Speeding up HILN – MPEG-4 Parametric Audio Encoding with Reduced Complexity

Heiko Purnhagen, Nikolaus Meine, Bernd Edler

Laboratorium für Informationstechnologie
University of Hannover
Schneiderberg 32, 30167 Hannover, Germany
Phone: +49-511-762-5033, Fax: +49-511-762-5052
{purnhage,meine,edler}@tnt.uni-hannover.de

Parametric modelling permits an efficient representation of audio signals and is utilised for very low bit rate coding by the MPEG-4 Standard. Here we look at the MPEG-4 parametric audio coding tools "Harmonic and Individual Lines plus Noise" (HILN) which are based on a decomposition of the audio signal into components that are described by appropriate source models and represented by model parameters. Until now, HILN encoding mainly focused on maximum audio quality at the expense of high computational complexity. In this paper, different approaches to speed up HILN encoding are presented and the tradeoff between computational complexity and audio quality is analysed.

1 INTRODUCTION

In the context of evolving multimedia applications, new demands for efficient and flexible representation of audiovisual content arise. Besides high coding efficiency required to cope with the limited bandwidth of the internet or in mobile communication, also new functionality like flexible access to coded data and manipulation by the recipient is desired. To address these requirements and develop inter-operable solutions, ISO/IEC started its MPEG-4 standardisation activities "Coding of audiovisual objects." The MPEG-4 Audio Standard provides tools for coding of natural and synthetic audio objects and composition of such objects into an "audio scene." Natural audio objects (such as speech and music) can be coded at bitrates ranging from 2 kbit/s to 64 kbit/s and above using parametric or CELP-based speech coding tools and parametric or transform-based audio coding tools.

The MPEG-4 parametric audio coding tools "Harmonic and Individual Lines plus Noise" (HILN) permit coding of audio signals at bitrates of 4 kbit/s and above using a parametric representation of the audio signal. The basic idea is to decompose the input signal into components which are described by appropriate source models and represented by model parameters. This approach allows to utilise more advanced source modelling than the spectral decomposition used in transform-based coders. As known from speech coding, where

specialised source models resemble the human vocal tract, advanced source models can be advantageous in particular for very low bitrate coding.

In the HILN encoder, the input signal is decomposed into different signal components and then the model parameters for the components are estimated: *Individual sinusoids* are described by their frequencies and amplitudes, a *harmonic tone* is described by its fundamental frequency, amplitude, and the spectral envelope of its partials, and a *noise* signal is described by its amplitude and spectral envelope. The modelling of transient components is improved by optional parameters describing the amplitude envelope. Due to the very low target bitrates of typically 6 to 16 kbit/s, only the parameters for a small number of components can be transmitted. Therefore a perception model is employed to select those components that are most important for the perceived quality of the signal. The component parameters are finally quantised, coded, and multiplexed to form a bitstream. In the HILN decoder, the parameters of the components are decoded and then the component signals are re-synthesised according to the transmitted parameters. By combining these signals, the output signal of the HILN decoder is obtained.

The most complex task in the HILN encoder is the decomposition of the input signal into the perceptually most relevant components and the estimation of the component parameters. In our reference encoder, this is accomplished by an analysis/synthesis approach that iteratively extracts signal components from the current time frame of the input signal. Selection of the most relevant components is controlled by a perception model in the analysis/synthesis loop. A high-accuracy estimation of component parameters (e.g. the frequency of a sinusoid) is required to consider the properties of perception. In addition, components and their parameters are tracked from frame to frame. Until now, HILN encoder development mainly focused on maximum audio quality to proof the merit of the HILN parametric audio coding tools within the MPEG-4 standardisation process. This, however, has led to a quite high computational complexity of the encoder.

Since only the bitstream format and decoding procedure are standardised by MPEG, encoder optimisation is possible even after the standard was finalised. In this paper, different approaches to speed up HILN encoding are presented. In order to reduce computational complexity, simplified algorithms for signal decomposition, for parameter estimation, and for the perception model are investigated. In addition, some of these algorithms can be implemented very efficiently in the frequency domain. With these optimisations, real-time encoding on common PCs is easily possible. Based on HILN encoders implementing different optimisation approaches, the tradeoff between computational complexity and audio quality is analysed.

Section 2 gives an overview of the MPEG-4 audio standard and its parametric audio coding tools. The current HILN reference encoder used for the MPEG-4 verification test is described in Section 3. Section 4 presents the different approaches to speed up the HILN encoding process. The computational complexity of these encoders as well as their subjective audio quality is discussed in Section 5 and conclusions are drawn in Section 6.

2 AUDIO CODING IN MPEG-4

In audio coding, a variety of source models can be utilised in combination with appropriate models of the human perception to reduce the redundancy and irrelevance contained in the input signal. Fig. 1 shows the general block diagram of an audio encoder and decoder resulting from these considerations.

The choice of the optimal source and perception models depends on the desired target bitrates as well as

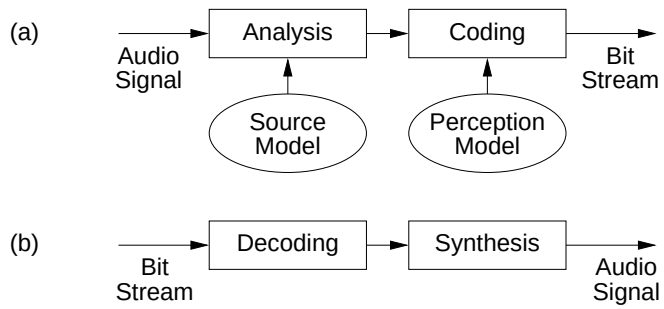

Fig. 1. Block diagram of an audio encoder (a) and decoder (b).

on the audio material to be coded. For coding of arbitrary audio signals, it is common to use the general assumption of a short-time stationary signal as source model together with a highly optimised perception model exploiting the masking effects in the human auditory system. Because of the time-to-frequency transform (e.g. MDCT) used to exploit the source model, such a general audio coder can be referred to as "transform coder" [1]. On the other hand, specialised speech coders make use of a source model derived from the speech generation process in the human vocal tract together with a simple perception model to obtain higher coding efficiency for speech signals than possible with a general audio coder [2]. For coding at very low bitrates, so-called parametric speech and audio coding techniques can provide better coding efficiency than the both approaches described above [3, 4, 5], especially for signals with low complexity content like clean speech or single instruments.

To provide optimal coding efficiency for all types of audio material over a large range of bitrates, the MPEG-4 Audio Standard described in the following Subsection combines all the coding techniques mentioned above.

2.1 The MPEG-4 Audio Standard

The MPEG-4 Audio Standard provides tools for coding of natural and synthetic audio objects and composition of such objects into an "audio scene" [6, 7, 8]. Natural audio objects (such as speech and music) can be coded at bitrates ranging from 2 kbit/s to 64 kbit/s and above using parametric speech coding (HVXC), CELP-based speech coding, parametric audio coding (HILN) or transform-based general audio coding (AAC, TwinVQ). The natural audio and speech coding tools support bitrate scalability, also known as embedded coding. In addition, the parametric coding tools also provide speed and pitch change functionality in the decoder. Synthetic audio objects can be represented using a Text-To-Speech Interface (TTSI) or the Structured Audio (SA) synthesis tools. The SA tools are also used to add effects – like reverberation – and mix different audio objects to compose the final "audio scene" that is presented to the listener.

The first version of the MPEG-4 Standard [9, 10], finalised in October 1998, provides a first set of natural audio coding tools, namely HVXC and CELP speech coding [11] and transform-based general audio coding [12]. With Version 2 [13, 14], finalised in December 1999, additional tools are added to MPEG-4 to provide new functionalities not available in MPEG-4 Version 1. These are tools for fine-step scalability for the general audio coding tools (BSAC), tools for low-delay general audio coding (Low-Delay AAC), tools

for very low bitrate parametric audio coding (HILN – “Harmonic and Individual Lines plus Noise”), and techniques to improve the error-robustness of all MPEG-4 audio and speech coding tools [15].

Fig. 2 shows the block diagram of a complete MPEG-4 Audio decoder. It includes the decoding tools for the audio objects defined in the Audio part of the MPEG-4 Standard [9, 13] as well as bitstream demultiplexing and scene composition defined the Systems part [10, 14].

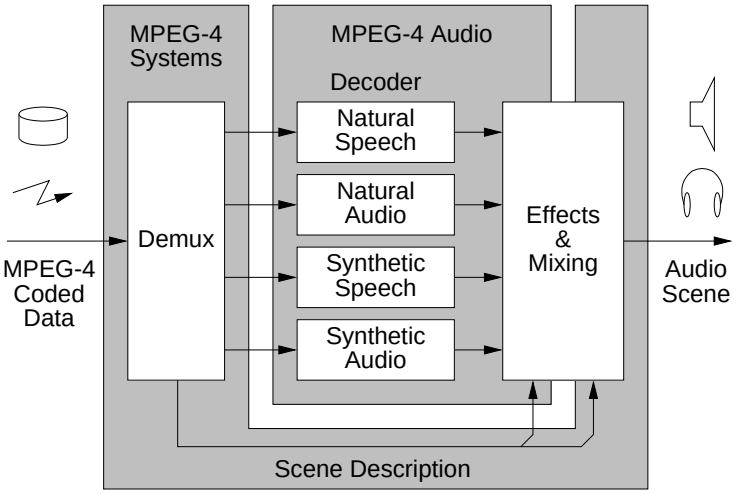

Fig. 2. Block diagram of a complete MPEG-4 Audio decoder.

2.2 HILN – MPEG-4 Parametric Audio Coding

The main focus of this paper is on the HILN tools. While a first version of these tools [16] was scheduled for Version 1 of MPEG-4, it was decided in autumn 1998 to move the HILN tools to Version 2 of MPEG-4 to permit further advances in coding efficiency and flexibility [17]. In this process, various aspects of the HILN tools were significantly improved [18, 19, 20, 21]. Thus significantly better subjective audio quality and bitrate scalability is provided now.

The next two Subsections give a more detailed overview of the HILN encoding and decoding process as specified in Version 2 of the MPEG-4 Audio standard. This is followed by a summary of the test results for HILN from the MPEG verification test [22].

2.3 The HILN Parametric Audio Encoder

The parametric audio coding tools “Harmonic and Individual Lines plus Noise” (HILN) as defined in Version 2 of the MPEG-4 Audio standard [13] permit coding of general audio signals at bitrates of about 4 kbit/s and above using a parametric representation of the audio signal [16, 18, 23]. 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.

It should be noted that only the bitstream format and the decoding process are defined in the normative part of the MPEG standard, while an example for a possible encoding process is given in an informative annex. This permits to optimise the encoding process and/or adapt it to special requirements – like low complexity – even after finalisation of the standard.

Fig. 3 shows the block diagram of a typical HILN parametric audio encoder as used during the development of the MPEG standard. First, one frame of the input signal is decomposed into different components and 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.

Sinusoidal components that "live" for more than one frame are handled as sinusoidal trajectories to ensure phase continuity at frame boundaries. Although the phase parameters of sinusoidal components are estimated as well, they are usually not conveyed in the bitstream. The modelling of transient signals is improved by optional parameters describing the component's amplitude envelope. To describe the spectral envelope of the harmonic tone and the noise signal, LPC modelling as well known from the Linear Predictive Coding of speech signals is employed. The frequency response of an all-pole LPC synthesis filter is used as spectral envelope, and the number of LPC parameters used to describe the spectral envelope is adapted to provide the desired level of spectral details.

As shown in Fig. 3, the signal decomposition and parameter estimation can be implemented as a three step process:

- First all sinusoidal components are extracted from the current frame of the input signal.
- If several sinusoids share a common fundamental frequency, they are grouped as a single *harmonic tone* to permit efficient coding, while all remaining sinusoids are handled as *individual sinusoids*.
- After extraction of all sinusoidal components, the magnitude spectrum of residual signal is used to find the parameters of the *noise component*.

Due to the very low target bitrates of typically 6 to 16 kbit/s, only the parameters for a small number of components can be transmitted. Therefore a perception model is employed to select those components that are most important for the perceptual quality of the signal.

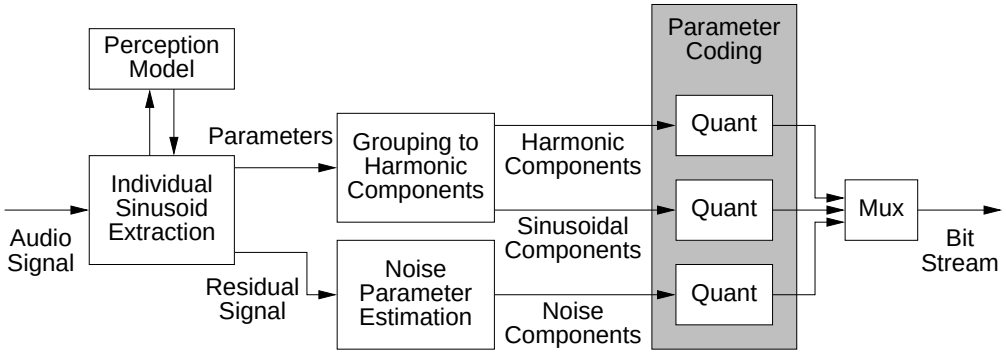

Fig. 3. Block diagram of an HILN parametric audio encoder.

The components' parameters are finally quantised, coded, and multiplexed to form a bitstream. Non-uniform quantisation is used for the parameters to take into account perceptual properties like the "just noticeable differences" of amplitude and frequency. In HILN amplitudes are typically quantised on a logarithmic scale with 1.5 dB step size and frequencies are typically quantised on a Bark scale with $\frac{1}{32}$ Bark step size (i.e. ≈ 3 Hz below 500 Hz and ≈ 10 cent above 500 Hz). Intra- and inter-frame prediction of parameters in combination with advanced entropy conding schemes is used to code the quantised parameters in order to obtain high coding efficiency.

Bitrate scalability, also known as embedded coding, is accomplished by transmitting the parameters of the perceptually most important components in a so-called base bitstream. Additional enhancement bitstreams then convey the parameters of further signal components to obtain a refined description of the audio signal.

2.4 The HILN Parametric Audio Decoder

The block diagram of the HILN parametric audio decoder is shown in Fig. 4. First the parameters of the components are decoded and then the component signals are re-synthesised according to the transmitted parameters. By combining these signals, the output signal of the HILN decoder is obtained. Because of the low phase sensitivity of the human ear, phase information for sinusoids is usually not transmitted, while phase continuity of sinusoidal trajectories is provided by the synthesis tool.

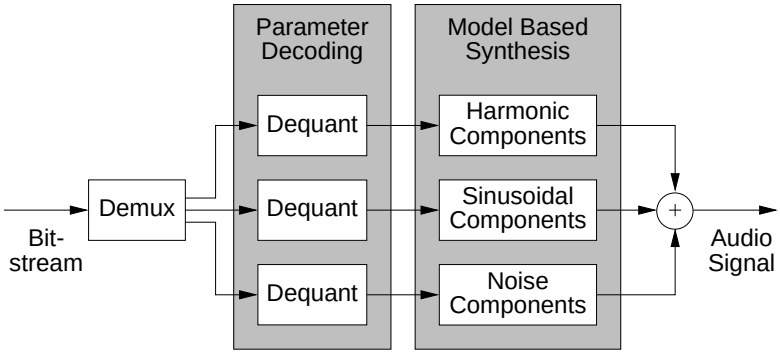

Fig. 4. Block diagram of the HILN parametric audio decoder.

A very interesting property of this parametric coding scheme arises from the fact that the signal is described in terms of frequency and amplitude parameters. This signal representation permits speed and pitch change functionality by simple parameter modification in the decoder without additional complexity.

2.5 MPEG-4 HILN Verification Test Results

The MPEG-4 Audio Version 2 coding tools have recently undergone a verification test [22]. As part of this listening test, the subjective audio quality of the HILN coding tools was compared to the transform-based coding tools TwinVQ and AAC available in MPEG-4 Version 1 at bitrates of 6 kbit/s and 16 kbit/s using

critical audio signals. The HILN tools were operated at 6 and 16 kbit/s (non-scalable) and in a scalable configuration using a 6 kbit/s base and a 10 kbit/s enhancement layer. The BS.1284 quality scale with the 8 kHz bandwidth original signal as reference was used for grading. The overall test results for all 7 audio items and 16 listeners are shown in Fig. 5.

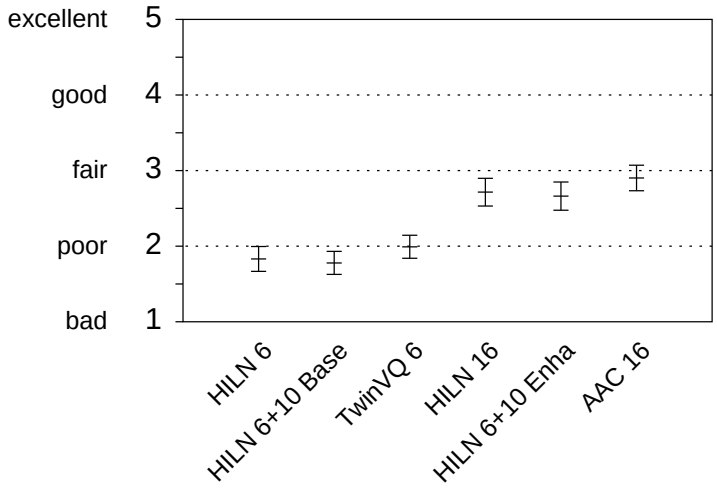

Fig. 5. MPEG-4 Version 2 verification test results for HILN at 6 and 16 kbit/s showing overall mean grading and 95% confidence interval for 7 items and 16 listeners (from [22]).

From the test results, it can be concluded that HILN at both 6 and 16 kbit/s shows performance comparable to other MPEG-4 coding technology operating at similar bitrates while providing the additional capability of independent speed and pitch change in the decoder. The bitrate scalability feature of HILN did not show a penalty in quality compared to non-scalable HILN.

A detailed analysis of different items assessed in this test shows that HILN tends to perform better than the transform coders TwinVQ and AAC for signals with low complexity content. On the other hand, the transform coders tend to perform better for very complex sound mixtures. For speech signals however, the CELP speech coder outperforms both HILN and the transform coders. This indicates that adaptive selection of the coding scheme according to the characteristics of the input signal – as possible with MPEG-4 – can provide an improved overall performance.

3 REFERENCE ENCODER FOR VERIFICATION TESTS

During the development phase of the MPEG-4 standard an encoder was implemented, which was more focused on compression efficiency than on computational efficiency. This implementation also was used in the MPEG-4 verification testing procedure, which included thorough subjective evaluations. Therefore it is used here as a reference for all further development.

The encoding is performed on overlapped frames of input samples. A typical frame length (hop size) for signals sampled at $f_s = 16$ kHz is $T = 32$ ms. The encoding process consists of several steps:

- windowing,
- estimation of parameters for an amplitude envelope,
- parameter estimation and component extraction for individual sinusoids,
- grouping of individual sinusoids to harmonic components,
- spectral modelling for the residual as a noise component,
- parameter encoding for all components.

These steps are explained in more detail in the following Subsections.

3.1 Windowing

Normally the input signal is processed in frames obtained using a window function with 50% overlap, i.e. the window of each frame reaches up to the centres of both neighbouring frames. However, if the energy of signal components in the overlap regions outside the actual frame in relation to the energy within the frame would exceed a given threshold, a window with shorter overlap is selected. This way the influence of strong transients on neighbouring frames is reduced.

3.2 Estimation of Amplitude Envelope Parameters

For sinusoidal modelling of signal segments containing strong amplitude variations, amplitude envelope parameters are estimated for the whole input signal prior to the individual sinusoid parameter estimation. Since later for each sinusoid the decision on the use of the envelope can be taken individually, the envelope estimation should be focused on the strongest variation within a frame. For this purpose the procedure shown in Fig. 6 was developed.

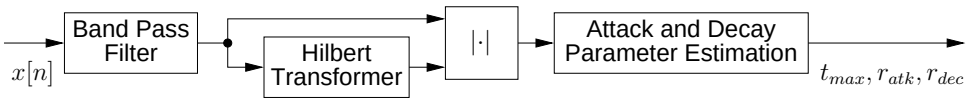

Fig. 6. Block diagram of the envelope parameter estimator.

First the signal is fed through a band pass filter which attenuates signal components at the upper and lower band edges. Then an FIR Hilbert transformer generates a signal approximating the imaginary part of an analytic signal corresponding to the positive frequency components of the filter output signal. The magnitude calculated from the output signals of band pass filter and Hilbert transformer then gives an envelope function. This envelope function is normalised with respect to its maximum and then approximated by a triangular shape. The approximation is specified by parameters for the time of its maximum t_{max} and the angles for left slope r_{atk} (attack rate) and right slope r_{dec} (decay rate). These parameters are obtained using regression techniques with appropriate weighting depending on amplitude and distance from the maximum. To give an example, Fig. 7 shows the original envelope and the approximated envelope as specified by the obtained envelope parameters for one frame of signal “castanets.”

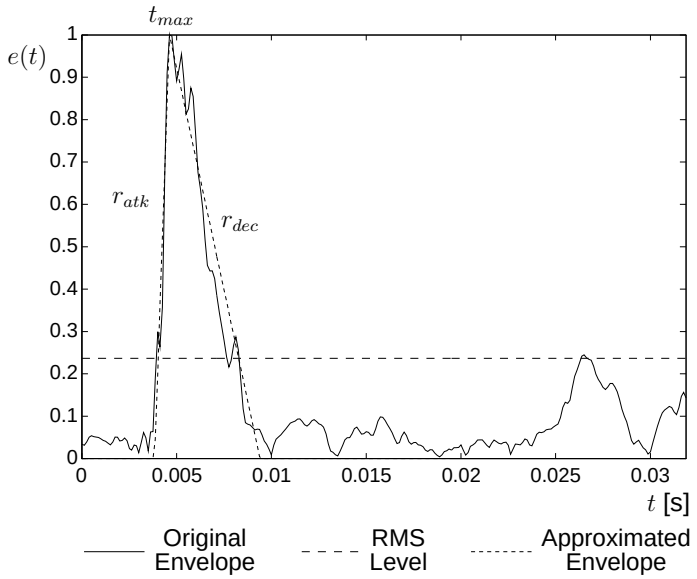

Fig. 7. Normalised amplitude envelope $e(t)$ and envelope parameters t_{max}, r_{atk}, and r_{dec} for one frame of signal "castanets."

3.3 Parameter Estimation for Individual Sinusoids

The parameters of individual sinusoids are extracted iteratively in an analysis/synthesis loop as shown in Fig. 8. This is the most critical procedure with respect to computational complexity. In order to illustrate the required operations, in the following the major processing steps are described in more detail.

Prior to the start of the iterative procedure the input signal magnitude spectrum $|X(f)|$ is calculated with an FFT. For the following description of the processing inside the loop it is assumed that in the i-th iteration $i-1$ sinusoids already have been processed and a signal $s_{i-1}(t)$ containing all of them has been synthesised. Then the magnitude spectrum $|S_{i-1}(f)|$ of the synthesised signal is calculated with an FFT. By subtraction, limitation to positive values, and squaring, a power spectrum is calculated which indicates how much the signal spectrum exceeds the synthesised spectrum:

$$\tilde{E}_i(f) = \left(\max(0, |X(f)| - |S_{i-1}(f)|)\right)^2 . \tag{1}$$

The next step is the detection of the maximum ratio of $\tilde{E}_i(f)$ over an estimated masked threshold $M_{i-1}(f)$ which would be generated by the synthesised signal $s_{i-1}(t)$. The corresponding frequency is used as a coarse frequency estimate $f_{c,i}$ of the i-th sinusoid. Its accuracy corresponds to the FFT frequency resolution.

In order to obtain a higher frequency resolution and better estimates for sinusoids with varying frequencies and amplitudes, a high accuracy parameter estimation is performed on the residual $r_i(t) = x(t) - s_{i-1}(t)$ from previous iterations in a frequency interval surrounding $f_{c,i}$. The structure of the phase regression based frequency estimator is shown in Fig. 9. The surrounding of $f_{c,i}$ is obtained from a complex band pass filter

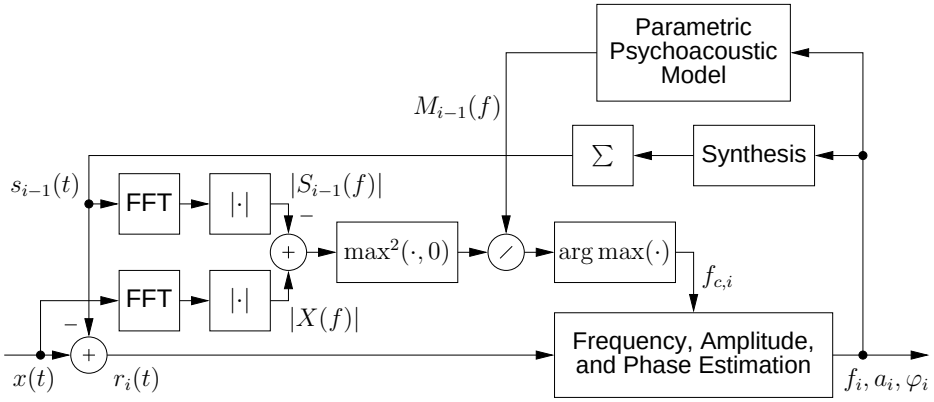

Fig. 8. Analysis/synthesis loop for extraction and parameter estimation for individual sinusoids.

which is implemented by complex modulation, i.e. frequency shift by $-f_{c,i}$, and low pass filtering. Due to the band limitation the sampling rate can be reduced significantly before phase calculation and regression. In practice the filtering and downsampling was implemented by multiplying shifted versions of the window functions and accumulation. A typical value for the number of input values for the regression is 17, corresponding to $D = 32$ for a typical frame size of $N = Tf_s = 512$ samples. Linear regression on the phase values yielded in a parameter for a constant frequency difference to $f_{c,i}$, while a quadratic regression additionally gives the parameters for a linearly changing frequency. Both can be described by the frequency offsets $\Delta f_{l,i}$ and $\Delta f_{r,i}$ at the left and right boundary of the current frame. In the case of constant frequency obviously both parameters are equal. Finally the absolute frequency parameters $f_{l,i}$ and $f_{r,i}$ are obtained by adding $f_{c,i}$. An example of the second order phase regression is given in Fig. 10 for a synthetic sweep with additive noise (SNR $= 20$dB).

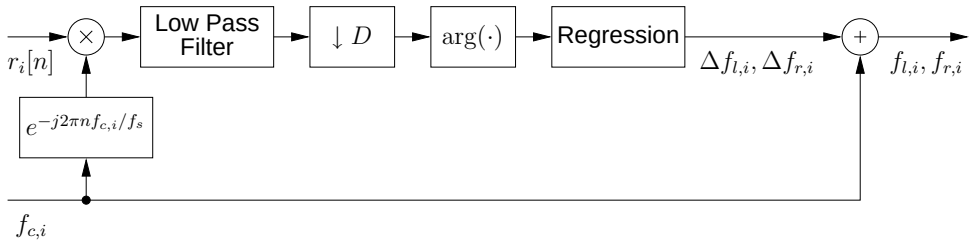

Fig. 9. High accuracy frequency estimator in the surrounding of $f_{c,i}$.

Once the frequency parameters are obtained, amplitude and phase are estimated. For this purpose complex sinusoids with constant frequency and with linearly changing frequency are generated. Amplitudes and phases are then determined as magnitudes and phases of complex correlation coefficients between the complex sinusoids and the residual $r_i(t)$. The same procedure is applied to the complex sinusoids additionally multiplied by an envelope function generated from the parameters described above. Therefore a total of 4 correlation operations are performed and the best match of the following combinations is selected:

- constant frequency and constant amplitude

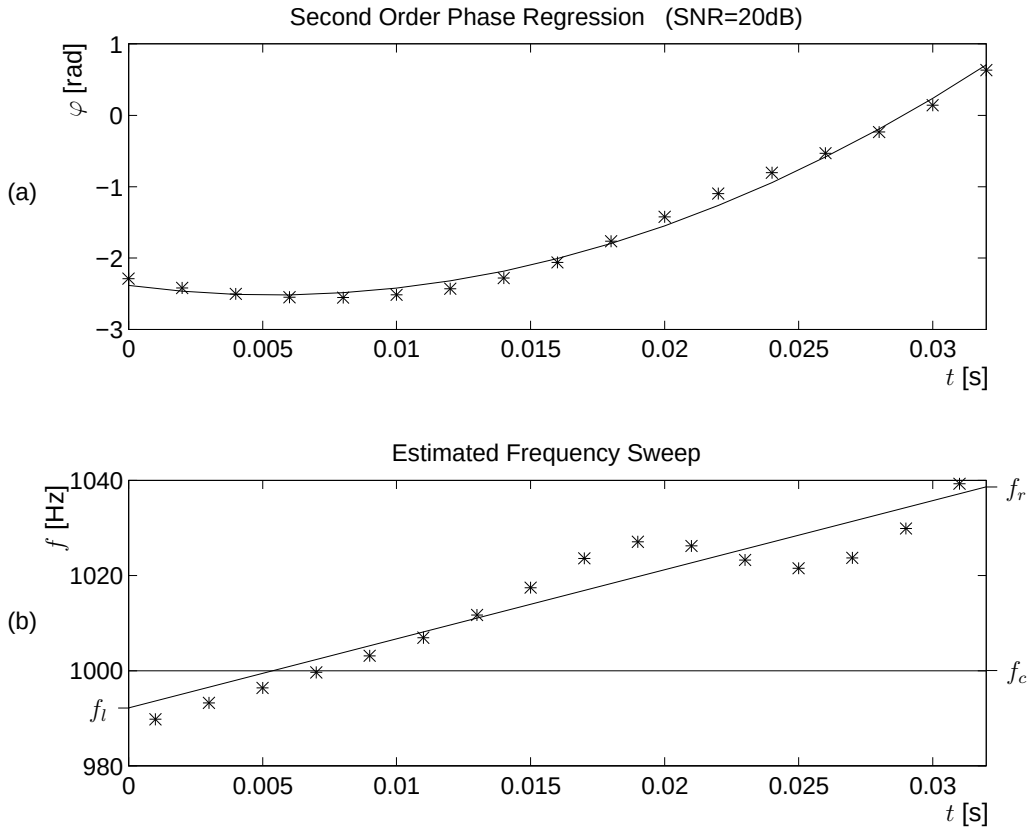

Fig. 10. Second order phase regression (a) for estimation of a linearly changing frequency (b) within a frame of $T = 32$ ms for a synthetic sweep with additive noise (SNR $= 20$dB).

- linearly changing frequency and constant amplitude
- constant frequency and amplitude envelope
- linearly changing frequency and amplitude envelope.

After the parameter estimation the next iteration is to be prepared by adding the new component to the synthesised signal and by updating the masked threshold correspondingly. This can be done quite efficiently using a parametric psychoacoustic model which makes direct use of the frequency and amplitude parameters and thus avoids an additional spectral analysis.

3.4 Grouping to Harmonic Components

The parameter estimation for harmonic components can be done efficiently on the basis of previously extracted individual sinusoids by searching patterns of frequencies which are multiples of a common fundamental frequency. The goal of this grouping is an efficient representation using only the fundamental frequency and the amplitudes of the partials.

However, if the fundamental frequency is relatively low, the total number of sinusoids – i.e. harmonic and individual lines – that can be transmitted at a given bitrate may be significantly higher than without harmonic components. Therefore the analysis/synthesis loop described above must be executed more often to provide a sufficient number of parameter sets. This leads to an increase of the total computational complexity.

The LPC parameter based modelling of the spectral envelope for the partials on the other hand does not contribute significantly.

3.5 Noise Component Modelling

The residual from the iteration loop for extracting sinusoids is regarded as a noise component. The estimation of the energy parameter and of the LPC parameters describing the spectral envelope are of relatively low computational complexity. However, the LPC based spectral modelling approach can be heavily influenced by sinusoids not extracted in the loop. Therefore even the noise modelling requires a careful selection of the number of iterations. Especially at very low bitrates the required number of extracted sinusoids often significantly exceeds the number of transmitted harmonic and individual lines.

3.6 Parameter Encoding

Compared to the computational complexity of the individual sinusoid extraction loop the quantisation and coding of the obtained parameters is almost neglectable. Nevertheless a brief overview on the used techniques is given.

For the parameters of the individual sinusoids one of two possible coding modes is selected depending on whether they are regarded to be continued from the previous frame or not. The continuation decision is taken by comparing all start amplitudes and start frequencies to all end amplitudes and end frequencies of the previous frame.

For “new lines” without predecessor the frequency and amplitude parameters are quantised according to non-linear scales optimised with respect to perception. Then they are sorted with ascending frequencies and the frequency parameters are encoded using a “sub-division code” which successively restricts the possible

interval between the previously encoded parameter and the desired signal bandwidth [18]. This algorithm enables very efficient entropy coding with variable length codewords and has low computational and memory requirements. The amplitude parameter coding is also based on the sub-division code, but amplitudes are set in relation to a separately transmitted maximum amplitude value for all signal components in the frame.

For "continued lines" changes of the quantised frequency and amplitude parameters are transmitted using a variable length code, which exploits the fact that small changes occur much more frequently than large changes.

The LPC parameters for the spectral envelopes of harmonic tones and noise components are transmitted as Logarithmic Area Ratios (LAR) with uniform quantisation. The LAR representation permits to easily vary the LPC filter order from frame to frame to adapt to the required detail of spectral modelling. Also here, variable length codewords are used and predictive coding is applied, if there is sufficient correlation to the previous frame.

4 HILN ENCODER OPTIMISATION

As mentioned above, the most complex task in the HILN encoder is the decomposition of the input signal into the perceptually most relevant components and the estimation of the component parameters. Until now, HILN encoder development mainly focused on maximum audio quality to proof the merit of the HILN parametric audio coding tools within the MPEG-4 standardisation process. This, however, has led to a quite high computational complexity of the encoder. Therefore, this reference encoder is unsuitable for applications like "live streaming" which require encoding in real-time.

In order to make real-time HILN encoding feasible, different approaches to speed up HILN encoding are presented in this Section. Simplified algorithms for signal decomposition, for parameter estimation, and for the perception model are utilised to reduce computational complexity of the encoding process.

In the reference encoder, the analysis/synthesis loop for sinusoidal components – which is by far the most complex module – is implemented in the time domain. However, since the sinusoidal signal components are highly localised in frequency, a frequency domain implementation can offer major advantages in terms of computational complexity.

The fast encoder introduced in the following Subsections makes use of such a frequency domain implementation. Estimation of sinusoids with non-constant frequency is not supported since such "sweeps" are not as good localised in frequency, which would require a more complex algorithm again. Also the optional amplitude envelope $e(t)$ leads to a "wider" spectrum of a sinusoidal component. Thus, both a simple version of the fast encoder without envelope support as well as a more complex version that includes envelope estimation is presented. In addition, the psychoacoustic model used to identify the most significant sinusoidal signal components is also simplified and now applied outside the extraction loop.

With these optimisations, real-time encoding on common PCs is easily possible. Section 5 analyses the computational complexity and subjective audio quality of the fast encoder in comparison to the reference encoder.

4.1 Sinusoid Extraction by Matching Pursuit

Matching pursuit refers to an iterative method for computing signal decompositions in terms of a linear combination of vectors $\vec{g}_m$ from a highly redundant dictionary with M elements [24, 25, 26]. It is a "greedy" algorithm in that at each stage i of the iteration the vector $\vec{g}_{m_i}$ in the dictionary is found that best matches the current residual $\vec{r}_i$.

The algorithm is started with a windowed segment $r_1[n] = x[n]w[n]$ of the input signal $x[n]$ as first "residual" $\vec{r}_1$. The window $w[n]$ is used to avoid discontinuities at the frame boundaries. For a frame length (hop size) of N samples, the window function must fulfill

$$\sum_u w^2[uN+n] = 1 \quad \forall\, n \tag{2}$$

in order to provide perfect reconstruction by windowed overlap/add in case of a vanishing residual $\vec{r}_I = \vec{0}$. The same window is applied to the sinusoidal dictionary elements used here

$$g_m[n] = s_m w[n] e^{j2\pi \frac{m}{M} n} \tag{3}$$

where an appropriate scaling factor s_m is used to ensure unity two-norm $||\vec{g}_m|| = 1$ for all $m = 0 \ldots M-1$. In each stage i of the iteration, the optimal dictionary element $\vec{g}_{m_i}$ and the corresponding weight c_i is determined to minimise the norm of the residual

$$||\vec{r}_{i+1}|| \rightarrow \min \tag{4}$$

for the next stage. Since $x[n]$ and thus $\vec{r}_i$ have real values, the next residual can be calculated as follows:

$$\vec{r}_{i+1} = \vec{r}_i - c_i \vec{g}_{m_i} - c_i^* \vec{g}_{m_i}^* \tag{5}$$

For a given m the optimal real and imaginary component of $c = a + jb$ according to Eq. 4 can be found by setting the partial derivatives of $||\vec{r}_{i+1}||$ with respect to a and b to 0. This leads to the following equation system:

$$\begin{bmatrix} \mathrm{Re}(\vec{g}_m)\mathrm{Re}(\vec{g}_m) & \mathrm{Re}(\vec{g}_m)\mathrm{Im}(\vec{g}_m) \\ \mathrm{Re}(\vec{g}_m)\mathrm{Im}(\vec{g}_m) & \mathrm{Im}(\vec{g}_m)\mathrm{Im}(\vec{g}_m) \end{bmatrix} \begin{bmatrix} 2a \\ -2b \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(\vec{g}_m)\vec{r}_i \\ \mathrm{Im}(\vec{g}_m)\vec{r}_i \end{bmatrix} \tag{6}$$

At each stage i of the iteration m_i and its appropriate c_i are chosen so that $||\vec{r}_{i+1}||$ is minimal, which corresponds to $|c_i|$ being maximal.

If a symmetric window $w[n] = w[-n]$ is used, $\mathrm{Re}(\vec{g}_m)$ and $\mathrm{Im}(\vec{g}_m)$ become orthogonal so that Eq. 6 can be simplified:

$$2a = \frac{\mathrm{Re}(\vec{g}_m)\vec{r}_i}{\mathrm{Re}(\vec{g}_m)\mathrm{Re}(\vec{g}_m)} \tag{7}$$

$$-2b = \frac{\mathrm{Im}(\vec{g}_m)\vec{r}_i}{\mathrm{Im}(\vec{g}_m)\mathrm{Im}(\vec{g}_m)} \tag{8}$$

4.2 Sinusoid Extraction in the Frequency Domain

Although the matching pursuit algorithm was described in the time domain in the previous Subsection, it can also be implemented in the frequency domain [25]. Instead of the real signal $x[n]$, its corresponding analytic signal is considered now to confine the further processing to non-negative frequencies only. In the frequency domain, the inner products in Eq. 6 become correlations. $G_0[l]$ denotes the discrete Fourier transform of $g_0[n] = s_0 w[n]$ zero-padded for the transform length M. Since the frequency domain representations of the other dictionary elements $G_m[l] = G_0[l-m]$ can be derived by simple displacement on the frequency axis, $G_0[l]$ is also referred to as "prototype."

In the encoder presented here, a window function $w[n]$, $n = -N, \ldots, N-1$ as shown in Fig. 11 is utilised for a frame length of $N = Tf_s$.

$$w^2(t) = \begin{cases} \cos^2(\pi \frac{t^2}{T^2}) & |t| \leq \frac{T}{2} \\ 1 - w^2(T - |t|) & \frac{T}{2} < |t| < T \\ 0 & T \leq |t| \end{cases} \tag{9}$$

It fulfills Eq. 2, has a better localisation in time than the common Hanning window and is smooth to get a compact representation in the frequency domain, as shown in Fig. 12. Fig. 13 shows the basic structure of the individual sinusoid extraction by matching pursuit in the frequency domain.

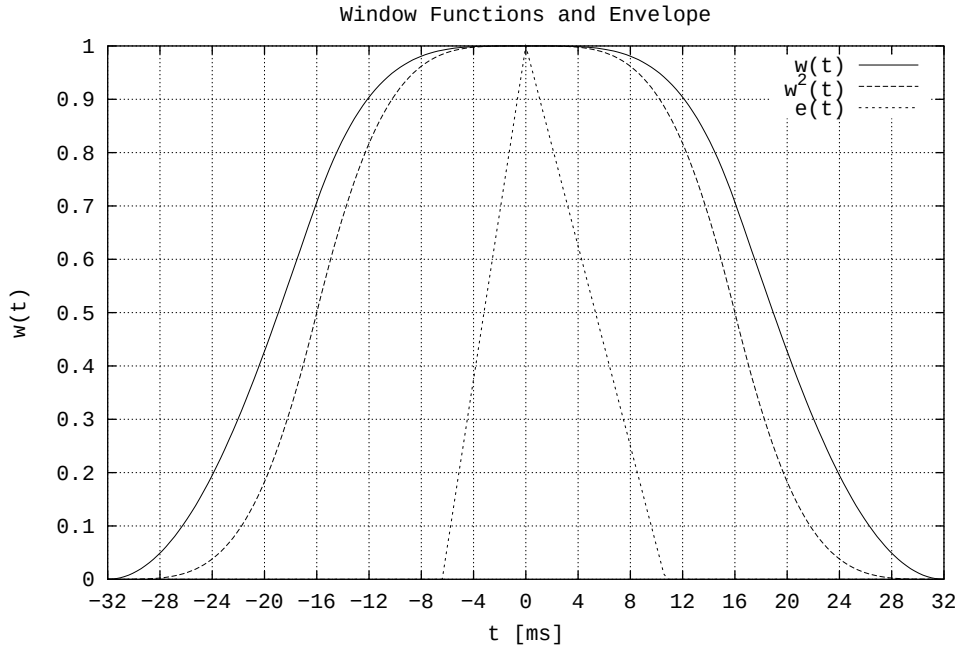

Fig. 11. Window function $w(t)$ used in the fast encoder and an example of an amplitude envelope $e(t)$ (frame length $T = 32$ ms).

The calculations performed in the algorithm shown in Fig. 13 can be greatly simplified if a small error is permitted. In the frequency domain the energy of the prototype $G_0(f)$ concentrates around $f = 0$, as can

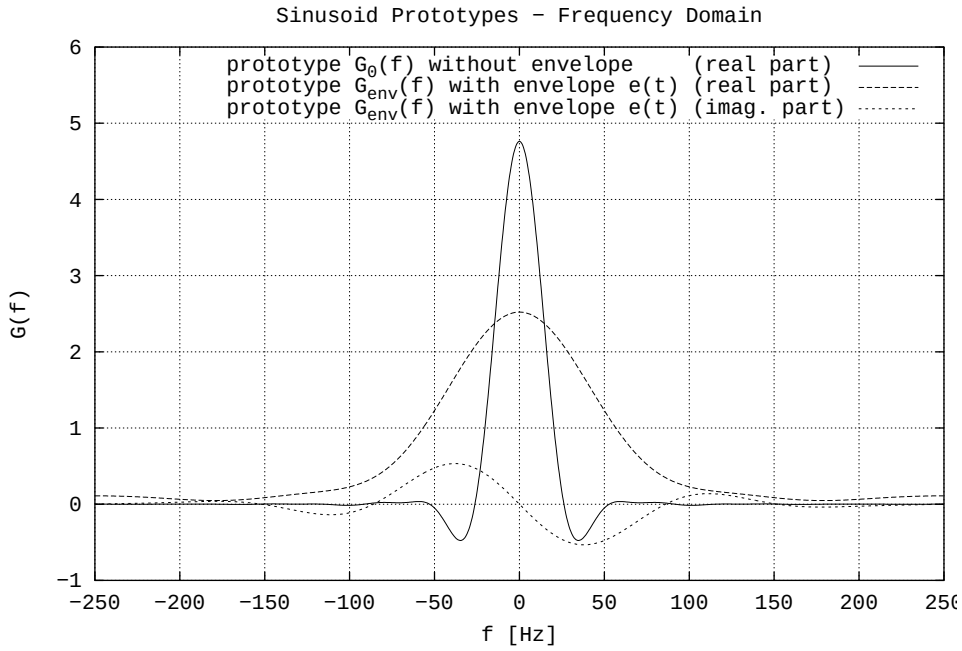

Fig. 12. Prototype $G_0(f) = W(f)$ for sinusoidal matching pursuit in the frequency domain and prototype $G_{env}(f)$ including the amplitude envelope $e(t)$ as shown in Fig. 11.

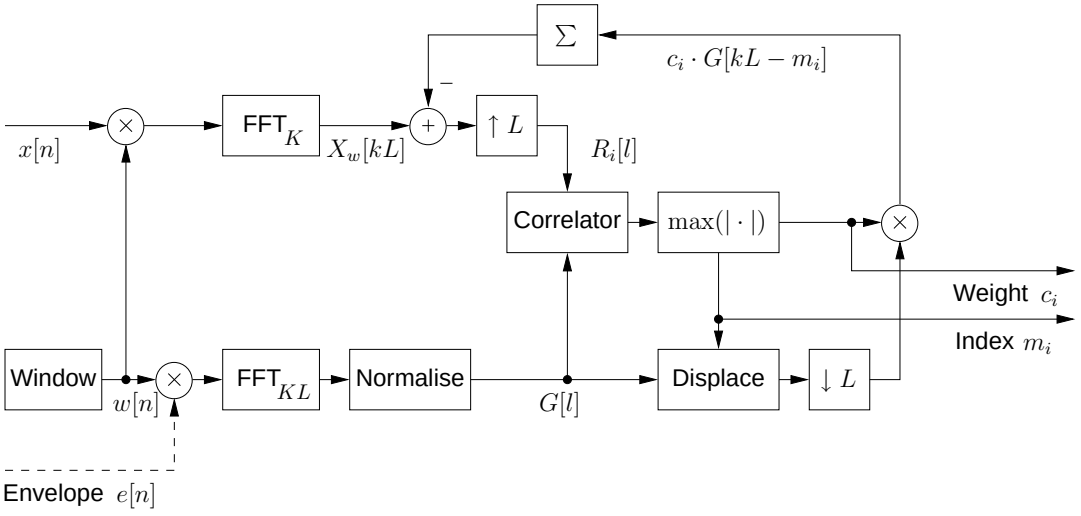

Fig. 13. Simplified block diagram of fast individual sinusoid extraction in the frequency domain.

be seen from Fig. 12. Therefore only a small section of the prototype is stored. This reduces the complexity of the correlation and calculation of the residual signal significantly. To further reduce the computational complexity of the correlation, a three step refinement procedure is used to find the optimal m_i. In the first step the correlation is done with an extremely cropped prototype on a coarse grid – the grid of the original FFT with transform length $K \geq 2N$. In the second step, the correlation is calculated only in the surrounding of the maximum found by the first step now using longer prototypes and a finer grid. The third step provides a further refinement in the same way and delivers m_i on a frequency grid which is $L = 8$ times as fine as the grid of the original FFT. Thus a frequency resolution typically better than 2 Hz is achieved. To efficiently find the maximum in the first step, the initial power spectrum $X_w[kL]$ is split into short segments. For each segment its maximum and the corresponding index is calculated and stored. To find the overall maximum only the segment-maxima have to be scanned. After removing a synthesized signal from the spectrum, only the segments that intersect with the prototype have to be recalculated.

In addition to subtracting the synthesized signal from the residual as shown in Eq. 5, a notch filter $H_{notch}(f - f_i)$ is applied at f_i to obtain the new residual signal in the frequency domain $R_{i+1}(f)$. Thus components of the sinusoid which were not subtracted because of the simple parameter estimation are removed as well. Omitting the notch filter results in an undesirable high energy of the residual signal. The notch filter used here is defined in the frequency domain by the following equation:

$$H_{notch}(f) = 1 - e^{-f^2/f_0^2} \tag{10}$$

If the frequency f_0 (i.e. bandwidth) is too small, the residual signal is too loud. If f_0 is too large, other sinusoidal components may be removed. Fig. 14 shows the frequency response $H_{notch}(f)$ for $f_0 = 50$ Hz as used here.

4.3 Fast HILN Encoder using Frequency Domain Implementation

The general block diagram of the fast HILN encoder using the frequency domain implementation of the sinusoidal matching pursuit presented in the previous Subsection is shown in Fig. 15.

The residual signal is used to calculate the noise parameters in the same way as in the reference encoder. However, since the residual signal is only available in the frequency domain, an inverse Fourier transform is required to convert it back in to the time domain to estimate the LPC noise parameters.

Harmonic components are not utilised in this fast encoder since rather advanced and complex algorithms would be needed to prevent degradations of the subjective signal quality in case of inappropriate grouping of sinusoids.

The fast individual sinusoid extraction uses a simple energy metric $||\vec{r}|| \rightarrow \min$ and does not utilise any psychoacoustic model. Instead, a very simple parametric psychoacoustic model is used to re-order the extracted individual sinusoids according to their perceptual relevance based on their mutual masking effects.

The simple version of the fast HILN encoder does not use optional amplitude envelopes. However, a second, more complex version of the fast HILN encoder permits utilisation of amplitude envelope parameters. To generate these parameters, the envelope of the input audio signal is estimated and a second prototype $G_{env}(f)$ using this envelope $e(t)$ is calculated for every frame as shown in Fig. 11 and Fig. 12. The correlation is now done for both prototypes $G_0(f)$ and $G_{env}(f)$, and the one leading to lower residual energy

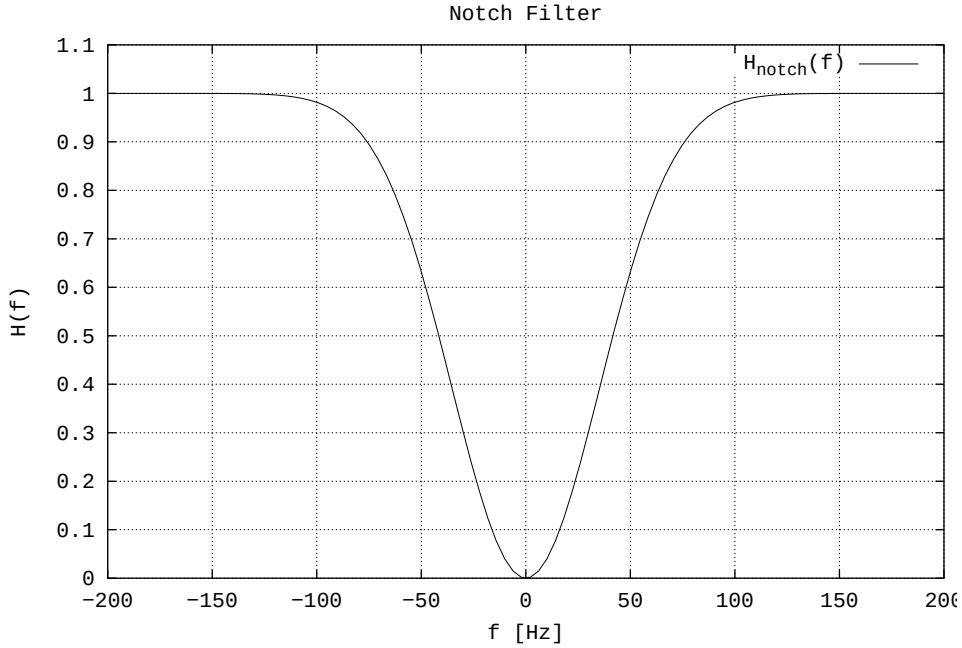

Fig. 14. Frequency response of the notch filter $H_{notch}(f)$.

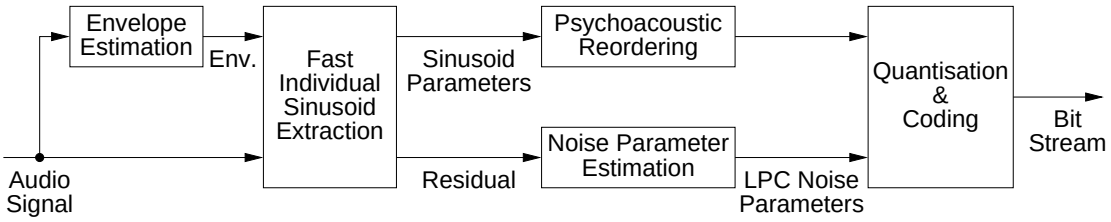

Fig. 15. General block diagram of the fast HILN encoder.

is selected. Since the amplitude envelope leads to an increased bandwidth of the prototype, the correlation described in Subsection 4.2 needs to be carried out for a larger part of the prototype.

5 RESULTS

In this Section, the tradeoff between computational complexity and audio quality is analysed for the three encoders discussed in this paper:

- reference encoder as used for the MPEG-4 verification tests,
- fast encoder as presented in Section 4 without envelope estimation,
- fast encoder as above but with envelope estimation.

All three encoders are fully implemented in ANSI C. The average CPU load required for real-time HILN encoding of an audio signal sampled at 16 kHz with a bitrate of 6 or 16 kbit/s was measured on different workstations equipped with Pentium, SPARC, and Alpha CPUs. The CPU load figures in MHz are given in Table 1 for the three different encoders. They indicate that real-time decoding is easily possible with todays PCs using the fast HILN encoder presented here. The computational complexity of an encoder generating a 16 kbit/s bitstream is about twice as high as for 6 kbit/s. It can also be seen that enabling the envelope estimation in the fast encoder roughly doubles the computational complexity.

Encoder	Bitrate	Pentium III		UltraSPARC		Alpha 21264	
	[kbit/s]	CPU Load [MHz]	Rel. Speed	CPU Load [MHz]	Rel. Speed	CPU Load [MHz]	Rel. Speed
Reference Encoder	6	26000	1	23600	1	11600	1
Fast Encoder	6	24	1080	25	950	13	870
Fast Encoder (Env.)	6	51	510	51	460	24	490
Reference Encoder	16	31000	1	28700	1	14100	1
Fast Encoder	16	46	680	46	620	27	530
Fast Encoder (Env.)	16	100	310	105	270	48	290

Table 1. Encoding speed of reference encoder, fast encoder, and fast encoder with envelope estimation for audio signals sampled at $f_s = 16$ kHz measured on Intel Pentium III (500 MHz), SUN UltraSPARC (168 MHz), and Alpha 21264 (666 MHz).

A more detailed analysis of the fast HILN encoder (without envelopes) as presented here was carried out on a 500 MHz Pentium III using the CPUs internal clock cycle counter. Table 2 shows relative execution time for the major modules of the fast encoder without envelope estimation. In Table 3 a refined execution time analysis of the sinusoid extraction module is given. These figures show that the computational complexity of the fast encoder is well balanced between its different modules.

The average CPU load required for real-time HILN decoding of the bitstreams generated by the different encoders as shown in Table 1 also was measured on the same workstations. The CPU load figures in MHz are given in Table 4. They indicate that real-time decoding is easily possible with todays PCs. Typically the decoder complexity lies between the complexity of the fast encoder without envelopes and the complexity of the fast encoder with envelopes. The figures show also that the CPU load for decoding a 16 kbit/s bitstream

Programme Module	Clock Cycles	Rel. Execution Time
Transform	101670	14.54 %
Sinusoid Extraction	250260	35.78 %
Inverse Transform	156020	22.31 %
Find Continue Relations	31040	4.44 %
Psychoacoustic Reordering	51810	7.41 %
Noise Parameter Calculation	86040	12.30 %
Coding	22540	3.22 %
Sum	699380	100.00 %

Table 2. Clock cycles counts per frame for the major programme modules of the fast encoder (without envelopes) encoding at 6 kbit/s measured on Intel Pentium III (500 MHz) ($f_s = 16$ kHz, $T = 32$ ms, extracting $I = 40$ sinusoids, 16 LPC noise parameters).

Programme Module	Invocations / Frame	Clock Cycles	Rel. Execution Time
Power Spectrum Initialisation	1	33100	4.73 %
Power Spectrum Update (1st Step)	40	1055	6.03 %
Search Maximum (1st Step)	40	1267	7.25 %
Fine Freq. Estim. (2nd & 3rd Step)	40	2760	15.79 %
Quantisation	40	347	1.98 %
Sum (Sinusoid Extraction)		250260	35.78 %

Table 3. Detailed clock cycle counts for the sinusoid extraction module in Table 2.

is somewhat more than twice as high as for a 6 kbit/s bitstream due to the higher number of sinusoids to be synthesised. Bitstreams encoded with the reference encoder can include a harmonic tone component and thus require a slightly higher decoder CPU load than bitstreams generated with the fast encoder. However, an HILN decoder anyway must be able to decode bitstreams up to a certain complexity to be compliant to the MPEG standard.

It should be noted that the official MPEG-4 reference software decoder [27] (implemented in ANSI C) was used for these measurements. This decoder is only partly optimised for low computational efficiency. Therefore a fully optimised HILN decoder is expected to have significantly lower CPU load.

Bitstream generated by:	Bitrate [kbit/s]	Pentium III CPU Load [MHz]	UltraSPARC CPU Load [MHz]	Alpha 21264 CPU Load [MHz]
Reference Encoder	6	36	32	26
Fast Encoder	6	35	32	25
Fast Encoder (Env.)	6	32	32	23
Reference Encoder	16	81	63	60
Fast Encoder	16	63	54	51
Fast Encoder (Env.)	16	66	54	48

Table 4. Decoding speed of reference decoder for the bitstreams generated by different encoders as shown in Table 1 measured on Intel Pentium III (500 MHz), SUN UltraSPARC (168 MHz), and Alpha 21264 (666 MHz).

To assess the subjective quality of the fast HILN encoder with and without envelope estimation in comparison to the reference encoder used in the MPEG-4 Audio Version 2 verification tests [22], an informal listening test was carried out by the authors. The same set of 39 items (mono, 16 kHz sampling rate) as used in the verification test was encoded with both versions of the fast encoder at bitrates of 6 and 16 kbit/s. This test set has a total duration of about 11 minutes and covers a broad range of audio material including speech, single instruments, pop, rock, classical music as well as radio programmes.

In this listening test, no difference in subjective quality of reference encoder and fast encoder with envelope estimation was observed for a large majority of the test items at both bitrates. For about 10% to 20% of the items, the fast encoder with envelopes delivered a slightly worse or worse quality compared to the reference.

For most items, the fast encoder without utilisation of amplitude envelopes delivered quality similar to that of the fast encoder with envelope. However, for percussive signals, the lack of amplitude envelope parameters resulted in a clearly worse subjective quality compared to the fast encoder with envelopes.

Some of the factors responsible for the lower subjective quality of the fast encoder compared to the reference encoder are:

- lack of sweep estimation,
- envelope estimation not as good as in reference encoder,
- much simpler psychoacoustic model,
- suboptimal sinusoid extraction leaving too much residual signal thus causing unnatural high level of the noise component.

For a very few items, it was observed that the fast encoder with envelopes provided a slightly better quality that the reference encoder. This can be ascribed to a sometimes non-optimal estimation of harmonic tone parameters in the reference encoder which can result in characteristic artifacts. The fast encoder on the other hand does not use a harmonic tone component.

6 CONCLUSIONS

In this paper, fast implementations of an MPEG-4 HILN parametric audio encoder in ANSI C were presented. They easily permit real-time HILN encoding on todays PCs while providing a subjective audio quality comparable to that of the HILN reference encoder operating at the same bitrate for most of the audio material tested. This corresponds to a speed-up by a factor of about 300 to 1000 compared to the reference encoder used for the MPEG verification tests.

Further improvement of the subjective audio quality achieved by the fast encoder – especially for critical items – is expected by optimisation of the envelope estimation, the psychoacoustic model, and other modules.

In addition the fast encoder as well as the reference encoder might be improved using advanced decomposition and parameter estimation techniques currently under development [28, 29, 30]. This includes improved tracking of sinusoidal components, better discrimination between tonal and noise-like components as well as a more detailed modelling and utilisation of perceptual properties of the human auditory system.

REFERENCES

[1] K. Brandenburg and M. Bosi, “Overview of MPEG Audio: Current and Future Standards for Low Bit Rate Audio Coding,” *J. Audio Eng. Soc.*, Vol. 45, No. 1/2, pp. 4–21, Jan./Feb. 1997.

[2] B. Edler, “Speech Coding in MPEG-4,” *Int. J. of Speech Technology*, Vol. 2, No. 4, pp. 289–303, May 1999.

[3] B. Edler and H. Purnhagen, “Parametric Audio Coding,” *Proc. International Conference on Signal Processing (ICSP2000)*, Beijing, August 2000.

[4] H. Purnhagen, “Advances in Parametric Audio Coding,” *Proc. IEEE WASPAA*, Sep. 1999.

[5] R. McAulay and T. Quatieri, “Speech Analysis/Synthesis Based on a Sinusoidal Representation,” *IEEE Trans. ASSP*, Vol. 34, No. 4, pp. 744–754, Aug. 1986.

[6] R. Koenen, *Overview of the MPEG-4 Standard*, ISO/IEC JTC1/SC29/WG11 N3156, Dec. 1999. http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm

[7] *Official MPEG Home Page*. http://www.cselt.it/mpeg/

[8] *MPEG Audio Web Page*. http://www.tnt.uni-hannover.de/project/mpeg/audio/

[9] ISO/IEC 14496-3:1999, *Coding of audio-visual objects – Part 3: Audio (MPEG-4 Audio Version 1)*, ISO/IEC International Standard, 1999.

[10] ISO/IEC 14496-1:1999, *Coding of audio-visual objects – Part 1: Systems (MPEG-4 Systems Version 1)*, ISO/IEC International Standard, 1999.

[11] M. Nishiguchi, "MPEG-4 speech coding," *Proc. AES 17th International Conference*, Sep. 1999.

[12] B. Grill, "The MPEG-4 General Audio Coder," *Proc. AES 17th International Conference*, Sep. 1999.

[13] ISO/IEC 14496-3/AMD1:2000, *Coding of audio-visual objects – Part 3: Audio (MPEG-4 Audio Version 2)*, ISO/IEC International Standard, 2000.

[14] ISO/IEC 14496-1/AMD1:2000, *Coding of audio-visual objects – Part 1: Systems (MPEG-4 Systems Version 2)*, ISO/IEC International Standard, 2000.

[15] H. Purnhagen, "An Overview of MPEG-4 Audio Version 2," *Proc. AES 17th International Conference*, Sep. 1999.

[16] H. Purnhagen, B. Edler, and C. Ferekidis, "Object-Based Analysis/Synthesis Audio Coder for Very Low Bit Rates," *AES 104th Convention*, Preprint 4747, May 1998.

[17] ISO/IEC, *MPEG-4 Audio verification test results: Audio on Internet*, ISO/IEC JTC1/SC29/WG11 N2425, Oct. 1998.
http://www.tnt.uni-hannover.de/project/mpeg/audio/public/w2425.pdf

[18] H. Purnhagen and N. Meine, "HILN - The MPEG-4 Parametric Audio Coding Tools," *Proc. IEEE ISCAS 2000*, May 2000.

[19] H. Purnhagen and N. Meine, *Core Experiment Proposal on Improved Parametric Audio Coding*, ISO/IEC JTC1/SC29/WG11 M4492, Mar. 1999.

[20] N. Meine, *LPC-Spektralmodelle für einen Analyse/Synthese Audio Coder*, Diplomarbeit, Universität Hannover, Jan. 1999.

[21] B. Feiten, R. Schwalbe, and F. Feige, "Dynamically Scalable Audio Internet Transmission," *AES 104th Convention*, Preprint 4686, May 1998.

[22] ISO/IEC, *Report on the MPEG-4 Audio Version 2 Verification Test*, ISO/IEC JTC1/SC29/WG11 N3075, Dec. 1999.
http://www.tnt.uni-hannover.de/project/mpeg/audio/public/w3075.pdf

[23] B. Edler, H. Purnhagen, and C. Ferekidis, "ASAC - Analysis/Synthesis Audio Codec for Very Low Bit Rates," *AES 100th Convention*, Preprint 4179, May 1996.

[24] M. Goodwin, "Matching Pursuit With Damped Sinusoids," *Proc. IEEE ICASSP*, 1997.

[25] T. Verma and T. Meng, “Sinusoidal Modeling using Frame-Based Perceptually Weighted Matching Pursuits,” *Proc. IEEE ICASSP*, 1999.

[26] M. Goodwin, *Adaptive Signal Models: Theory, Algorithms, and Audio Applications* , PhD thesis, University of California, Berkeley, 1997.

[27] ISO/IEC, *14496-5/FPDAM1 Coding of audio-visual objects – Part 5: Reference Software (MPEG-4 Version 2 Reference Software)*, ISO/IEC JTC1/SC29/WG11 N3309, Mar. 2000.

[28] S. Levine, *Audio Representations for Data Compression and Compressed Domain Processing*, PhD thesis, CCRMA, Stanford University, Dec. 1998.

[29] K. Vos, R. Vafin, R. Heusdens, and W.B. Kleijn, “High-Quality Consistent Analysis-Synthesis In Sinusoidal Coding,” *Proc. AES 17th International Conference*, Sep. 1999.

[30] T. Verma, *A Perceptually Based Audio Signal Model with Application to Scalable Audio Compression*, PhD thesis, Stanford University, 2000.

Further references and related links can be found at:
`http://www.tnt.uni-hannover.de/~purnhage/`