CCIT Report #526

April 2005

Packet Loss Concealment for Audio Streaming Based on the GAPES Algorithm

Hadas Ofir[1], David Malah[2]

[1] Department of Electrical Engineering Technion IIT,Haifa 32000, Israel
hadaso@tx.technion.ac.il

[2] Department of Electrical Engineering Technion IIT,Haifa 32000, Israel
malah@ee.technion.ac.il

ABSTRACT

In this work we present a novel approach for audio packet loss concealment, designed for MPEG-Audio streaming, based only on the data available at the receiver. The proposed method is based on the GAPES (Gapped-data Amplitude and Phase Estimation) algorithm for replacing the missing data, using interpolation in the spectral domain. The MPEG standard uses the Modified Discrete Cosine Transform (MDCT) for compression. However, better interpolation results are obtained by converting the data to the Discrete Short-Time Fourier-Transform (DSTFT) domain. This conversion is done directly using an efficient procedure developed in this work. This technique was tested subjectively and was found to provide better performance than previously reported works, even with a packet loss rate of 30%.

1. INTRODUCTION

With the growing popularity of the internet and the advancement in modem technology, there is increasing interest in using the internet for multimedia broadcasting, such as audio and video. This kind of broadcasting is usually referred to as *streaming media*, and its low cost and convenience make it appealing to both media distributors and consumers.
Audio streaming operates by first compressing a digital audio file and then breaking it into small packets, which are consecutively sent over the internet. When the packets reach their destination, they are decompressed and reassembled into a form that can be played by the user's system. To maintain the illusion of seamless playing, the packets are "buffered". That is, a number of them are downloaded to the user's machine before playback. As those buffered packets are played, more packets are being downloaded and queued up for playback. This way, the client experiences only a small delay of a few seconds, waiting for the buffer to build up, instead of waiting several minutes, or even hours, for the complete files to be downloaded.

However, since internet delivery doesn’t assure quality of service, data packets are often delayed or discarded during network congestions. When the stream of arriving packets becomes too slow, the client’s audio player has nothing to play, thus an annoying gap is created in the streamed media.

Previous researches on this subject [1-3] show that most of the receivers in such internet connections experience a mean loss rate of about 10% or lower, but it is still possible that at certain times the loss rate will go to the extremes, such as up to 20% or 30% loss rate, or no loss at all. At small loss rates (10% or less) the packet losses are random and become more and more correlated as the loss rates go higher [2]. Hence, when the network load is low to moderate, the packet losses are usually isolated audio packets [1]. Each loss, unless concealed in some way, produces an annoying disturbance. The common approach for dealing with such cases is to interpolate the gap, approximating the original waveform, so that a human listener will not notice the disturbance. Packet loss concealment algorithms are usually divided into two categories: *receiver-based* methods where only the data available at the receiver is used for the concealment and *sender-based* methods, where the sender changes the encoded bitstream, adding some redundancy or additional side information that the receiver can later use for the concealment process. In this work we focus on a receiver-based solution.

When designing a receiver-based algorithm for concealment of audio packet loss, the designer wishes to interpolate the gap in a way that will sound natural: In the case of short gaps the replacing signal should have characteristics similar to the lost signal, so that the listener won't even notice it was replaced. Since audio signals are in many cases short-term stationary, these characteristics can be estimated from the surrounding packets, assuming they are available. In the case of long gaps, or gaps that are very close to each other, where it is impossible to perfectly restore the characteristics of the lost segment, it is usually demanded that the replacement should at least sound smooth rather than annoying. However, since typical audio packets correspond to 10-30 msec of audio, the gap created by even a single lost packet is relatively wide (441-1323 samples at 44.1 kHz sampling rate) and is therefore difficult to interpolate.

Previous works on receiver-based audio packet loss concealment start from simple techniques, such as *noise substitution*, *waveform substitution* [4] and *packet repetition*, on to advanced techniques that use interpolation in the compressed domain for MPEG audio coders [5,6] or interpolation using sinusoidal (parametric) audio modeling [7].

For reasons to be explained in the sequel, the proposed solution in this work is to interpolate the gap in the DSTFT domain using only data available at the receiver-side. The algorithm was designed for MPEG Audio coders and although it was implemented on an MPEG-1 Audio coder (a.k.a. MP3), it can be easily adapted also for MPEG-2/4 AAC. Our experiments show that the reconstruction is almost transparent for a 10% loss rate, and yields good quality at higher loss rates, with an acceptable quality achieved even at 30% loss. Besides its good performance, the benefits of this solution are that it assumes no parametric modeling and that it can deal with different loss patterns.

The remainder of the paper is organized as follows: Section 2 gives a brief overview of the MPEG-Audio coding scheme. Section 3 explores the proposed process for efficiently converting MDCT to DSTFT and vice versa. Section 4 describes the proposed concealment algorithm. Section 5 describes the quality tests and their results, and Section 6 concludes the paper.

2. MPEG-AUDIO CODING

The MPEG-1 Audio coder compresses signals sampled at rates of 32, 44.1 or 48 kHz, to rates in the range of 32 to 320 kbps. The standard offers a choice of three independent layers of compression, with increasing codec complexity and better quality. The most interesting among them is MPEG-1 Layer 3, a.k.a. MP3. In recent years, the MP3 coder has become the number one tool for internet audio delivery and is now known in most households as an efficient way to store audio files. The reasons for its success include the fact that MP3 is an open, well defined, standard, along with the fact that it is supported by most hardware manufacturers (sound-cards, CD-ROMs, CD-Writers etc.) and that numerous versions of its encoder and decoder are available on the internet as shareware. But the most important reason of all is the fact that MP3 gives good quality with a small file size, since the quality degradation arising from the MP3 lossy compression process is almost negligible at typical rates. Tests show [11] that at a 6:1 compression ratio (as is the case for a 48 kHz sampled stereo signal represented by 16 bits per sample and encoded into a 256 kbps MP3 file) and under optimal listening conditions, expert listeners could not distinguish between coded and original audio clips with statistical significance.

MP3 and its successors (such as MPEG-2/4 AAC) are part of a family called *perceptual audio coders*. These coders achieve relatively high compression by exploiting characteristics and limitations of the human auditory system. The most useful of them are the *frequency masking property*, the *temporal masking property*, and the *absolute threshold of hearing*.

When a signal is coded, a psychoacoustic model is applied to it in order to determine the signal-to-mask ratio (SMR), based on these three properties. Then, compression is achieved by a quantization process that shapes the quantization noise so it is always underneath the masking threshold, and hence is unnoticeable. Needles to say, that the quality of the psychoacoustic model has a great impact on the quality and efficiency of the encoding process.

2.1. Encoding Process

The MP3 encoder applies the psychoacoustic model in the following manner: The signal is divided into segments of 576 samples each. The psychoacoustic model calculates the SMR for every critical band of the human auditory system within each segment. The model also determines the type of window to be used for the segment in the MDCT process, according to the segment's short- or long-term characteristics.

In parallel to this, each segment passes through 32 filters of a uniform filter-bank creating 32 equal-width sub-bands. Then, each sub-band is further divided into 18 frequency lines by applying an MDCT to each of the sub-bands' signals. Next, the resulting 576 frequency lines are arranged into groups, each group corresponding to a single critical-band, and each of them is quantized separately. The quantization step for each of the critical-band groups is determined by an iterative algorithm that compares the ratio between the energy of the un-quantized signal and the quantization noise in each critical-band to the SMR ratio that was determined for that band. This iterative algorithm controls both the bit-rate and the distortion level, so that the *perceived* distortion is as small as possible, within the limitations of the desired bit-rate. The quantized values and side information (such as the window type for the MDCT) are encoded using Huffman code tables to form a bit-stream. Every two segments of 576 samples form a single MP3 *packet*.

2.2. MDCT and Window Types

The MDCT is a real-valued transform, turning 2N time samples into N MDCT coefficients, and is defined by:

$$X^{MDCT}[k] = \sum_{n=0}^{2N-1} x_w[n] \cdot \cos\left(\frac{\pi}{N} \cdot \left(n + \frac{N+1}{2}\right)\left(k + \frac{1}{2}\right)\right) \quad ,0 \le k \le N\text{-}1 \tag{1}$$

Where $x_w[n]$ represents the original signal multiplied by a window function. The inverse transform is defined by:

$$\hat{x}_w[n] = \frac{2}{N}\sum_{k=0}^{N-1} X^{MDCT}[k] \cdot \cos\left(\frac{\pi}{N} \cdot \left(n + \frac{N+1}{2}\right)\left(k + \frac{1}{2}\right)\right) \quad ,0 \le n \le 2N\text{-}1 \tag{2}$$

The MDCT is a lossy transform since the reconstructed samples, $\hat{x}_w[n]$, contain aliasing in the time domain. However, by overlapping consecutive segments and by defining several conditions on the window functions that the direct and inverse MDCT transforms use, the aliasing can be completely canceled, so that a perfect reconstruction of the un-quantized signal is achieved. The first condition for perfect reconstruction is that the direct and inverse transforms use the same window function. In addition, the window functions of consecutive time segments should overlap by 50% and they should also relate to each other by several other conditions, as specified in [8].

The aliasing cancellation is achieved by multiplying each block of 2N aliased samples, $\hat{x}_w[n]$, with its corresponding window function. Then, each half-block is merged with the overlapping half-block from the neighboring segment into one segment of N samples by an overlap-and-add (OLA) procedure. Hence, we need a total of 3 consecutive MDCT blocks in order to perfectly reconstruct a whole block of 2N samples. Fig. 1 illustrates this concept.

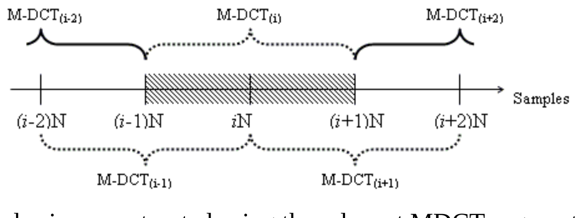

Fig.1: One segment of 2N samples is reconstructed using the relevant MDCT segment and also the MDCT segments on both its sides (dotted brackets).

The MP3 standard allows the psychoacoustic model to choose one of 4 possible window functions, which satisfy the conditions required for perfect reconstruction, to be used when applying the MDCT to each segment. The four windows are denoted [13]: ‘Long’, ‘Start’ (long-to-short), ‘Short’, and ‘Stop’ (short-to-long), and they are marked as

“type 0” to “type 3”, respectively. The length of the ‘Long’ window allows better frequency resolution for audio segments with stationary characteristics, while the length of the ‘Short’ window provides better time resolution for transients. The ‘Start’ and ‘Stop’ windows are transition windows between the ‘Long’ and ‘Short’ window types. There are 3 basic half-window units, described in Fig. 2, that are used in direct or reverse form to create each of the 4 window functions that are shown in Fig. 3. Note that the 3 half-window units are of different lengths: the 'long' and 'short-to-long' halves are 18 samples long, where the 'short' half is only 6 samples long.
The long window types (type 0, 1 and 3) are 36-samples long, and are used for computing an MDCT of 36 samples to 18 coefficients. The short window, however, is actually built of three overlapping 12-samples sub-windows that are used for three short MDCT transforms of 6 coefficients, each. Under these definitions, all the windows result in the same total length with the same number of MDCT coefficients. As was already implied, switching between long and short windows is not instantaneous: a transition window should always come between them. One possible ordering can be seen in Fig. 4. Utilizing this fact can be very helpful in case of a packet loss, since for a single packet loss, the window types of the lost segments can be recovered in most cases by observing the window types of neighboring packets. For example: **long-missing$_1$-missing$_2$-stop** could only match a single possible pattern, where missing$_1$ is a **start** window and missing$_2$ is a **short** window.

3. INTEROPLATION DOMAIN

As has been mentioned, MPEG-Audio coders compress the audio signal in the MDCT domain. Specifically, the MP3 encoder turns each segment of 576 samples in time into 576 MDCT coefficients. Each MP3 packet contains two such segments, therefore one lost MP3 packet creates a gap of 1152 samples. Or in other words, a gap of two coefficients per frequency line in the MDCT domain. Since a smaller gap is easier to interpolate, it makes sense to design a packet-loss concealment algorithm that works in the MDCT domain, rather than in the time domain, as suggested in [6]. In that work, the authors applied an adaptive missing-samples restoration algorithm for auto-regressive time-domain signals in the MDCT domain, considering the coefficients of each frequency line along time as a separate sequence of samples.

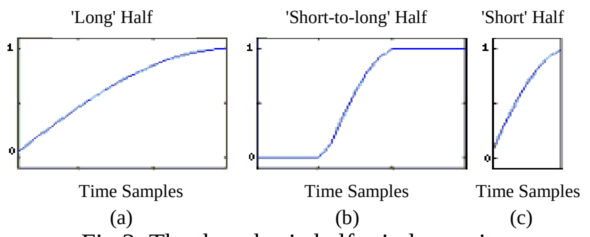

Fig.2: The three basic half-window units:
(a) long, (b) short-to-long, (c) short

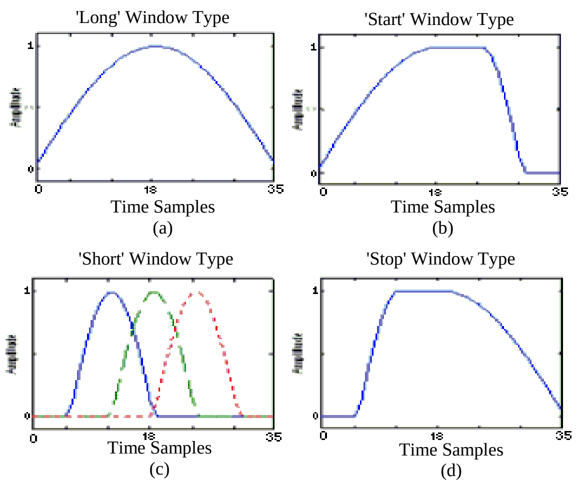

Fig.3: The four different window types defined in MP3:
(a) ‘Long’, (b) ‘Start’, (c) ‘Short’, (d) ‘Stop’

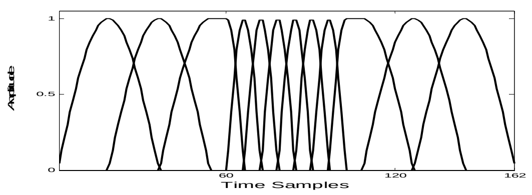

Fig.4: One possible ordering of the different windows

There are, however, several disadvantages in working in the MDCT domain: First, the MDCT coefficients typically show rapid sign changes from frame to frame, at each frequency line. These sign changes reflect phase changes in the complex spectral domain [12]. Second, because different window types have different frequency resolutions, the MDCT coefficients of two consecutive time segments at a certain frequency line might represent different resolutions: For example, consider the case where the first of two consecutive segments uses a 'Start' window while the second uses a 'Short' window. Since the 'Short' window supports 3 short MDCT transforms, each of the coefficients represents $\frac{1}{6}$ of the frequency band. However, each of the coefficients of the 'Start' window represents $\frac{1}{18}$ of the band. Since the data is not represented at the same resolution, it would not make sense to estimate the coefficients of one segment from the other.

A possible way to cope with the first limitation above would be to work in a domain that has a less fluctuating representation of the signal, thus providing better interpolation results. A solution to the second limitation would be to convert the MDCT coefficients back into the time domain and then again to the frequency domain, this time using the same window length for all segments. The DSTFT domain answers both requirements and therefore was chosen as our working domain.

Yet, there is a disadvantage in the conversion to the DSTFT domain, or to every other domain for that matter: Since consecutive MDCT windows overlap by 50%, a loss of one MDCT frame affects not only the reconstruction of the corresponding block, but also the reconstruction of neighboring blocks, as shown in Fig. 1. Therefore, a loss of L consecutive MP3 packets, which corresponds to $2L$ consecutive MDCT frames, is translated into a gap of $2L+2$ DSTFT frames: $2L$ absent frames in the middle and 2 corrupted frames at the edges of the gap. An example for this is given in Fig. 5.

3.1. From MDCT to DSTFT and vice versa

In order to convert from one working domain to the other, we have developed expressions for direct conversion from MDCT to DSTFT, and vice versa, applying the DSTFT to the segments originally used by the MDCT, with a symmetric window that has two halves that complement each other to the value of 1. As was explained in section 2.2, three consecutive MDCT blocks are required in order to reconstruct a whole segment of $2N$ samples. Only then can we apply a DSTFT on this segment. The conversion back from the DSTFT domain to the MDCT domain requires an OLA procedure in a similar manner.

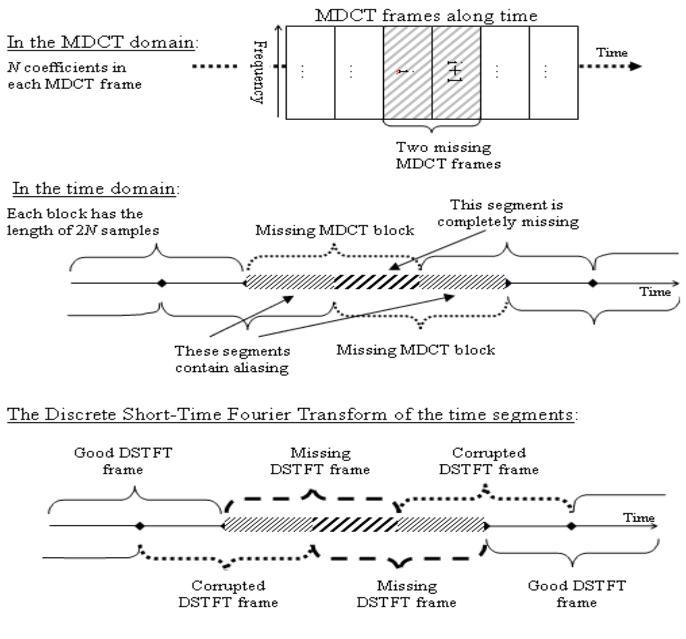

Fig.5: How a loss of a packet in the MDCT domain affects the DSTFT domain: 2 lost MDCT frames affect the reconstruction of 2+2=4 DSTFT frames

Since there are 4 possible window types for the MDCT, and considering the allowed ordering of the windows, we could theoretically get 12 different expressions for the conversion of 3 consecutive MDCT segments into one DSTFT segment. But, after changing the order of the summations and some algebraic manipulation, all these expressions converge into one general structure that uses 4 functions as "building-blocks", given below. The same happens in the conversion of DSTFT segments back to the MDCT domain.

The general expression for the conversion from the MDCT domain to the DSTFT domain is:

$$\begin{aligned} X_{(n)}^{DSTFT}[m] &= \sum_{k=0}^{N-1} X_{(n)}^{MDCT}[k]\cdot\left(g_d^1[m,k]+(-1)^m\cdot g_r^2[m,k]\right) \\ &+\sum_{k=0}^{N\text{-}1} X_{(n-1)}^{MDCT}[k]\cdot g_d^2[m,k]+\sum_{k=0}^{N-1} X_{(n+1)}^{MDCT}[k]\cdot\left((-1)^m\cdot g_r^1[m,k]\right) \quad ,0\le m\le N \end{aligned} \tag{3}$$

Where $g_d^1[m,k], g_r^1[m,k]$ and $g_d^2[m,k], g_r^2[m,k]$ are the “building-block" functions referred to earlier. Each building-block function is chosen among 3 possible complex functions, depending on the window type of the segment that we wish to convert and on the window types of neighboring segments. Table 1 shows the different cases, where the first column is the window type of the converted segment. Note that since the DSTFT is performed on a 2*N* real-valued sequence, the output is conjugate-symmetric in the frequency domain. Therefore, it suffices to calculate only the first half of the output. In total, there are 12 complex functions, based on the 3 possible half-window units that were shown in Fig. 2. The explicit expressions for the different functions that appear in Table 1 are given in appendix A.

In a similar manner, the conversion from the DSTFT domain back into the MDCT domain can also be expressed as a general expression of a sum of products, while in this case we need to calculate only the real part of the result, since the MDCT coefficients are real-valued. The expression is given in (4) below, where * denotes ‘complex conjugate’ value. The building blocks here are chosen from the same functions as the direct conversion, according to Table 2.

$$\begin{aligned} X^{MDCT}[k] &= \frac{1}{N\cdot N_{block}}\sum_{m=0}^{2N-1} X_{(n)}^{DSTFT}[m]\cdot\left(\left(g_d^1[m,k]^*+g_r^1[m,k]^*\right)+(-1)^m\cdot\left(g_d^2[m,k]^*+g_r^2[m,k]^*\right)\right) \\ &+\frac{1}{N\cdot N_{block}}\sum_{m=0}^{2N-1} X_{(n-1)}^{DSTFT}[m]\cdot(-1)^m\cdot\left(g_d^1[m,k]^*+g_r^1[m,k]^*\right) \qquad ;0\le k\le N-1 \\ &+\frac{1}{N\cdot N_{block}}\sum_{m=0}^{2N-1} X_{(n+1)}^{DSTFT}[m]\cdot\left(g_d^2[m,k]^*+g_r^2[m,k]^*\right) \end{aligned} \tag{4}$$

Table 1: The different functions appearing in (3). Explicit expressions appear in Appendix A.

Function / Window type	$g_d^1[m,k]$	$g_r^1[m,k]$	$g_d^2[m,k]$	$g_r^2[m,k]$
‘Long’	g_d^1_long	g_r^1_long	g_d^2_long	g_r^2_long
‘Start’	g_d^1_long	g_r^1_short	g_d^2_long	g_r^2_short2long
‘Short’	g_d^1_short	Next window is ‘Stop’: g_r^1_short2long	Previous window is ‘Start’: g_d^2_short2long	g_r^2_short
		Next window is ‘Short’: g_r^1_short	Previous window is ‘Short’: g_d^2_short	
‘Stop’	g_d^1_short2long	g_r^1_long	g_d^2_short	g_r^2_long

Table 2: The different functions appearing in (4). Explicit expressions appear in Appendix A.

Function / Window type	$g_d^1[m,k]$	$g_r^1[m,k]$	$g_d^2[m,k]$	$g_r^2[m,k]$	N_{block}
'Long'	g_d^1_long	g_r^1_long	g_d^2_long	g_r^2_long	N
'Start'	g_d^1_long	g_r^1_long	g_d^2_short2long	g_r^2_short2long	N
'Short'	g_d^1_short	g_r^1_short	g_d^2_short	g_r^2_short	$N_s = \frac{N}{3}$
'Stop'	g_d^1_short2long	g_r^1_short2long	g_d^2_long	g_r^2_long	N

The fact that we have one general expression for each direction of the conversion can be used to create an efficient procedure to convert between the two domains: Each general expression can be optimized into an efficient computer-function, while the "building-block" functions can be calculated off-line and stored. The same 12 "building-block" functions are used in the expressions for both conversion directions. Each "building-block" function contains $N(N-1)$ complex values and $2N$ real values (for $m=0,N$), so in total, $2N^2$ real numbers need to be stored for each function. In our case $N = 18$.

4. CONCEALMENT ALGORITHM

The proposed concealment method estimates the missing data in the DSTFT domain based on the GAPES algorithm [10]. The GAPES algorithm was chosen because of several of its qualities: As opposed to the algorithm suggested in [6], the GAPES algorithm assumes no parametric modeling of the signal. It can deal with more loss patterns than [6], because it is less limited, and can be applied to both complex and real signals. Although this paper describes an algorithm that involves GAPES in the DSTFT domain, it is important to note that we also considered applying GAPES directly in the MDCT domain. The benefits of such configuration are that there is no need to convert the data to any other domain, and that there are much fewer calculations since the signal is real-valued. The limitations of this configuration, however, are those mentioned in section 3.1: the problem of dealing with windows having different resolutions, and the rapid sign changes. Our subjective tests showed that this configuration is inferior to the GAPES-in-DSTFT configuration, especially at high loss rates. The results of the subjective comparison tests are given in section 5.

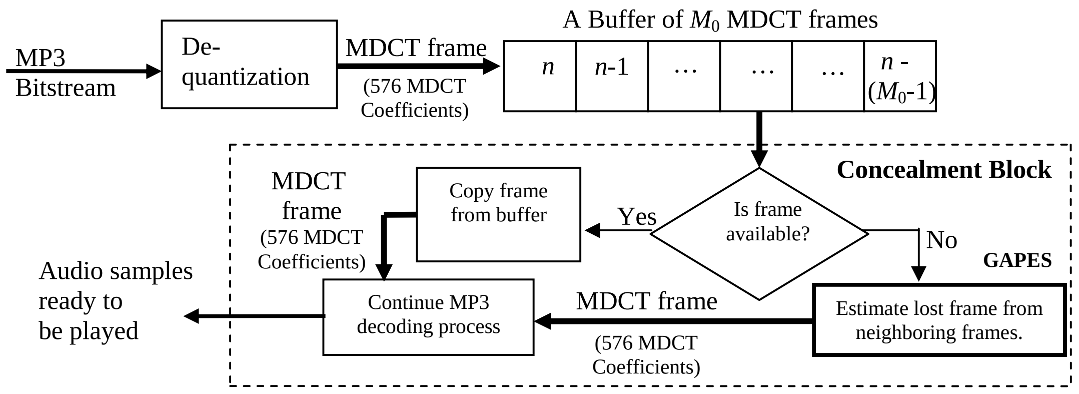

Fig.6: A diagram of the proposed decoding process that includes a concealment block

Fig. 6 shows the block diagram of an MP3 decoder, after adding the concealment block. In the decoder, every new MP3 packet is decoded up to the MDCT level (i.e., de-quantized) resulting in two MDCT frames. The M_0 most recent MDCT frames are stored in a buffer, indexed from 0 to M_0-1. Their associated data (i.e., each frame's window type) is also stored separately. If a packet is lost, the corresponding MDCT values are set to zero and a flag is raised, indicating that this frame is actually missing. The window type of each of the missing frames is determined so that they comply with the window types of neighboring frames. Then, according to the system's delay, a delayed MDCT frame is copied from the buffer and decoded into waveform samples. In the case where the delayed frame is actually a missing frame, we estimate its MDCT values before continuing with the decoding process.

Fig. 7 shows the concealment process: The MDCT frames are converted to the DSTFT domain, where the data along the time axis at each frequency is considered as an independent complex signal, and a single iteration of the GAPES algorithm is applied separately on each of these signals. After that, the data is converted back to the MDCT domain and then back again to the DSTFT domain, in order to merge the estimated MDCT frames with the available ones, using the OLA procedure that is incorporated in the conversion expressions. The process above is iterated until the difference between two consecutive reconstructions is sufficiently small. The estimated MDCT coefficients replace the coefficients of the lost frame and the MP3 decoding process continues.

It has already been explained that in order to convert an MDCT frame into the DSTFT domain we need to have both the previous and the next MDCT frames, and in order to convert that DSTFT frame back to the MDCT domain we need the next DSTFT frame too. This implies that the algorithm has to maintain a delay of at least two MDCT frames at all times, i.e., at least one MP3 packet. Adding more delay is not necessary, but it can improve the performance of the GAPES algorithm, as explained in Appendix B. The extra delay will not have much significance to the listener, since adding delay of a few packets will only postpone the beginning of the whole streaming session by less than a second.

The buffer, of length M_0, that holds the most recent MDCT frames, should not be too short, so there is enough data for the estimation. The buffer should not be too long either, since as the buffer grows longer, the frames at both ends will have less correlation with the missing frames around the middle, and hence it will be less effective to include them in the estimation. In the case of MP3, each MDCT frame contains 576 MDCT coefficients: N=18 coefficients for each of the 32 uniform sub-bands. Hence, this buffer can be viewed as a real-valued matrix of 576 rows and M_0 columns (as shown in Fig. 7).

Due to packet loss, several MDCT frames in the buffer may be missing, meaning that the original values of some of the matrix's columns are unknown and are currently set to zero. Let's assume that there are $Q_0 < M_0$ missing frames in the buffer in some general loss pattern. Since we are dealing with a streaming application, we can assume that by the time we need to conceal a lost frame, all the frames prior to it had already been reconstructed, and therefore they are considered as part of the available data. We can choose to conceal more than one frame at a time, depending on the loss pattern: lost frames that are gathered closely together are concealed together, where distant frames are concealed separately. Concealing several frames at once can reduce the computational overhead of the whole concealment process. We will refer to a concealment of one or more MDCT frames at once as a *concealment session*. Let Q denote the number of MDCT frames that are concealed in one concealment session. The value of Q is reassigned at every session. Since each MP3 packet contains 2 MDCT frames, one concealment session always contains at least 2 concealed frames (i.e., $2 \le Q \le Q_0$). After deciding on the value of Q, we define the Q lost frames and the available frames that surround them, as the buffer section that will participate in the reconstruction of this concealment session. The length of this buffer section is denoted M, where $Q<M \le M_0$.

In the first stage of the algorithm, the MDCT frames that were selected to participate in the reconstruction are converted to the DSTFT domain. The conversion is done separately for each of the 32 sub-bands, and after the conversion is completed we have a complex-valued matrix of (M-2) columns and $36 \cdot 32$ rows of DSTFT coefficients: 2N=36 coefficients for each of the 32 uniform sub-bands. The DSTFT frames are indexed from 1 to (M-2). Since the DSTFT coefficients of each transform are conjugate-symmetric it is enough to store only 19 (0 to N) coefficients of each sub-band, i.e., the matrix can be downsized into $19 \cdot 32$ rows.

At this point, each row in the DSTFT matrix, that represents the DSTFT coefficients at some fixed frequency along the time axis, is considered as an independent complex signal with missing samples, described by the row vector:

$$\underline{x}^k \triangleq \left\{ x_1^k, x_2^k, ..., x_{M-2}^k \right\} \quad (5)$$

Where k is the frequency index: $0 \le k \le 19 \cdot 32\text{-}1$. The locations of the missing samples in the vector are actually the indices of the missing MDCT frames, since every sample in the vector $\underline{x}^k$ corresponds to a single MDCT frame in

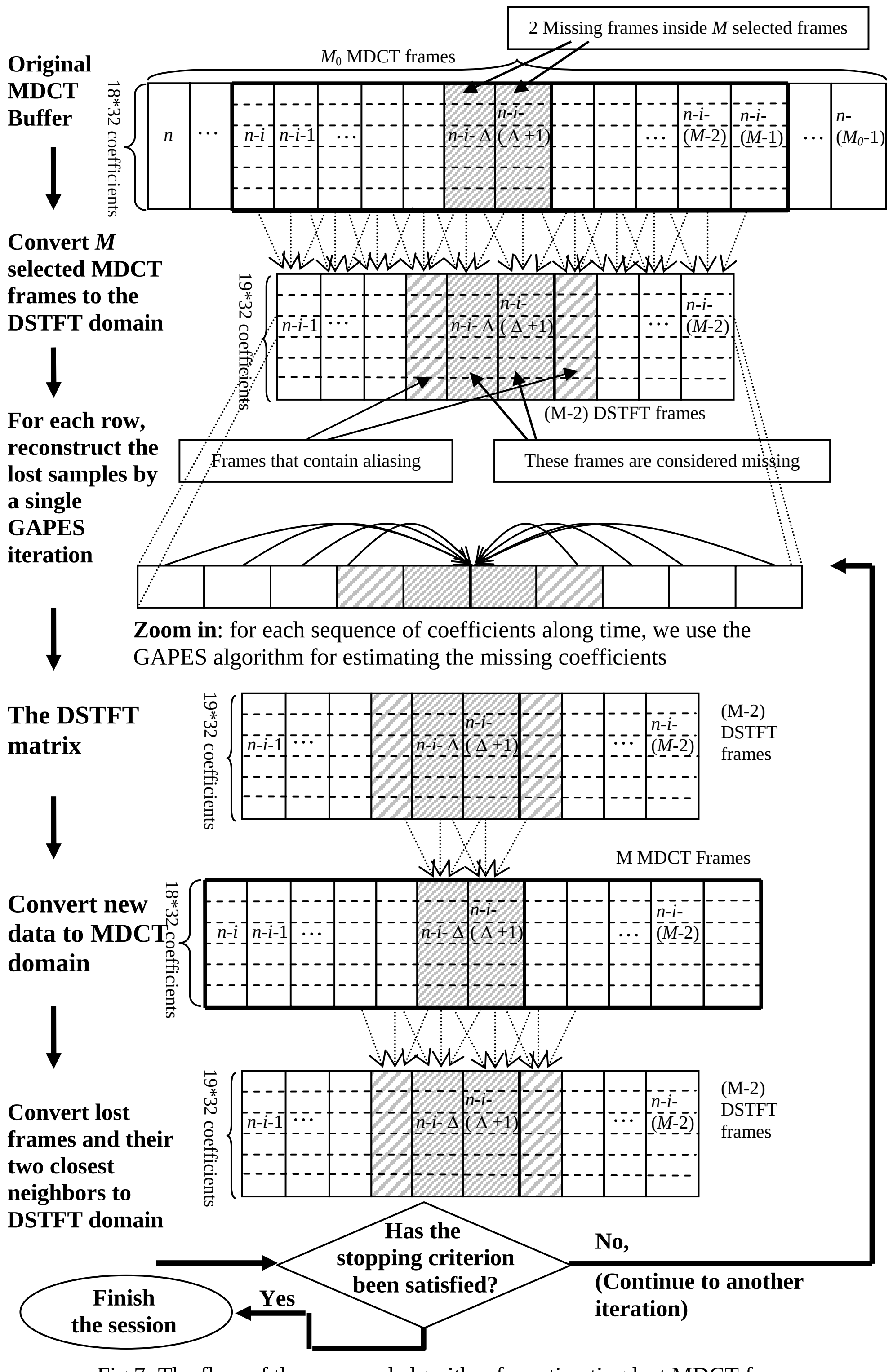
2 Missing frames inside M selected frames
M_0 MDCT frames
Original MDCT Buffer
18*32 coefficients
n
n-i
n-i-1
n-i- Δ
n-i-(Δ+1)
n-i-(M-2)
n-i-(M-1)
n-(M_0-1)
Convert M selected MDCT frames to the DSTFT domain
19*32 coefficients
(M-2) DSTFT frames
Frames that contain aliasing
These frames are considered missing
For each row, reconstruct the lost samples by a single GAPES iteration
Zoom in: for each sequence of coefficients along time, we use the GAPES algorithm for estimating the missing coefficients
The DSTFT matrix
(M-2) DSTFT frames
M MDCT Frames
Convert new data to MDCT domain
Convert lost frames and their two closest neighbors to DSTFT domain
Has the stopping criterion been satisfied?
No, (Continue to another iteration)
Yes
Finish the session

Fig.7: The flow of the proposed algorithm for estimating lost MDCT frames

the buffer. The missing samples are estimated from the available samples by applying the GAPES algorithm.

4.1. The GAPES Algorithm

The GAPES algorithm [10] reconstructs the missing data, assuming that the missing data has the same spectral content as the available data that surrounds it. For the purpose of applying GAPES, each row of the DSTFT matrix, $\underline{x}^k$, is considered as a sequence of samples with complex values. The spectrum of these complex samples is represented by a set of spectral coefficients, $\left\{\hat{\alpha}\left(\omega_k\right)\right\}$, that are calculated on a pre-defined frequency grid $\left\{\omega_k\right\}$. The spectral coefficients of the complex samples are estimated from the available data using the APES (Amplitude and Phase Estimation) algorithm [9], and then the algorithm reconstructs the set of missing samples so that their spectral content will approximate the spectrum of the available data, in the least-squares (LS) sense.

The APES algorithm uses an adaptive filter-bank approach for the spectral estimation: for each frequency ω_k on the frequency grid, a *data-dependent* narrowband filter, $\underline{h}_k$, is designed by requiring that the output of the filter $\underline{h}_k$ will be as close as possible, in the LS sense, to a sinusoid with frequency ω_k. I.e., the filter $\underline{h}_k$ passes the frequency ω_k without distortion, and at the same time, attenuates all the other frequencies as much as possible. Each spectral coefficient $\hat{\alpha}\left(\omega_k\right)$ is obtained by first filtering the data with the corresponding filter, $\underline{h}_k$, and then calculating the DFT coefficient of the filtered data at frequency ω_k. Then, assuming that the missing samples have the same spectral content as the available samples, we can determine their values by requiring that the output of the filter, $\underline{h}_k$, fed by the full data sequence that contains both available and estimated samples, is as close as possible, in the LS sense, to the sinusoid $\hat{\alpha}\left(\omega_k\right)e^{-j\omega_k t}$.

This concludes a single GAPES iteration. The algorithm is iterated, as explained next, until the reconstruction is sufficiently good. At the first iteration, APES uses only the available samples for determining $\left\{\underline{h}_k\right\}$ and $\left\{\hat{\alpha}\left(\omega_k\right)\right\}$, while in the next iterations the algorithm uses all the data (known and estimated samples) for the spectral estimation. The formulae summarizing the GAPES algorithm are given in Appendix B.

After a single GAPES iteration is applied on all the row vectors of the DSTFT matrix, we have an estimate of the DSTFT coefficients of the missing frames. Since each DSTFT frame represents data from overlapping segments, we need to merge the data from consecutive frames. The merging process achieves two goals: First, by merging the data we get a continuous waveform that sounds better. Second, we can use the merging process to improve the estimation: re-calculating the lost DSTFT frames using the information stored in the available MDCT frames that are nearest to the loss will give a better basis for the next GAPES iteration, and in addition, the estimated MDCT frames along with the available MDCT frames can be used for reducing the aliasing in the DSTFT frames closest to the loss (see Fig. 5), by re-calculating them too. In our scheme, this merging is achieved by converting the estimated DSTFT frames back to the MDCT domain, and then recalculating the lost DSTFT frames and their nearest neighbors by converting the corresponding MDCT frames back to the DSTFT domain. It is important to note, though, that in order to merge the data, one doesn't have to go as far as to the MDCT domain, but may perform it directly in the time domain. The benefit of using conversion to and from the time domain would be simpler conversion schemes such as FFT. The limitation of such conversion would be the need for more memory space, in order to hold the time-samples of the frames that require merging.

In the final stage of the algorithm it is decided whether the estimation is good enough. As a stopping criterion for the concealment algorithm, we use a threshold over the average squared difference per subband, as defined in (6). This ratio is calculated over all rows in the DSTFT matrix corresponding to a sub-band (each sub-band contains 19 rows):

$$D(sb) = \frac{1}{19}\sum_{k=0}^{19}\left(\frac{1}{|S|}\sum_{i\in S}\frac{\left|\hat{\alpha}_{k,i}^{new} - \hat{\alpha}_{k,i}^{old}\right|^2}{\left|\hat{\alpha}_{k,i}^{old}\right|^2}\right) \qquad (6\,)$$

Where $0 \le sb \le 31$, S is the set of indices of the Q lost MDCT frames that are concealed in this session: $|S| = Q$. $D(sb)$ is compared to a predefined threshold. If it is smaller than the threshold, then the estimation is sufficiently good, so we can stop at this point and continue with the MP3 decoding process. If it is larger than the threshold, then we apply another iteration on this subband, as shown in Fig. 7. In addition, it is recommended to limit the number of iterations in order to deal with cases where the convergence is very slow, or where the average difference values are "stuck" in a limit cycle.

To conclude, we can describe the algorithm in 5 main steps, applied to each subband of the uniform filter-bank:

1. Initialization:
   - Store zero values in place of each lost MDCT frame.
   - Determine the parameters for this concealment session:
     Q, the number of frames to be concealed and *M*, the length of buffer section.
   - Convert the MDCT buffer to the DSTFT domain.

2. Estimate the DSTFT coefficients of the lost frames:
   - Treat each sequence of DSTFT coefficients at a certain frequency, along the time axis, as a complex signal with
     missing samples.
   - Estimate the lost samples by applying a single GAPES iteration on each sequence.

3. Restore the values of the lost MDCT frames:
   Convert the corresponding DSTFT frames into the MDCT domain.

4. Correct the values in the DSTFT domain:
   Correct the values of the lost frames and their closest neighbors by converting the corresponding MDCT frames to the DSTFT domain.

5. Check the stopping criterion, per sub-band:
   If it is satisfied, stop the algorithm for this sub-band in the current session and use the MDCT frames that were obtained last. If not, return to (2) for another iteration.

It should be mentioned that the above described algorithm uses the Forward version of GAPES. Another option is to apply the Forward-Backward version [9]. In the Forward-Backward version the data is run through the filters both forward and backward, instead of using forward-filtering only. Unfortunately, this method did not show any significant improvement in our tests, to justify its complexity.

4.2. Complexity Considerations

The proposed algorithm shows good performance, but at the expense of complexity: A single iteration of the algorithm, per sub-band, requires: $O\left(48\cdot N^2\right)+\left(N+1\right)\cdot\left[O\left(12\cdot M^3\right)+O\left(M^2\cdot\log_2 M\right)\right]$ multiplications.

The first option is to compromise quality and use the GAPES algorithm in the MDCT domain, as was described at the beginning of this section. This option reduces the complexity to $N \cdot \left[O\left(2 \cdot M^3\right) + O\left(\frac{1}{2} M^2 \cdot \log_2 M\right) \right]$ multiplications. For example, for N=18 and M=14, GAPES-in-DSTFT requires 6 times the amount of multiplications needed by GAPES-in-MDCT.

Another option that can help to reduce the number of calculations in GAPES-in-DSTFT, is reducing the number of iterations on the basis of psychoacoustic considerations. Assuming that the MP3 encoder shapes the quantization noise according to the signal-to-mask ratios (SMR), we can get a rough estimate of these ratios at the decoder, from the closest neighbors of the lost MDCT frame: the quantization step size in each critical band gives an indication on the allowed level of noise, and the energy of the quantized signal in each band can be used to estimate the original signal's energy in the band. Based on the SMR estimation we can determine which critical bands are more sensitive to noise than others: A smaller SMR ratio indicates a high masking level relatively to the signal’s energy. Hence, it can be assumed that this band can also tolerate more “reconstruction noise”, i.e., less accurate estimation. Similarly, a bigger SMR ratio indicates a lower masking level relatively to the signal’s energy, meaning that it is more sensitive to noise addition. This information can be used to create different stopping-thresholds for different sub-bands, without compromising the quality of the resulting signal, by calculating a weight function on the threshold values: Sub-bands that contain coefficients from more sensitive critical bands will have lower thresholds than sub-bands that correspond to less sensitive critical bands. This method shows good performance and it saves many iterations.

5. QUALITY TESTS

This section reports on the results of two comparative subjective quality tests of algorithms for packet loss concealment, designed for wide-band audio signals encoded by an MPEG audio coder. The first test compares GAPES-in-MDCT to GAPES-in-DSTFT, in terms of listeners’ preference. The second test compares the GAPES-in-DSTFT algorithm to previously reported works: *packet repetition*, suggested in the MP3 standard [13, annex E], and *statistical interpolation* (SI), suggested in [6]. The SI method was expanded in order to deal with random loss patterns, so that when the conditions weren’t suitable for using the original algorithm we used packet repetition instead.

The packet repetition method introduces no delay since the decoder simply replaces a lost MDCT frame with the previous frame. In both versions of GAPES and in the SI method, a buffer of 9 MP3 packets, i.e., $M_0 = 18$, was used: 3 packets from the past, the current packet for decoding and 5 packets from the future, creating a delay of about 130 msec at a sampling rate of 44.1 kHz. Also, the stopping criteria threshold that was used in all methods was fixed to $D(sb)=10^{-2}$ (see (6)), and the number of iterations was limited to a maximum of 6 iterations. The window function that was used for the DSTFT was an even length Hann window that can be put in the following form, which is equal to the square of the 'Long' window type used in the MP3 standard:

$$w[n] = \sin^2\left(\tfrac{\pi}{2N}\left(n+\tfrac{1}{2}\right)\right), \quad 0 \le n \le 2N-1 \qquad (7)$$

The subjective tests were carried out by informal listening. All the participants are inexperienced listeners with normal hearing and in the age range of 25-35 years. Each of the listeners was asked to compare pairs of audio files, where the packet losses in each file were concealed by a different method, and to decide which of the two he, or she, prefers.

Table 3: Examined files

No.	File Name	Nature of music
1	Beatles17.wav	POP music
2	Piano10.wav	A single piano
3	Bream1.wav	Guitar with violins

The audio files that were used in the tests are specified in Table 3. All the files are stereo signals ,15-17 seconds long each, sampled at 44.1 kHz and coded by the LAME MP3 [14] encoder at a bit-rate of 128 kbps per channel. The methods were tested for 10%, 20% and 30% loss rates. The packet losses were simulated using random patterns, with the largest possible gap allowed being 3 packets.

5.1. GAPES: MDCT vs. DSTFT

This test compares the performance of GAPES-in-DSTFT to GAPES-in-MDCT, as described at the beginning of Section 4. Eight listeners were asked to compare pairs of files containing packet losses, each concealed by a different method, and to determine which of the two files sounds less disturbing. The results are presented in Table 4, where for each pair the numbers indicate how many listeners voted **in favor** of the method. The results show clearly that GAPES-in-DSTFT performs better, especially at high loss rates.

Table 4: Comparative test results of GAPES-in-MDCT vs. GAPES-in-DSTFT. The numbers indicate how many listeners voted **in favor** of each method.

File No. 1		
Loss Rate	GAPES-in-MDCT	GAPES-in-DSTFT
10%	2	6
20%	0	8
30%	0	8

File No. 2		
Loss Rate	GAPES-in-MDCT	GAPES-in-DSTFT
10%	3	5
20%	0	8
30%	0	8

File No. 3		
Loss Rate	GAPES-in-MDCT	GAPES-in-DSTFT
10%	2	6
20%	1	7
30%	0	8

5.2. GAPES vs. Previous Works

16 listeners were asked to compare pairs of files, the same way as in the previous section, where in this test we examined 3 methods: packet repetition, statistical interpolation, and the proposed method. Table 5 clearly shows that the proposed algorithm performs better than the two previously reported methods.

Table 5: Comparative test results of GAPES-in-DSTFT vs. Previously reported works. The numbers indicate how many listeners voted **in favor** of each method.

	File No. 1					
	GAPES-in-DSTFT vs. Repetition		GAPES-in-GDSTFT vs. SI		GAPES-in-DSTFT vs. Uncorrupted Original	
Loss Rate	GAPES	Rep.	GAPES	SI	GAPES	Original
10%	14	2	16	0	4	12
20%	16	0	16	0		
30%	16	0	16	0		

	File No. 2					
	GAPES-in-DSTFT vs. Repetition		GAPES-in-GDSTFT vs. SI		GAPES-in-DSTFT vs. Uncorrupted Original	
Loss Rate	GAPES	Rep.	GAPES	SI	GAPES	Original
10%	15	1	16	0	5	11
20%	16	0	16	0		
30%	15	1	16	0		

	File No. 3					
	GAPES-in-DSTFT vs. Repetition		GAPES-in-GDSTFT vs. SI		GAPES-in-DSTFT vs. Uncorrupted Original	
Loss Rate	GAPES	Rep.	GAPES	SI	GAPES	Original
10%	16	0	15	1	2	14
20%	15	1	16	0		
30%	12	4	14	2		

6. CONCLUSION

We have introduced a new packet loss concealment algorithm for wide-band audio signals encoded by MPEG audio coders, based on the GAPES algorithm in the DSTFT domain. We used comparative informal listening tests to test the algorithm at different loss rates, in the range of 10% to 30%, and different music types, and obtained that the proposed algorithm performs better than two previously reported algorithms: packet repetition [13, annex E] and statistical interpolation [6]. In addition, a direct conversion scheme was introduced that enables efficient conversion from the MDCT domain to the DSTFT domain and vice versa.

7. ACKNOWLEDGEMENTS

We would like to thank Dr. Erik Larsson, from George Washington University, Washington DC, for sending us his Matlab implementation of GAPES, which helped us develop our C implementation of the algorithm, and to all the colleagues who participated in the listening tests.

APPENDIX A

We give here the expressions for the 12 functions that were mentioned in section 3 as the “building-block” functions for the direct and reverse conversions, where $0 \le m \le N$ and $0 \le k \le N$-1.

$$\mathrm{g}_{\mathrm{d}}^{1}\_\mathrm{long}\left[m,k\right]=\sum_{n=0}^{N-1}\cos\left[\frac{\pi}{N}\cdot\left(n+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right]\cdot h^{long}\left[n\right]\cdot w\left[n\right]\cdot e^{-j\frac{\pi}{N}\cdot n\cdot m} \tag{A.1}$$

$$\mathrm{g}_{\mathrm{r}}^{1}\_\mathrm{long}\left[m,k\right]=\sum_{n=0}^{N-1}\cos\left[\frac{\pi}{N}\cdot\left(n+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right]\cdot h^{long}\left[n\right]\cdot w\left[N-n-1\right]\cdot e^{-j\frac{\pi}{N}\cdot n\cdot m} \tag{A.2}$$

$$\mathrm{g}_{\mathrm{d}}^{2}\_\mathrm{long}\left[m,k\right]=$$
$$\sum_{n=0}^{N-1}\cos\left[\frac{\pi}{N}\cdot\left(n+N+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right]\cdot h^{long}\left[N-1-n\right]\cdot w\left[n\right]\cdot e^{-j\frac{\pi}{N}\cdot n\cdot m} \tag{A.3}$$

$$\mathrm{g}_{\mathrm{r}}^{2}\_\mathrm{long}\left[m,k\right]=$$
$$\sum_{n=0}^{N-1}\cos\left[\frac{\pi}{N}\cdot\left(n+N+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right]\cdot h^{long}\left[N-1-n\right]\cdot w\left[N-n-1\right]\cdot e^{-j\frac{\pi}{N}\cdot n\cdot m} \tag{A.4}$$

$\mathrm{g}_{\mathrm{d/r}}^{1}\_\mathrm{short2long}\left[m,k\right]$ and $\mathrm{g}_{\mathrm{d/r}}^{2}\_\mathrm{short2long}\left[m,k\right]$ are the same as $\mathrm{g}_{\mathrm{d/r}}^{1}\_\mathrm{long}$ and $\mathrm{g}_{\mathrm{d/r}}^{2}\_\mathrm{long}$, respectively, except that they use the window function $h^{short2long}$ instead of h^{long}.

Before introducing the expressions for the short window type, it is important to note that the MDCT coefficients of a short window type segment (i.e., type 2) are organized differently than in a segment with long window types (i.e., type 0,1 and 3), since in a short window we have the coefficients of 3 short MDCT transforms. The functions in the case of short window type consider the ordering of the coefficients. Fig. 8 shows how the coefficients are organized in the output of the MDCT function at the encoder (and respectively, in the input of the Inverse-MDCT function at the decoder).

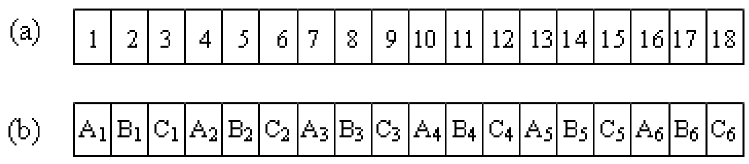

Fig.8: (a) Long window types – coefficients arranged in ascending order: 1-18. (b) Short window type – interleaving the coefficients of the 3 short MDCT transforms: A-C, according to their ascending order: 1-6.

The expressions of $\mathrm{g}^1_{\mathrm{d/r}}\_\mathrm{short}\left[m,k\right]$ and $\mathrm{g}^2_{\mathrm{d/r}}\_\mathrm{short}\left[m,k\right]$ are given next, where $N_s = \frac{N}{3}$, $0 \le m \le N$ and $0 \le k \le N$-1:

$$\mathrm{g}^1_{\mathrm{d}}\_\mathrm{short}[m,k] = \tag{A.5}$$

$$\begin{cases} \displaystyle\sum_{n=0}^{N_s-1}\left[\begin{aligned}&\cos\left(\frac{\pi}{N_s}\cdot\left(n+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[n\right]\cdot w\left[n+N_s\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+N_s\right)} \\ &+\cos\left(\frac{\pi}{N_s}\cdot\left(n+N_s+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[N_s-1-n\right]\cdot w\left[n+2N_s\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+2N_S\right)}\end{aligned}\right] & ,k=0,3,6,\ldots,N-3 \\ \displaystyle\sum_{n=0}^{N_s-1}\cos\left(\frac{\pi}{N_s}\cdot\left(n+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[n\right]\cdot w\left[n+2N_s\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+2N_s\right)} & ,k=1,4,7,\ldots,N-2 \\ 0 & ,k=2,5,8,\ldots,N-1 \end{cases}$$

$$\mathrm{g}^1_{\mathrm{r}}\_\mathrm{short}[m,k] = \tag{A.6}$$

$$\begin{cases} \displaystyle\sum_{n=0}^{N_s-1}\left[\begin{aligned}&\cos\left(\frac{\pi}{N_s}\cdot\left(n+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[n\right]\cdot w\left[N-1-\left(n+N_s\right)\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+N_s\right)} \\ &+\cos\left(\frac{\pi}{N_s}\cdot\left(n+N_s+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[N_s-1-n\right]\cdot w\left[N-1-\left(n+2N_s\right)\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+2N_S\right)}\end{aligned}\right] & ,k=0,3,6,\ldots,N-3 \\ \displaystyle\sum_{n=0}^{N_s-1}\cos\left(\frac{\pi}{N_s}\cdot\left(n+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[n\right]\cdot w\left[N-1-\left(n+2N_s\right)\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+2N_s\right)} & ,k=1,4,7,\ldots,N-2 \\ 0 & ,k=2,5,8,\ldots,N-1 \end{cases}$$

$$\mathrm{g}_{\mathrm{d}}^{2}\_\mathrm{short}[m,k] = \tag{A.7}$$

$$\begin{cases} 0 & ,k = 0,3,6,...,N-3 \\ \displaystyle\sum_{n=0}^{N_S-1} \cos\left(\frac{\pi}{N_S}\cdot\left(n+N_S+\frac{N_S+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[N_S-1-n\right]\cdot w\left[n\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot n} & ,k=1,4,7,...,N-2 \\ \displaystyle\sum_{n=0}^{N_S-1}\left[\begin{array}{l} \cos\left(\frac{\pi}{N_S}\cdot\left(n+\frac{N_S+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[n\right]\cdot w\left[n\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot n} \\ +\cos\left(\frac{\pi}{N_S}\cdot\left(n+N_S+\frac{N_S+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[N_S-1-n\right]\cdot w\left[n+N_s\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+N_S\right)} \end{array}\right] & ,k=2,5,8,...,N-1 \end{cases}$$

$$\mathrm{g}_{\mathrm{r}}^{2}\_\mathrm{short}[m,k] = \tag{A.8}$$

$$\begin{cases} 0 & ,k = 0,3,6,...,N-3 \\ \displaystyle\sum_{n=0}^{N_s-1} \cos\left(\frac{\pi}{N_s}\cdot\left(n+N_s+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[N_s-1-n\right]\cdot w\left[N-1-n\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot n} & ,k=1,4,7,...,N-2 \\ \displaystyle\sum_{n=0}^{N_s-1}\left[\begin{array}{l} \cos\left(\frac{\pi}{N_s}\cdot\left(n+\frac{N_s+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[n\right]\cdot w\left[N-1-n\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot n} \\ +\cos\left(\frac{\pi}{N_s}\cdot\left(n+N_s+\frac{N_S+1}{2}\right)\cdot\left(\left\lfloor\frac{k}{3}\right\rfloor+\frac{1}{2}\right)\right)\cdot h^{short}\left[N_s-1-n\right]\cdot w\left[N-1-\left(n+N_s\right)\right]\cdot e^{-j\frac{\pi}{N}\cdot m\cdot\left(n+N_s\right)} \end{array}\right] & ,k=2,5,8,...,N-1 \end{cases}$$

APPENDIX B

Here we summarize the Forward-Only GAPES algorithm according to [9] and [10]. The Forward-Backward version of the algorithm includes small modifications that can be found in the referenced articles. In addition, some numerical stability precautions are mentioned at the end of this appendix.

Consider a sequence of data samples of length *M*: $\underline{x} = \left\{x_0, x_1, ..., x_{M-1}\right\}$. Let's assume that *q* of the samples are missing in some general loss pattern. The object of the algorithm is to estimate them from the available samples.

Let $\underline{x}_a$ and $\underline{x}_u$ denote the two vectors holding the available samples and the unknown samples, respectively. A single iteration of the GAPES algorithm consists of 3 steps:

1) Initialization.
2) Using APES to estimate $\left\{\hat{\alpha}\left(\omega\right)\right\}$ and $\left\{\underline{h}\left(\omega\right)\right\}$: The initial estimate, in the first iteration, uses available data only and the next iterations use the full data.
3) Estimating the lost samples, $\underline{x}_u$.

Initialization

Set the size K of the frequency grid $\{\omega_k\}_{k=0}^{K-1}$, where $\omega_k \triangleq \frac{2\pi}{K}k$. We chose K to be the closest integer power of 2 greater than the segment's length M.

APES estimation

The first thing GAPES does is to use the APES algorithm [9] in order to estimate the spectral coefficients $\{\hat{\alpha}(\omega_k)\}$ and the filter-bank $\{\underline{h}_k\}$. APES estimates these parameters according to a LS-criterion requiring that the filter's output will be as close as possible to a sinusoid of frequency ω_k in the LS sense.

$$\min_{\underline{h}_k,\alpha(\omega_k)} \frac{1}{M-P+1}\sum_{l=0}^{M-P}\left|\underline{h}_k^* \cdot \underline{y}(l) - \alpha(\omega_k)\cdot e^{j\omega_k l}\right|^2 \tag{B.1}$$

Where P is the length of each filter $\underline{h}_k$, M is the length of the data sequence and * denotes "conjugate-transpose".

The estimation process is as follows, except that a special treatment is given to the initial estimation (the first iteration) as will be specified at the end of this section.

1. Set the length P of the FIR filters that form the $\{\underline{h}_k\}$ filter-bank. The value of P is limited by the condition introduced in step 4, below.

2. Define $L = M$-P+1. The output of filtering an M-sample sequence by a FIR filter $\underline{h} \in \mathbb{C}^{P\times 1}$ can be written as $\underline{h}^*\mathbf{Y}$, where $\mathbf{Y} \in \mathbb{C}^{P\times L}$ is a matrix formed from the column vectors:

$$\underline{y}(l) = \left[x(l), x(l+1), \ldots, x(l+P-1)\right]^T \tag{B.2}$$

   Where $l = 0,1,\ldots,L$-1. **Y** divides the data samples of the sequence $\underline{x}$ into L overlapping vectors of size $P \times 1$ with a shifted structure.

3. Next, we calculate the sample covariance matrix $\hat{\mathbf{R}} \in \mathbb{C}^{P\times P}$, using:

$$\hat{\mathbf{R}} = \tfrac{1}{L}\left(\mathbf{Y}\cdot\mathbf{Y}^*\right) \tag{B.3}$$

   In order to get a full-rank covariance matrix $\hat{\mathbf{R}}$, P should satisfy the condition:

$$P \le L \Rightarrow \quad P \le \frac{M-1}{2} \tag{B.4}$$

4. Apply an FFT of the length of the frequency grid, K, over each of the rows of **Y** and divide the coefficients by L, to get the matrix $\mathbf{G} \in \mathbb{C}^{P\times K}$. Each of the columns of this matrix can also be written as:

$$\underline{g}_k = \frac{1}{L}\sum_{l=0}^{L-1}\underline{y}(l)\cdot e^{-j\frac{2\pi}{K}kl} \tag{B.5}$$

Where $0 \le k \le K$-1.

5. Next, for every k, such that $0 \le k \le K$-1, calculate the following:

   - Define $\omega_k = \frac{2\pi}{K}k$.

   - Define the vector $\underline{a}_k$ to be:

$$\underline{a}_k = \left[1, e^{-j\omega_k}, \ldots, e^{-j(P-1)\omega_k}\right]^T \tag{B.6}$$

   - The solution of the minimization problem in (B.1) is given by the following:
     The filter $\underline{h}(\omega_k)$ is calculated by:

$$\underline{h}(\omega_k) = \frac{\mathbf{R}^{-1}\cdot\underline{a}_k + \dfrac{\mathbf{R}^{-1}\underline{g}_k\gamma(\omega_k)^*}{1-\rho(\omega_k)}}{\beta(\omega_k) + \dfrac{\gamma(\omega_k)\gamma(\omega_k)^*}{1-\rho(\omega_k)}} \tag{B.7}$$

Where
$$\begin{aligned}\rho(\omega_k) &\triangleq \underline{g}_k^*\mathbf{R}^{-1}\underline{g}_k,\\ \beta(\omega_k) &\triangleq \underline{a}_k^*\cdot\mathbf{R}^{-1}\cdot\underline{a}_k,\\ \gamma(\omega_k) &\triangleq \underline{a}_k^*\cdot\mathbf{R}^{-1}\underline{g}_k\end{aligned} \tag{B.8}$$

The spectral coefficient $\alpha(\omega_k)$ is calculated by:

$$\begin{aligned}\hat{\alpha}(\omega_k) &= \hat{\underline{h}}^*(\omega_k)\cdot\underline{g}_k\\ &= \frac{1}{L}\sum_{l=0}^{L-1}\left[\hat{\underline{h}}_k^*\cdot\underline{y}(l)\right]\cdot e^{-j\frac{2\pi}{K}kl}\end{aligned} \tag{B.9}$$

Which can be viewed as the FFT of the filtered sequence (as explained in [10]).

Initial estimation

Article [10] offers two approaches for the initial estimation of APES. The first approach bases the estimation of $\{\hat{\alpha}(\omega_k)\}$ and $\{\underline{h}_k\}$ on available data only. The second approach uses all the data, where the values of the lost samples are set to zero. Our experience shows that the first approach performs considerably better than the second one, so we use it in our solution. However, in order to use the first approach we need to consider the loss pattern and the number of lost samples and choose the filter's length, *P*, so that it will allow for a full-rank covariance matrix to be built using only the available data. The condition that *P* should satisfy can be described as follows: After creating the $\mathbf{Y} \in \mathbb{C}^{P\times L}$ matrix (mentioned in step 3, above) the requirement is that $\mathbf{Y}$ will have at least *P* columns that contain only available samples. Note that there is no limitation on the location of these columns, i.e., the valid columns may include only samples from the past, or only samples from the future. Naturally, an estimation that is based on samples from both sides of the loss will usually achieve better performance.

Given that this condition is satisfied, steps 1-4 in the algorithm above are re-defined with some small modifications for the initial estimation:

1. Choose the value for P so the condition above will be satisfied. In our solution, since the samples arrive in pairs (each MP3 packet contains two frames), and since we use the past frames as available data, $P \geq 2$ always.

2. Define $L = M\text{-}P\text{+}1$. Build the $\mathbf{Y} \in \mathbb{C}^{P \times L}$ matrix, as was defined before, only this time - all the columns in $\mathbf{Y}$ that contain one or more missing samples are set to zero.

   Define L' as the number of columns that contain only available samples (i.e., the ones that weren't zeroed). The values of the resulting matrix in this special case are the same as if we were to remove the un-desired columns from $\mathbf{Y}$ matrix completely, and used $\mathbf{Y}' \in \mathbb{C}^{P \times L'}$ instead, since the zeroed elements are always multiplied by zero elements, so they are not considered in the averaging.

3. Next, we calculate the sample covariance matrix $\hat{\mathbf{R}} \in \mathbb{C}^{P \times P}$, using:

$$\hat{\mathbf{R}} = \frac{1}{L'}\left(\mathbf{Y} \cdot \mathbf{Y}^*\right) \tag{B.3*}$$

4. Calculate the matrix $\mathbf{G} \in \mathbb{C}^{P \times K}$, where its columns can now be written as:

$$\underline{g}_k = \frac{1}{L'}\sum_{l=0}^{L-1} \underline{y}(l) \cdot e^{-j\frac{2\pi}{K}kl} \tag{B.5*}$$

   Where $0 \leq k \leq K\text{-}1$.

Step 5 is the same in the first iteration as in the next ones.

Estimation of lost samples

After the filter-bank and the spectral coefficients are found, they can be used to estimate the missing samples. Under the assumption that the missing data $\underline{x}_u$ have the same spectral content as the available data $\underline{x}_a$, we can estimate $\hat{\underline{x}}_u$ from the condition that the output of the filter $\underline{h}(\omega_k)$ fed with the data sequence made from $\underline{x}_a$ and $\hat{\underline{x}}_u$ is as close as possible to the sinusoid $\hat{\alpha}(\omega_k)e^{-j\omega_k l}$ for $0 \leq l \leq L\text{-}1$ and $0 \leq k \leq K\text{-}1$.

The LS criterion can be written as:

$$\min_{\underline{x}_u} \sum_{k=0}^{K-1}\sum_{l=0}^{L-1}\left|\underline{h}_k^* \underline{y}(l) - \hat{\alpha}(\omega_k) \cdot e^{j\omega_k l}\right|^2 \tag{B.10}$$

The LS criterion above can be also written as:

$$\min_{\underline{x}_u} \sum_{k=0}^{K-1} \left\| \mathbf{H}_k \underline{x} - \underline{z}_k \right\|^2 \tag{B.11}$$

Where: $$\mathbf{H}_k = \begin{bmatrix} \underline{h}_k^* & & & 0 \\ & \underline{h}_k^* & & \\ & & \ddots & \\ 0 & & & \underline{h}_k^* \end{bmatrix} \in \mathbb{C}^{L\times P} \tag{B.12}$$

$$\underline{z}_k \triangleq \hat{\alpha}(\omega_k) \cdot \left[1, e^{j\omega_k}, \ldots, e^{j(L-1)\omega_k}\right]^T \in \mathbb{C}^{L\times 1} \tag{B.13}$$

Expressing $\mathbf{H}_k \underline{x}$ as $\mathbf{H}_k \underline{x} \triangleq \mathbf{A}_k \underline{x}_a + \mathbf{B}_k \underline{x}_u$, where the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are simply the matrices holding only the rows from $\mathbf{H}_k$ that multiply the samples in the vectors $\underline{x}_a$ and $\underline{x}_u$, respectively.

Defining:

$$\underline{d}_k \triangleq \underline{z}_k - \mathbf{A}_k \underline{x}_a \tag{B.14}$$

We can re-write the LS criterion as:

$$\min_{\underline{x}_u} \sum_{k=0}^{K-1} \left\| \mathbf{B}_k \underline{x}_u - \underline{d}_k \right\|^2 \tag{B.15}$$

The solution to this minimization problem is given by:

$$\hat{\underline{x}}_u = \left(\sum_{k=0}^{K-1} \mathbf{B}_k^* \mathbf{B}_k \right)^{-1} \left(\sum_{k=0}^{K-1} \mathbf{B}_k^* \underline{d}_k \right) \tag{B.16}$$

Some numerical stability precautions:

1. In cases where the available samples, surrounding the loss, are zero or very close to zero, as is often the case for high frequency coefficients (indexed around 420-576) the sample covariance matrix could become singular, and therefore cannot be inverted. In such cases we prefer to avoid estimation and simply set the lost values arbitrarily to zero.

2. In order to avoid cases of singularity or near-singularity, in the general case, a regularization matrix, εI , is added to the sample covariance matrix before inverting it.

REFERENCES

[1] J-C Bolot, H. Crepin and A.V. Garcia, "Analysis of Audio Packet Loss in the Internet", Proceedings of NOSSDAV 1995, Fifth Int. workshop on Network and Operating Sys. Support for Digital Audio and Video, April 1995, pp.163-17.

[2] J-C Bolot, "Characterizing End-to-end Packet Delay and Loss in the Internet", Journal of High-Speed Networks December 1993 Vol. 2, pp. 305-323.

[3] M. Handley, "An examination of MBone performance", USC/ISI res. Rep. ISI/RR-97-450, April 1997.

[4] D.J.Goodman, G.B.Lockhart, O.J.Wasem, W.Wong, "Waveform substitution techniques for recovering missing speech segments in packet voice communications", IEEE Trans. On ASSP, vol. 34, Dec. 1986, pp. 1440-1448.

[5] P. Lauber and R. Sperschneider, "Error Concealment for Compressed Digital Audio", AES 111th convention, September 2001, NY, pp. 1-11.

[6] S. Quackenbush and P.F. Driessen, "Error Mitigation in MPEG-Audio Packet Communication Systems", AES 115th convention, October 2003 NY, pp. 1-11.

[7] J. Lindblom and P. Hedelin, "Packet loss concealment based on sinusoidal extrapolation", ICASSP, May 2002, Vol. 1 pp. 173-176.

[8] Y. Wang, M. Vilermo, "Modified discrete cosine transform-its implications for audio coding and error concealment", AES Journal, vol.51, Jan.-Feb. 2003, pp.52-61.

[9] Petre Stoica, Hongblin Li and Jian Li, "A new derivation of the APES filter", IEEE Signal Processing Letters, August 1999, Vol.6 pp. 205-206.

[10] Petre Stoica and Erik G.Larsson, "Adaptive filter-bank approach to restoration and spectral analysis of gapped data", The Astronomical Journal (by the American Astronomical Society), October 2000, pp. 2163-2173.

[11] D.Y. Pan, "A tutorial on MPEG/audio compression", Multimedia, IEEE, Vol. 2, Issue: 2, Summer 1995, pp. 60-74.

[12] José M. Tribolet, "Frequency Domain Coding of Speech", IEEE Trans. On ASSP, Vol. 27, Oct. 1979, pp. 512-530.

[13] MPEG-1: Coding of moving pictures and associated audio for digital storage media at up to 1.5 Mbit/s, part 3: Audio. International Standard IS 11172-3, ISO/IEC JTC1/SC29 WG11, 1992.

[14] The LAME Project: An open source of an MP3 coder: http://lame.sourceforge.net