*, volume 1, pages 61–64. IEEE, August 2000.

[9] Seymour Shlien. The Modulated Lapped Transform, Its Time-Varying Forms, and its Applications to Audio Coding Standards. *IEEE Transactions on Speech and Audio Processing*, 5(4):359–366, July 1997.

[10] Jürgen Herre and James D. Johnston. Enhancing the Performance of Perceptual Audio Coders by Using Temporal Noise Shaping (TNS). In *101st AES Convention, Los Angeles , CA, USA*. Audio Engineering Society, November 1996.

[11] Oliver Riuol and Martin Vetterli. Wavelets and Signal Processing. *IEEE Signal Processing Magazine*, 8(4):14–38, October 1991.

[12] Richard A. Haddad and Thomas W. PArsons. *Linear Prediction and Levinson Recursion*, pages 582–592. Computer Science Press, 1991.

[13] Aki Härmä and Unto K. Laine. A Comparison of Warped and Conventional Linear Predictive Coding. *IEEE Transactions on Speech and Audio Processing*, 9(5):579–588, July 2001.

[14] ISO/IEC 13818-3 Information Technology – Gerneric Coding of Moving Pictures and Associated Audio – Part 3: Audio, 1994.

[15] ISO/IEC 13818-7 Information Technology – Gerneric Coding of Moving Pictures and Associated Audio Information – Part 7: Advanced Audio Coding (AAC), 1997.

[16] ISO/IEC 14496-3 Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, 2000.

[17] Heiko Purnhagen. An Overview of MPEG-4 Audio Version 2. In von Recklinghausen [34], pages 157–168.

[18] Heiko Purnhagen and Nikolaus Meine. HILN – the MPEG-4 Parametric Audio Coding Tools. In *IEEE International Symposion on Circuits and Systems, 2000 (ISCAS 2000), Geneva, Switzerland*, volume 3, pages III–210–III–204. IEEE, May 2000.

[19] Eric D. Scheirer and Youngmoo E. Kim. Generalized Audio Coding with MPEG-4 Structures Audio. In von Recklinghausen [34], pages 189–204.

[20] Louis B. Fielder, Marina Bosi, Grant Davidson, Mark Davis, Craig Todd, and Steve Vernon. *AC-2 and AC-3: Low-Complexity Transform-Based Audio Coding*, pages 54–72. In Gilchrist and Grewin [35], 1996.

[21] James D. Johnston, Deepen Sinha, Sean Dorward, and Schuyler R. Quackenbusch. *AT&T Perceptual Audio Coding (PAC)*, pages 73–82. In Gilchrist and Grewin [35], 1996.

[22] Kyoya Tsutsui, Hiroshi Suzuki, Osamu Shimoyoshi, Mito Sonohara, Kenzo Akagiri, and Robert M. Heddle. *ATRAC: Adaptive Transform Acoustic Coding for MiniDisc*, pages 95–101. In Gilchrist and Grewin [35], 1996.

[23] Gerald D. T. Schuller, Yu Bin, Huang Dawei, and Bernd Edler. Perceptual Audio Coding Using Adaptive Pre- and Post-Filters and Lossless Compression. *IEEE Transactions on Speech and Audio Processing*, 10(6):379–390, September 2002.

[24] ITU-R BS.1116-1 Methods for the Subjective Assessment of Small Impairments in Audio Systems including Multichannel Sound, 1997.

[25] Gilbert A. Soulodre, Theodore Grusec, Michel Lavoie, and Louis Thibault. Subjective Evaluation of State-of-the-Art 2-Channel Audio Codecs. In *104th AES Convention, Amsterdam, The Netherlands*, May 1998.

[26] ITU-R BS.1534-1 Method for the subjective assessment of intermediate quality level of coding systems, 2003.

[27] ITU-R 1234.7 Perceptual Asessment of Audio Quality (PEAQ), 1998.

[28] Thilo Thiede, William C. Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John G. Beerends, Catherine Colomes, Gerhard Keyhl, Michael andStoll, Karlheinz Brandenburg, and Feiten Bernhard. PEAQ — the ITU Standard for Objective Measurement of Perceived Audio Quality. *JAES*, 48(1/2):3–29, January 2000.

[29] P. Kabal. An Examination and Interpretation of ITU-R BS.1387: Perceptual Evaluation of Audio Quality. TSP Lab Technical Report, Dept. Electrical & Computer Engineering, McGill University, May 2002.

[30] PEAQ Information web site http://www.peaq.org.

[31] EAQUAL source code download: http://www.mp3-tech.org/programmer/sources/eaqual.tgz.

[32] Alberto D. Dueñas, Rafael Pérez, Begoñas Rivas, Enrique Alexandre, and Antonio S. Pena. A Robust and Efficient Implementation of MPEG-2/4 AAC Natural Audio Coders. In *112th AES Convention, Munich, Germany*, number Preprint 5556. Audio Engineering Society, May 2002.

[33] Ogg Vorbis Specification http://www.xiph.org/ogg/vorbis/doc/Vorbis_I_spec.html.

[34] Daniel R. von Recklinghausen, editor. *The Proceedings of the 17th Int. Conference: High Quality Audio Coding*. AES, September 1999.

[35] Neil Gilchrist and Christer Grewin, editors. *Collected Papers on Digital Audio Bit-Rate Reduction*. AES, 1996.