An Experimental High Fidelity Perceptual Audio Coder

Bosse Lincoln
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

March 1998

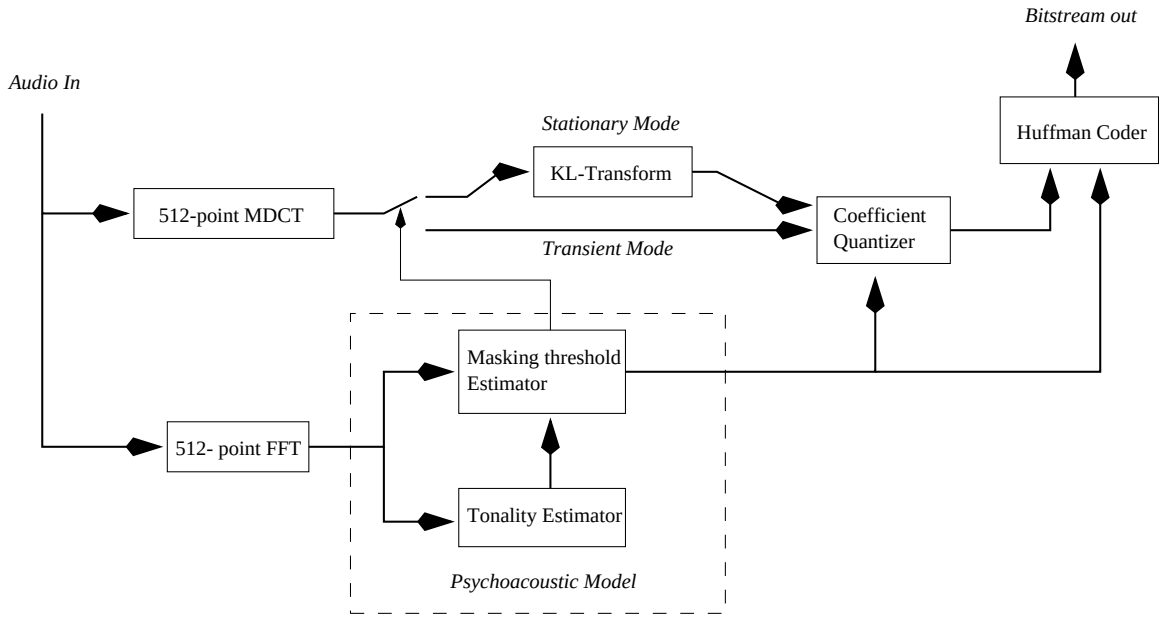

Figure 1:

1 Introduction

High quality audio in, e.g., CD or DAT players require large amounts of data. A CD stream consists of 44100 16-bit samples per channel, per second, which corresponds to 1.4 Mbit/s in stereo. Audio at that bitrate contains a lot of redundancies, which can be exploited in a *lossless coder* to get the bitrate down to about half of that. The human auditory system, though has many limitations, and thus lossy coder which exploits those properties, can be made much more efficient — 10 to 12 times less bits can often be used without perceptual loss. The audio coding community uses this extensively, and has come much further in this field than for example the video coders.

In this project, I have read papers in current perceptual audio coding, and with concepts from those (and some new ideas), I have implemented an experimental audio transform coder. The coder is not intended to be state-of-the-art but rather a tool for me to learn the difficulties that arise in a coder of this kind. The report is structured in the following way.

- *2 Current Implementations* describes some of the well-known implementations and standards that exist, and some ideas from those.
- *3 Human Audio Perception: Masking* describes the masking properties of the human auditory system, and the implemented model of this in the coder.
- *4 Audio Coding* goes through and motivates the quantization and bit coding used in the coder.
- *5 Results and Conclusions* shows bitrates and "quality" of some encoded audio clips. The audio clips are presented on the web.

2 Current Implementations

There are numerous implementations of perceptual coders in use in commercial products. Sony's ATRAC, used in the MiniDisc system, and the famous MPEG-1 Layer I, II and III [8] are well known examples . MPEG-2 AAC (Advanced Audio Coding, also called NBC, non-backwards-compatible) [7], and Dolby's AC-3 is currently considered top-of-the-line. The differences between the different coders are in the human audio perceptual model, the type of subband coding, and becoming more important, in special tricks to handle special artifacts in e.g transient sounds like castanets.

Some examples of the current implementations will be presented here, ordered after the type of subband splitting they use. All perceptual audio coders use some kind of subband splitting (or almost all, see e.g [9]). Any linear subband splitting can be interpreted as linear transforms, though, the the difference is really in the name of the method.

2.1 Subband Coders

Subband splitting of a signal is often performed with a two-channel uniform filterbank. The filterbank is used several times to achieve finer bandsplitting, in a filter tree:

$$\begin{aligned} 0 - 20000\ Hz &\rightarrow 0 - 10000\ Hz, 10000 - 20000\ Hz \\ 0 - 10000\ Hz &\rightarrow 0 - 5000\ Hz, 5000 - 10000\ Hz\ ... \end{aligned}$$

This way, the coder can split the input data into uniform bands, in each of which a quantizer, adopted to the masking threshold (see section 3), is applied.

The filterbank filterbank consists of four filters, $F_0(\omega)$, $F_1(\omega)$, $G_0(\omega)$ and $G_1(\omega)$. The F's are used to split the input data in a high and a low frequency band. Both bands are then subsampled with a factor of 2. To reconstruct, both bands are upsampled again, and filtered with the G's. Thus, the output $\hat{X}(\omega)$ can be written as

$$\begin{aligned}\hat{X}(\omega) &= \frac{1}{2}(F_0(\omega)G_0(\omega) + F_1(\omega)G_1(\omega))X(\omega) \\ &+ \frac{1}{2}(F_0(\omega+\pi)_0(\omega) + F_1(\omega+\pi)G_1(\omega))X(\omega+\pi) \text{ (aliasing).} \end{aligned} \quad (1)$$

To obtain aliasing cancellation, the following is required:

$$G_0(\omega) = F_1(\omega+\pi) \quad (2)$$

$$-G_1(\omega) = F_0(\omega+\pi). \quad (3)$$

Optimally,

$$\frac{1}{2}(F_0(\omega)G_0(\omega) + F_1(\omega)G_1(\omega)) = 1,\ \omega \in [0..\pi], \quad (4)$$

so that perfect reconstruction is achieved. This is, though, often hard to achieve. The filters used in most coders are Quadrature Mirror Filters (QMF), which achieve aliasing cancellation by choosing F_1 as the mirror image of F_0 (around $\pi/2$):

$$F_1(\omega) = F_0(\omega+\pi) = -G_1(\omega) = G_0(\omega+\pi). \quad (5)$$

To get good reconstruction without relying on the aliasing cancellation, which cannot be relied on in the presence of quantization noise, the QMF filters have to have a steep pass-to-stop-band transition.

2.1.1 MPEG-1, Layer I and II

The ISO MPEG-1 audio coding standard, described in [8], consists of three different layers (I, II and III). The layers differ quite a bit in coding and psychoacoustic models.

Layer I and II both use only a QMF filterbank of order 511, which has a 96 dB rejection of sidelobes and a steep pass-to-stop-band transition. The filterbank splits the input data in 32 subbands.

The psychoacoustic model consists of a 512 point FFT (1024 for layer II), from which a masking threshold is calculated. Peaks of the spectrum are considered tonal components and treated differently from the remaining noise components. The tonal and noise masking thresholds are added, to produce a masking threshold. This is used together with a linear quantizer to achieve appropriate rate-distortion performance in each subband, so that each subband has the same noise-to-mask-ratio and the bits add up to the preferred bit-rate. The quantized subband data is fixed-length encoded.

MPEG-1 Layer I at 384 kbit/s is used in Philips Digital Compact Cassette (DCC), and Layer II at 256 kbit/s is used for Digital Broadcast Audio (DBA).

2.2 Wavelet Coders

A wavelet transform can be viewed as a subband decomposition as described in section 2.1. The difference, though, is that the subband tree often is unbalanced in a wavelet decomposition, which means that the frequency splitting is nonuniform. This way, a filterbank which resembles the critical bands (see section 3.1.3) can easily be formed.

An example of a wavelet coder can be found in [10]. The authors claim perceptually nearly lossless coding at about 80 kbit/s at a coding complexity which requires only about 50% of the processing power of a Pentium-75 for either decoding or encoding.

2.3 Transform Coders

Even if the above methods can be interpreted as transforms, there are coders which more explicitly use linear transforms, such as the FFT, DCT (Discrete Cosine Transform), MDCT (Modified Cosine Transform) and so on. Currently most used is the MDCT (see section 4.1.1), which combines 50% overlap with "critical subsampling". Dolby's AC-3 uses purely MDCT for coding, and MPEG-1 layer III and MPEG-2 AAC [1] use hybrid subband-MDCT methods.

2.3.1 MPEG-2 AAC

A quick explanation what is done in the scalable sampling rate profile of MPEG-2 AAC [7] is interesting to understand the current state-of-the-art. The audio data is first split into four uniform subbands using a Polyphase Quadrature Filter (PQF). For each of the four subbands an individual gain is transmitted as side information. The gain-controlled subband data is then transformed using an MDCT of length 256 (or 32 for transient conditions). The window used for the MDCT is either the Kaiser-Bessel derived (KBD) or the sine window, which has different spectral characteristics, suitable for different signals. For transient conditions, a shorter window is used for improved time resolution.

The MDCT coefficients are predicted from the two preceding frames, using a separate LMS-adapted (Least Mean Square) predictor for every frequency band. This improves coding efficiency for stationary signals. Residuals after the prediction are non-uniformly quantized and coded using one of 12 different Huffman codes.

In MPEG-2 AAC there are a lot of optional extra features. One of the most interesting is Temporal Noise Shaping (TNS), which works well for transient signals. The idea is, that a tonal signal in the time domain has transient peaks in the frequency domain. The dual of this, is that a signal which is transient in the time domain is "tonal" in the frequency domain, i.e consists mainly of a few sines. "Tonal" sounds are easily predicted using a LPC approach. Thus, a simple linear predictor is used to predict the next *spectral* sample (going from low frequencies to high) from it lower-frequency neighbors.

3 Human Audio Perception: Masking

The human auditory system has some interesting properties, which are exploited in perceptual audio coding. We have a dynamic frequency range from about 20 to 20000 Hz, and we hear sounds with intensity varying over many magnitudes. The hearing system may thus seem to be a very

[1] In the Scalable Sample Rate (SSR) profile.

wide-range instrument, which is not altogether true. To obtain those characteristics , the hearing is very *adaptive* — what we hear depends on what kind of audio environment we are in. In the presence of a strong white noise, for example, many weaker sounds get *masked* (see section 3.2), and thus we cannot hear them at all. Some of these masking characteristics are due to the physical ear, and some are due to the processing in the brain.

Using masking principles, experiments have been performed by others where correctly shaped noise has been added to audio data without audible effect down to an SNR of 25 dB. On the other hand, deliberately "wrongly" shaped noise, i.e noise with high energy in sensitive areas can be audible up to an SNR of 90 dB.

I will now show some of the most important masking properties of the ear, and the models of those. The models are combined in the coder to produce a *masking threshold* curve every 256 samples (5.8 ms), which is used to quantize the audio data. According to the model, noise under that threshold is completely inaudible to the listener. See section 3.4 for a description how the masking threshold is used in quantization.

3.1 Basics

In this section, some of the most basic masking properties of the ear are described. These are always present, and are mainly due to the properties of the pysical ear.

3.1.1 Absolute Threshold of Hearing

The simplest property to exploit is the ATH (Absolute Threshold of Hearing). We simply cannot hear sounds which are too weak. The ATH varies a lot with the frequency of the test sound, which can be seen in Fig. 2. The curve is approximated from experiments on a number of subjects (see [1]).

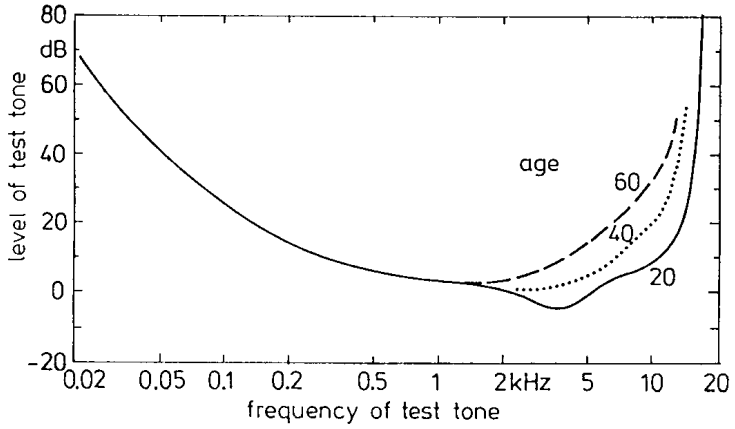

Figure 2: The absolute threshold of hearing. The numbers indicate the age of the test subjects.

3.1.2 Implemented Model for ATH

From [2] I take the approximation of the curve in Fig. 2 as

$$ATH(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4, \tag{6}$$

where $ATH(f)$ denotes the ATH in dB, and f is the frequency in Hz. The problem in digital audio coding, is that one cannot know what absolute level the sound will be played at. One common solution is to set the lowest point on the curve in eq. 6 to be equal to the sound pressure level of a sine with amplitude ± 1 LSB. In the coder, 16-bit samples with normalized amplitude to ± 1 are used. Thus, the smallest possible sine has amplitude $A = 1/2^{15}$, and has a power of $A^2/2 = 1/2^{2 \cdot 15+1} = -93.32\text{dB} \approx -90$ dB. The model used in the coder will thus be $ATH'(f) = ATH(f) - 90$.

3.1.3 Critical Bandwidths

Much of what is done in Simultaneous Masking (see section 3.2) is based on the existence of *critical bands*. The hearing works much like a non-uniform filterbank, and the critical bands can be said to approximate the characteristics of those filters. Critical bands does not really have specific "on" and "off" frequencies, but rather width as a function of frequency – critical bandwidths.

The parameters for critical bandwidths can be derived in many ways [1], of which most give consistent results – which in some sense proves their presence. The critical bandwidths were first derived from the tone masking of white noise. Under the assumption that the tone was masked when the power of the noise in that critical band was equal to the about 1/4 of the power of the tone, critical bandwidth was determined to be

$$BW(f) = \begin{cases} 100\ Hz & f < 500\ Hz \\ 0.2f\ Hz & f \geq 500\ Hz \end{cases} \tag{7}$$

These bandwidths are used to form a critical band scale, rather similar to the logarithmic musical scale. Conversion between this *bark* frequency scale and Hz can be approximated via the function [2]

$$z(f) = 13 \arctan(0.00076f) + 3.5 \arctan((f/7500)^2). \tag{8}$$

3.2 Simultaneous Masking

Simultaneous masking is a property of the human auditory system where some sounds simply vanish in the presence of other sounds with certain characteristics (so called *maskers*).

In the model described in the implemented coder, a short-term frequency representation of the audio is used to help estimate the masking function. To get a frequency representation, a 512-sample FFT is performed on the audio every 256 samples to form $\mathcal{F}(f)$. Every frequency bin is mapped to a corresponding critical band (a real number), using eq. 8.

3.2.1 Spreading Function

In experiments [1][2], masking experiments has been made with two narrowband noise maskers, with a frequency difference Δf, masking a tone in between, and the other way around. It shows that the masking from the two maskers is approximately constant as long as the Δf is less than the

critical bandwidth at that frequency. But masking does not only occur within the critical band, but also spreads to neighboring bands. A *spreading function* $SF(f, a)$ can be defined, where f is the frequency and a the amplitude of a masker. This function would give a masking threshold for neighboring frequencies of a single tone. The simplest function would be a triangular function with slopes of +25 and -10 dB / bark, but a more sophisticated one is highly nonlinear and depends on both frequency and amplitude of masker.

3.2.2 Implemented Model for Spreading Function

A spreading function which takes into account both in-band masking and inter-band masking is (taken from [2])

$$sf(z) = 15.81 + 7.5(z + 0.474) - 17.5\sqrt{1 + (z + 0.474)^2}, \tag{9}$$

where z is the frequency in barks. This equation is modified to take into account the decreasing slope at higher masker amplitudes i in the following manner:

$$SF(z) = (15.81 - i) + 7.5(z + 0.474) - (17.5 - i)\sqrt{1 + (z + 0.474)^2}, \tag{10}$$

where $i = \min(5 \cdot |\mathcal{F}(f)| \cdot BW(f), 2.0)$, f is the frequency of the masker and $BW(f)$ is the critical bandwidth at f. A higher value of i gives a flatter $SF'(z)$, and the formula for i is an experimentally found heuristic which compensates a frequency bin if it is only a small part of a wide critical band. Setting the max of i to 2.0 was necessary for some test sounds, where e.g a base drum could make a large part of the high frequency spectra vanish. See Fig. 3 for an illustration of the spread function.

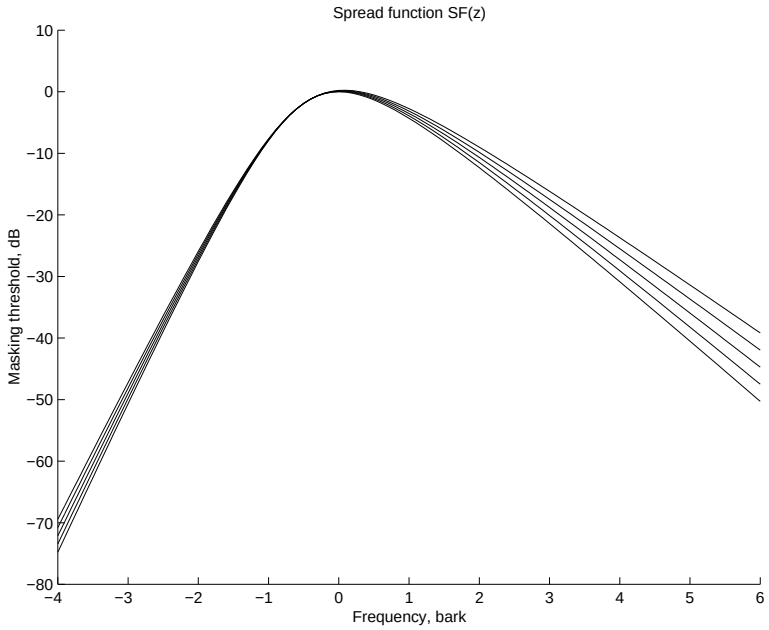

Figure 3: The masking spread function for a single masker. The scaled amplitude i varies from 0 to 2, where 2 corresponds to the wider spread.

3.2.3 Tonality of Maskers

Experiments have shown that there is a big difference between noise maskers and tone maskers, and also the characteristics of the masked audio. [1] mostly describes the cases *noise-masking-tone* and *tone-masking-noise*. When coding data, the resulting quantization errors is seen as noise, and thus only the tonality of the *masker*, and not the masked data, has to be estimated. The tonality is used in section 3.2.5 for estimation of the masking threshold.

3.2.4 Implemented Model for Tonality

The tonality of a frequency bin $\mathcal{F}(f)$ is estimated by looking at the predictability of the *phase* and *magnitude* of that Fourier coefficient. The predictors are defined as follows:

$$\hat{\varphi}_t = 2 \cdot arg(\mathcal{F}_{t-1}(f)) - arg(\mathcal{F}_{t-2}(f)), \text{ and} \tag{11}$$

$$\hat{M}_t = |\mathcal{F}_{t-1}(f)|. \tag{12}$$

The phase is thus linearly extrapolated from two former time instances, and magnitude is simply assumed to be the same as last. This gives no prediction error for *one* stationary sine within the frequency band. The tonality is then estimated from the maximum prediction error of the last two phase values and the last magnitude:

$$t(f) = 1 - \max(\varphi_{err_t}, \varphi_{err_{t-1}}, M_{err_t}), \tag{13}$$

where $\varphi_{err_t} = \hat{\varphi}_t - arg(\mathcal{F}_t(f))/\pi$ and $M_{err_t} = (\hat{M}_t - |\mathcal{F}_t(f)|)/\max(\mathcal{F}_t(f), \hat{M}_t)$. This model gives a weighted average t of about 0.9 for highly tonal sting music, and 0.3 for white noise. Of course the parameters in the masking threshold (section 3.2.5) estimation is adapted to these (non-ideal) values.

3.2.5 Implemented Model for Masking Threshold

To produce the masking threshold from the spread function and the masker, we need to know the *tonality* $t(z)$ and the spreading function $SF(z)$ of the masker. After an idea taken MPEG-1 layer I [2], the following model for masking threshold $M(z)$ is used:

$$M(z) = P(z_m) + SF(z - z_m) - k(t(z))z_m - l(t(z)) \text{ (db)}, \tag{14}$$

where z_m is the frequency in barks for the masker, z is the masked frequency in barks and $P(z_m)$ is the power of the masker in dB. The functions $k(t)$ and $l(t)$ are experimentally found as a linear combinations of constants for pure noise and pure tones:

$$k(t) = 0.3t + 0.5(1 - t), \text{ and} \tag{15}$$

$$l(t) = 34t + 20(1 - t) \tag{16}$$

3.2.6 Addition of Simultaneous Masking

The masking from different maskers has to be added to form the final masking function. One could argue that the maskers should be added as powers or amplitudes. According to [1], though, addition of two equal maskers can give a resulting masker up 12 dB higher than two maskers alone. This would mean, that the addition of maskers would be defined as follows:

$$SUM(M_1, M_2) = 40log(10^{M_1/40} + 10^{M_2/40}), \tag{17}$$

i.e as the addition of *square roots of amplitudes*. This model does not work for big differences in M_1 and M_2, so in the coder the addition was defined as addition of amplitudes:

$$SUM(M_1, M_2) = 20log(10^{M_1/20} + 10^{M_2/20}) \tag{18}$$

Using this model for summation, all spectral components of the audio are considered maskers, and their individual masking thresholds (calculated in section 3.2.5) added.

An example of the produced masking threshold, including both simultaneous and temporal masking (described in section 3.3), can be seen in Fig. **??**.

3.3 Temporal Masking

Temporal masking is the characteristic of the auditory system where sounds are hidden due to maskers before or even after that time. The effect of masking after a strong sound is called *post-masking*, and can be in effect up to 200 ms. The *pre-masking*, where a sound actually is masked by something which appears *after* it, is relatively short and may last up to 20 ms. An illustration of this can be found in Fig. **??**.

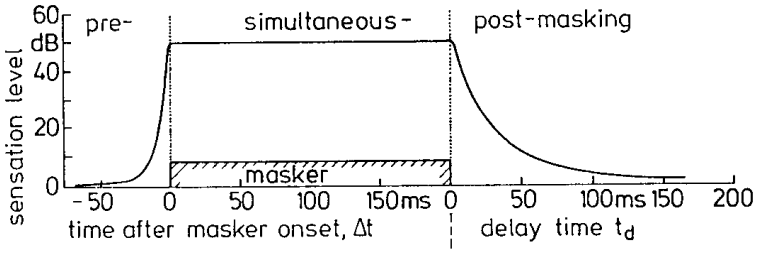

Figure 4: The effect of temporal masking. Picture taken from [1].

3.3.1 Implemented Model for Post-Masking

As can be seen in Fig. **??**, the post-masking effect varies with the length of the masker burst, from 5-10 ms up to 200 ms. This effect is exploited in the coder in by a one-pole filter on the masking threshold over time:

$$M'_t = 0.85M'_{t-1} + 0.15M_t \tag{19}$$

The final output mask is then taken as $M_t = \max(M'_t, M_t)$. With this definition, a short burst does not affect M'_t much, while a longer burst will sustain the masking for a number of frames even though M_t goes low again.

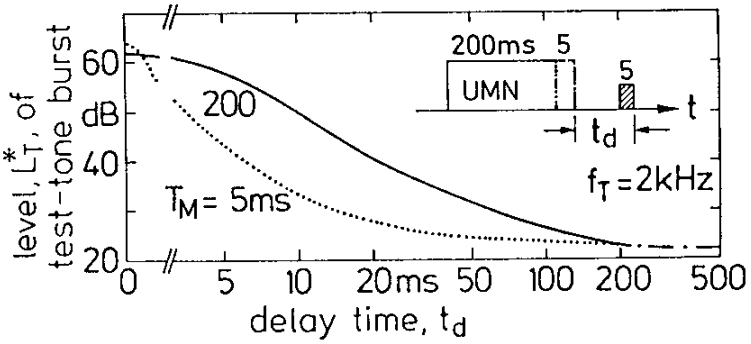

Figure 5: Temporal masking with varying masker burst size. Picture taken from [1].

3.3.2 Implemented Model for Pre-Masking

The pre-masking is too short to be exploited in the same way as in post-masking, but it is still important. Pre-masking comes in useful to hide the effect of *pre-echos*, which can become audible in transient sounds. Pre-echos comes from the fact that quantized transform coefficients produce noise in all time instants in the time domain. A quiet signal block with a transient in the end (e.g a drum) will thus be noisy even before the transient, where it can be heard. By making the transform blocks short enough, this effect can be hidden by the pre-masking.

In this coder, the audio is transformed in 512-length MDCT blocks (11.6 ms), every 256 samples (5.8 ms). This should be enough to hide pre-echos.

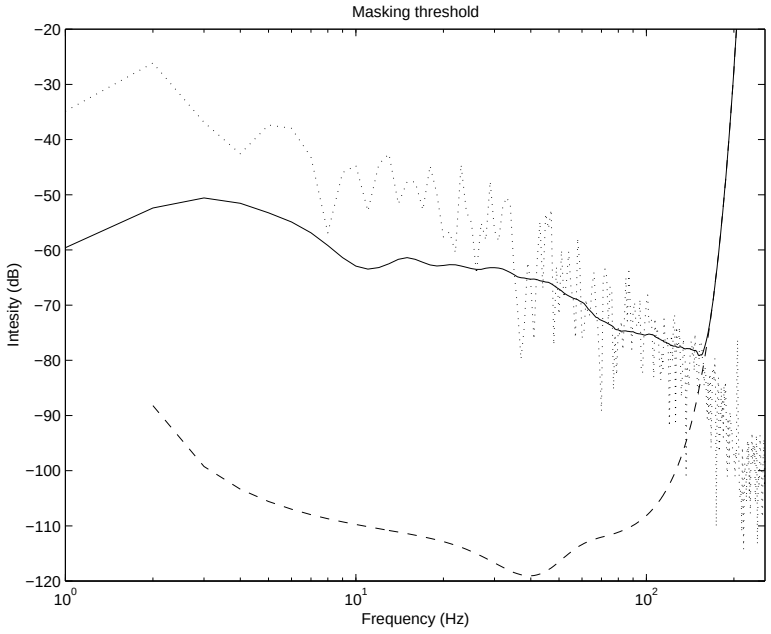

Figure 6: An example of the masking threshold produced by the psychoacoustic model in the coder. The example frame is number 500 (after 2.9 s) in sample `jacob.wav`.

3.4 Quantization using the Masking Threshold

The main reason for using a psychoacoustic model for audio compression is that given a masking threshold $M_t(f)$, the amplitude at that frequency may be quantized with a step size proportional to $M_t(f)$. The quantization can be seen as introduction of noise with power proportional to $M_t(f)$:

$$\mathcal{F}_Q(f) = \mathcal{F}(f) + noise(f)$$

The quantization error can then easily be adjusted to be lower than the masking threshold, and thus become inaudible.

In the implementation of the coder, the psychoacoustic model is adjusted using only a quantizer with step size $M_t(f)$ on every transform coefficient. This way, the coding is kept separate from the psychoacoustic model. Thus, when I start to design the coder, I can be certain to get perceptually perfect data independent of coding method.

4 Audio Coding

All lossy source coding techniques can be interpreted as vector quantization (VQ) with variable length coding [2]. The coding scheme used in this coder uses an orthonormal transform (see 4.1.3) as a fixed codebook for vector quantization of MDCT (see 4.1.1) transform coefficients. Variable length coding is done by a number of *Huffman codes* on the VQ coefficients.

Before I can move on to the actual coding, I introduce the transforms used in the coder.

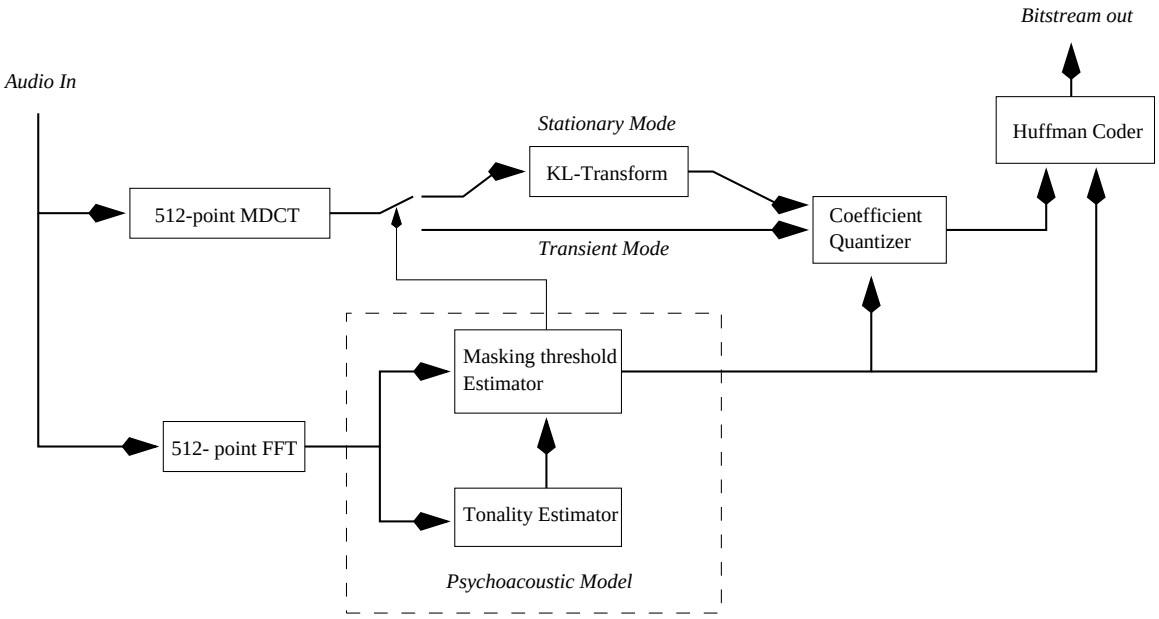

Figure 7: A schematic view of the coder.

[2] See e.g A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992. (Fourth Printing, 1995)

4.1 Transforming

Two different types of linear transforms are used in the coder. The first one is the Modified Discrete Cosine Transform, which is used to represent the audio blocks in the frequency domain, so that the masking threshold can be used directly for quantization. The second is the Karhunen Lòeve Transform (KLT), which is used to efficiently encode blocks of MDCT coefficients, where the coefficient blocks correspond closely to critical bands. The KLT is used because of its optimality in sense of energy compaction.

One might argue that a linear transform of a linear transform is just another linear transform, and thus I am just wasting time. In this case, though, the frequency basis is needed to quantize the data according to the masking threshold, and thus the KLT cannot immediately be used without the MDCT step.

A linear transform can be thought of in many different ways. The MDCT here is best viewed as a subband filter bank, while the KLT is viewed as a change of basis in an n-dimensional space, where n is the length of the transform block.

4.1.1 Modified Discrete Cosine Transform (MDCT)

The MDCT is a linear orthogonal lapped transform, based on the idea of time domain aliasing cancellation (TDAC). It was first introduced in [3], and further developed in [4].

MDCT is critically sampled, which means that though it is 50% overlapped, a sequence data represented with MDCT coefficients takes equally much space as the original data. This means, that a single block of IMDCT data does not correspond to the original block, on which the MDCT was performed, but rather to the odd part of that. When subsequent blocks of inverse transformed data are added (still using 50% overlap), the errors introduced by the transform cancels out $\Rightarrow$ TDAC. Thanks to the overlapping feature, the MDCT is very useful for quantization. It effectively removes the otherwise easily detectable blocking artifact between transform blocks. The used definition of MDCT is (a slight modification from [5]) is:

$$X(m) = \sum_{k=0}^{n-1} f(k)x(k)\cos(\frac{\pi}{2n}(2k+1+\frac{n}{2})(2m+1)), \text{ for } m = 0..\frac{n}{2}-1 \tag{20}$$

and the IMDCT:

$$y(p) = f(p)\frac{4}{n}\sum_{m=0}^{\frac{n}{2}-1} X(m)\cos(\frac{\pi}{2n}(2p+1+\frac{n}{2})(2m+1)), \text{ for } p = 0..n-1 \tag{21}$$

where $f(x)$ is a window with certain properties (see [5]). The sine window

$$f(x) = \sin(\pi\frac{x}{n}) \tag{22}$$

has the right properties, and is used in this coder. The MDCT in the coder is performed with a length of 512, and thus 256 new samples are used for every block.

4.1.2 A Fast MDCT Implementation

The MDCT can be calculated using FFT. The naive approach, though, requires a $2n$ length FFT for a length n block, because of the odd transform. There are faster approaches though [6]. The

MDCT can be rewritten as an odd-time odd-frequency discrete Fourier transform (O^2DFT)

$$X(m) = \Re(O^2DFT_{shft(fx)}(m)) = \Re(\sum_{k=0}^{N-1} f(k-\frac{n}{4})x(k-\frac{n}{4})e^{\frac{-j\pi}{2n}(2k+1)(2m+1)}). \tag{23}$$

[6] presents a fast algorithm for calculating

$$\mathcal{W} = O^2DFT(odd(f(k-\frac{n}{4})x(k-\frac{n}{4}))) = X(m) \tag{24}$$

as

$$\mathcal{W}_{2k} = \Re(\mathcal{P}_k) \tag{25}$$
$$\mathcal{W}_{n/2+2k} = \Im(\mathcal{P}_k) \tag{26}$$
$$\mathcal{W}_{2k+1} = -\mathcal{W}_{n-2(k+1)} \tag{27}$$

where

$$\mathcal{P}_k = 2e^{-j\frac{2\pi}{n}(k+\frac{1}{8})} \underbrace{\sum_{r=0}^{n/4-1} ((x(2r) - jx(n/2+2r))e^{-j\frac{2\pi}{n}(r+\frac{1}{8})})e^{-j\frac{2\pi}{n/4}rk}}_{n/4 \text{ point FFT}}. \tag{28}$$

Thus, the MDCT can be calculated using only one $n/4$ point FFT and some pre- and post-rotation of the sample points. The IMDCT can be calculated in a similar way. See the code for a more detailed description.

4.1.3 Karhunen Lòeve Transform (KLT)

The KLT is a linear transform where the basis functions are taken from the statistics of the signal, and can thus be adaptive. It is optimal in the sense of *energy compaction*, i.e it places as much energy as possible in as few coefficients as possible. The KLT is also called *Principal Component Analysis*, and for discrete signals, the is also equivalent with the *Singular Value Decomposition*. The transform is generally not separable, and thus the full matrix multiplication must be performed:

$$\mathbf{X} = U^T\mathbf{x},\ \mathbf{x} = U\mathbf{X}, \tag{29}$$

where the U is the basis for the transform. U is estimated from a number of $\mathbf{x}_i,\ i \in [0..k]$:

$$U\Sigma V^T = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_k] = A \Rightarrow U = \text{eigvec}(AA^T) \tag{30}$$

The adaptiveness is not used in the coder, and the basis functions are calculated off-line in MatLab.

4.2 Quantization and Coding

In this section, I present a method for quantizing the audio data according the the masking threshold.

4.2.1 Coefficient Clustering - Frequency Domain

The first thing to be done in a more advanced approach is to cluster the masking thresholds in the frequency, so that quantization step sizes do not take too many bits. This is often done in critical bands. This coder uses 24 slightly adjusted critical bands, in each of which the amplitude mean of the masking threshold is used as quantization step size. The bands were adjusted to better use the total clipping which often occurs at high frequencies, due to the ATH.

Band	First bin	Last bin	Coefficient ratio
1	0	1	0.874486
2	2	3	0.810296
3	4	5	0.831361
4	6	7	0.835856
5	8	9	0.894511
6	10	11	0.938870
7	12	13	1.000000
8	14	15	1.000000
9	16	18	1.000000
10	19	21	1.000000
11	22	24	1.000000
12	25	28	1.000000
13	29	33	0.932482
14	34	38	0.962853
15	39	45	0.839628
16	46	53	0.789081
17	54	64	0.685002
18	65	77	0.684814
19	78	91	0.712828
20	92	108	0.683237
21	109	128	0.672980
22	129	149	0.681761
23	150	170	0.818602
24	171	255	0.894698

The table shows the clustering of the frequency bins into bands. The "Coefficient ratio" column is described below for stationary coding mode.

4.2.2 Coefficient Clustering - Time Domain

At tonal parts of the signal, the frequency coefficients are highly correlated in the time domain, since a tone corresponds to a stationary peak in the frequency domain. This is exploited in the encoder by always encoding four MDCT blocks at a time. To not get artifacts at transitions, i.e when the masking threshold changes abruptly, two modes of operation are introduced, one of which is chosen for each band:

1. **Transient mode.** The four MDCT blocks are encoded individually, and thus having an individual encoder step size per block and band. The MDCT coefficients are quantized and

encoded as described in section 4.2.4.

2. **Stationary mode.** The four MDCT blocks are jointly coded, using only one quantizer step size per band. The coefficients are transformed using a fixed KLT (section 4.1.3), quantized and encoded. The KLT basis was estimated from the tonal mono sequence `strings.wav`, which contains about 2000 frames.

The mode decision is done based on the mean of estimated variances of the masking threshold over the four blocks for all frequencies in the band:

$$Var = \sum_{k \in band(n)} \left(\sum_{p=0}^{3} (M(k,p) - \frac{1}{4} \sum_{l=0}^{3} M(k,l))^2 \right) \tag{31}$$

If $Var > thresh$, then the Transient mode is used, otherwise Stationary mode. The value $thresh = 10^{-3}$ which is used in the coder, was found empirically.

The Stationary mode tries to use the energy compaction property of the KLT in the following fashion: Since the first few coefficients of the KLT probably have higher energy then the later ones, the transform can without greater loss be performed with only a subset of the basis vectors U. Thus, the p last coefficients from the KLT are never trasmitted. Experiments has shown that this works fine in the bands with many frequency bins, which leads to the following heuristic for determining which coefficients to skip: Use the P first coefficients, where P is chosen so that

$$\sum_{k=0}^{P-1} X_k^2 \geq 0.9(\sum_{k=0}^{n} X_k^2)(1 + \frac{1}{band + 3}) \tag{32}$$

$$P \geq 1.5 \cdot band, \tag{33}$$

where $band \in [0..23]$ is the band number. This heuristic "cuts" the transform when enough energy has been included. More energy is required for lower bands, where tonal instruments, such as strings, sound very bad without that restriction.

An experiment on audio clip `music.wav` gives the average "coefficient ratio" in table 4.2.1, where 1.0 corresponds to sending *all* coefficients, and 0 to not sending any. The effect of the weighting equations above is clearly visible in the table. In e.g *music.wav*, the overall bitrate is 121 kbit/second without the KLT and 106 with. It should be noted also that the KLT option without the skipping of coefficients gives *no* bitrate savings. Thus, the only gain I get from the KLT is that the quantization noise from zeroed coefficients can be spread over the whole band.

4.2.3 Coefficient Quantization

The coefficients in a band are all quantized with the same step size Q. To store Q in the bit stream, it is indexed as

$$Q_i = \text{round}(10 \log(Q) + 35), \ (\min\ 0,\ \max\ 50), \tag{34}$$

and thereafter Huffman encoded. The Q controls a nonuniform coefficient quantizer, with the following characteristic.

$$x_Q = \text{sign}(x) \text{round} \left(\left| \frac{x}{Q} \right|^{1/c} \right), \tag{35}$$

where a $c = 1.3$ has proved to work well. A nonuniform quantizer does not follow the masking threshold theories in section 3.4, but works very well — it is used in e.g MPEG-2 AAC [7].

The coefficients of any orthogonal linear transform can be quantized as described above, since

$$\begin{aligned} Qerr(\mathbf{X}) &= (\mathbf{X}-\mathbf{X_Q})^T(\mathbf{X}-\mathbf{X_Q}) = (U^T(\mathbf{x}-\mathbf{x_Q}))^T(U^T(\mathbf{x}-\mathbf{x_Q})) = \\ &= ((\mathbf{x}-\mathbf{x_Q}))^T((\mathbf{x}-\mathbf{x_Q})) = Qerr(\mathbf{x}) \end{aligned} \tag{36}$$

and so the quantizer makes no difference between the coefficients from the different modes in section 4.2.2.

4.2.4 Bit Coding

The quantized coefficients are coded using an individual Huffman table for each band. The string of coefficients are encoded one by one until the last nonzero coefficient, where a separate Huffman table stores the number of zeros in the end of the block. The Huffman tables are not published here for the reason that they have not been fixed yet. In the coder experiment program, the Huffman trees are created dynamically based on information from the processed signal.

4.3 Stereo Coding

Stereo signals need some special considerations to exploit all redundancies:

- The *masking threshold* can be assumed to be the same for both channels. Thus, the mean of the two channels is fed to the masking threshold estimator. The threshold is of course only transmitted once for both channels.
- Often, the audio is centered, i.e approximately equal in both channels. This is used by encoding $m = \frac{1}{2}(l+r)$ and $d = l - m$, i.e the *mean* and *difference* signals.

Experiments has been performed on the assumption that, in every frequency band, the signal is only a *panned* monophonic signal, i.e the same signal is in both channels except for a scaling factor. The left and right channel is seen as two vectors, orthogonal to each other (see Fig. 8). The signal data in a band is used to least-square-fit a vector between the two vectors, representing the most probable direction (the optimal direction would be found with an SVD, though). This vector, and an orthogonal counterpart, is chosen to be the new basis for the signal, and the left and right signals are transformed to the new basis. Even without counting the bits for the basis information, this did not improve the results from just the mean-difference approach.

5 Results and Conclusions

The coder has been tested and developed mainly with about 10 second sampels from music CD's, most of which are in the table below. After finishing the development, the MPEG-2 testfile `music.wav`, containing many hard instruments, was tested. Some of the instruments were treated well, and some, like for example the castanets sounded rather bad. The castanetes of course require some kind of pre-echo detection to sound good. Some examples take from that `music.wav` are shown below.

All of the following samples can be found in `wav` format on the web, at `http://ccrma.stanford.edu/~bosse/`. Note that no rate controlling module is developed, and thus bitrates vary a lot with the type of

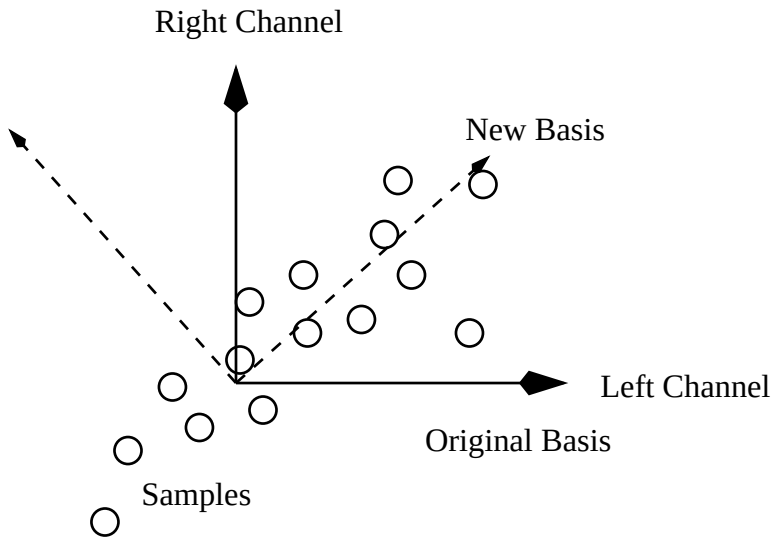

Figure 8: The left-right stereo model, discussed in section 4.3.

signal. From the values below, one can deduce that a fair perceptual lossless coding can be achieved at about 128 kbit/second for stereo data and 75 kbit/second for mono.

Audio Stream	Bits/s	M/S	Apparent artifacts
Mono audio			
`mixed.wav`	67 kb/s	M	
`jacob.wav`	71 kb/s	M	
`cardigans.wav`	73 kb/s	M	
`strings.wav`	58 kb/s	M	
Stereo audio			
`music.wav`	106 kb/s	S	Especially triangle and castanets
`tpd.wav`	118 kb/s	S	
`jacob.wav`	124 kb/s	S	
`castanets.wav`	103 kb/s	S	Very audible preecho
`instruments.wav`	108 kb/s	S	
`oasis.wav`	118 kb/s	S	The “s” in “sunday’
Low bitrate			
`oasis.wav`	89 kb/s	S	Easy to detect
`jacob.wav`	83 kb/s	S	Easy to detect

M/S means mono/stereo. The low bitrate signals are coded with a masking threshold multiplied by a factor $\frac{1}{0.5}$ and $\frac{1}{0.3}$ respectively.

The encoder described in this report is apparently rather undeveloped. To improve the coder, I would like to add some kind of transient coding, for example using wavelets. Transform-wavelet hybrid coders (see e.g [11]) has become more popular and show good results. Also, an adaptive prediction in either the time- or transform domain would decrease bitrate in stationary signals

(although some of this is exploited by the KLT).

This project did not result in a coder with many new features, but in some experience and knowledege for me in the field of high fidelity perceptual audio coders.

References

[1] Zwicker E, Fastl H, Psychoacoustics, Springer-Verlag, Berlin, Germany, 1990

[2] Painter T, Spanias A, A Review of Algorithms for Perceptual Coding of Digital Audio Signals, `http://www.eas.asu.edu/~speech/ndtc/dsp97.ps`

[3] Princen J, Bradley A, Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation, IEEE Transactions, ASSP-34, No.5, Oct 1986, pp. 1153-1161.

[4] Princen J, Johnson A, Bradley, A, Subband/Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation, Proc. of the ICASSP 1987, pp 2161-2164.

[5] Sporer T, Brandenburg K, Edler B, The Use of Multirate Filter Banks for Coding of High Quality Digital Audio, 6th European Signal Processing Conference (EUSIPCO), Amsterdam, June 1992, Vol.1 pp. 211-214.

[6] Gluth R, Regular FFT-Related Transform Kernels for DCT/DST-based polyphase filter banks, ICASSP 91, pp. 2205-8 vol.3

[7] Bosi M et al, ISO/IEC MPEG-2 Advanved Audio Coding, Journal of the Audio Engineering Society, No. 10, Oct 1997, pp. 789-813

[8] Brandenburg K, Stoll G, ISO-MPEG-1 Audio: A Genereic Standard for Coding of High Quality Digital Audio, Journal of the Audio Engineering Society, No. 10, Oct 1994, pp. 780-792

[9] Härmä, Laine U, Karjalainen M, An Experimental Audio Codec Based on Warped Linear Prediction of Complex Valued Signals, Proc. of the ICASSP 1997, pp 323-326.

[10] Dobson K, Yang J, Smart K, Guo K, High Quality Low Complexity Scalable Wavelet Audio Coding, ICASSP 97, pp. 327-330

[11] Sinha D, Johnston J, Audio Compression at Low Bit Rate Using a Signal Adaptive Switched Filterbank, Proc. of the ICASSP 1996, pp. 1053-1056