Packet Loss Concealment for Audio Streaming

Hadas Ofir

Packet Loss Concealment for Audio Streaming

Research Thesis

Submitted in Partial Fulfillment of The
Requirements for the Degree of
Master of Science
in Electrical Engineering

Hadas Ofir

Submitted to the Senate of
the Technion - Israel Institute of Technology

Sivan, 5766 Haifa June, 2006

The Research Thesis Was Done Under The Supervision of
Prof. David Malah in The Faculty of Electrical Engineering

Acknowledgements

I would like to express my gratitude to Prof. David Malah, for his devoted supervision and his professional guidance. Our fruitful discussions contributed immensely to the success of this research. Moreover, I wish to thank him for setting high standards and for supporting me throughout this research.

I wish also to show my appreciation to SIPL's staff: Nimrod, Yair, Ziva and Avi, for their help and technical support.

I would like to thank Dr. Erik Larsson, from George Washington University, Washington DC, for sending me his Matlab implementation of GAPES, which helped me develop the C implementation of the algorithm.

Many thanks to all my colleagues who participated in the listening tests.

Special thanks to my husband Barak for his support and patience, and to my family for their encouragement.

The Generous Financial Help of The Technion
is Gratefully Acknowledged.

Contents

List of Figures

List of Tables

Abstract

Internet audio streaming, based on MPEG-Audio coders such as MP3, has become very popular in recent years. However, since internet delivery doesn't guarantee quality of service, data packets are often delayed or discarded during network congestions, causing gaps in the streamed media. Each such gap, unless concealed in some way, produces an annoying disturbance. The common approach for dealing with such cases is to reconstruct the missing signal, approximating the original waveform, so that a human listener will not notice the disturbance. However, the gap created by even a single lost packet is relatively wide (around 1000 samples) and is therefore difficult to interpolate. Previous works start from simple techniques, such as noise substitution, waveform substitution and packet repetition, on to advanced techniques that use interpolation in the compressed domain for MPEG audio coders.

In this work we present a new algorithm for audio packet loss concealment, designed for MPEG-Audio streaming, based only on the data available at the receiver. The algorithm reconstructs the missing data in the DSTFT (Discrete Short-Time Fourier-Transform) domain using either GAPES (Gapped-data Amplitude and Phase Estimation) or MAPES-CM (Missing-data Amplitude and Phase Estimation - Cyclic Maximization) algorithms. The GAPES algorithm uses an adaptive filter-bank approach to estimate the spectral coefficients from the available data and then 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 MAPES-CM algorithm is a newer version, which uses an ML-estimator approach, and has slightly less complexity demands.

Since MPEG-Audio coders use the MDCT (Modified Discrete Cosine Transform) domain for compression, the data has to be converted first to the DSTFT domain. The conversion back and forth between the two domains is done using an efficient procedure that was also developed in this work.

The algorithm was subjectively evaluated by a group of listeners, and was found to perform better than previously reported methods, even at loss rates as high as 30%.

List of Symbols and Abbreviations

Symbols

$\underline{a}(\omega)$	Filter used for calculating DFT coefficient at frequency ω
A	Amplitude
$\underline{e}_l(\omega)$	Data snapshots of the residual terms
$e_n(\omega)$	Residual term (defined in eq. (5.1))
$E[\cdot]$	Expectation
g_d^1, g_r^1, g_d^2 and g_r^2	"Building block" functions
$\underline{h}(\omega)$	Data-dependent narrow-band filter, centered at frequency ω
$h[n]$	MDCT window function
$h_0(n)$	Prototype low-pass filter
$h_k(n)$	Impulse response of each sub-band filter
K	Size of frequency grid
K_0	Integer number, denoting a frequency bin
L	Number of overlapping data snapshot vectors, $\underline{y}_l$
M	Length of the $\underline{h}(\omega)$ filters
N	Number of bins in a single MP3 sub-band
N_{block}	Length of MDCT of specific block

N_s	Length of MDCT of 'short' block
p	Index of MDCT time-segments
P	Length of discrete-time data sequence
P_a	Number of available samples
P_m	Number of missing samples
$Pr\{\cdot\}$	Probability
Q	Number of consecutive lost packets
$\mathbf{Q}(\omega)$	Covariance matrix of $\underline{e}_l(\omega)$ vectors (defined in eq. (5.13))
$\hat{\mathbf{R}}$	Sample-covariance matrix of data snapshots $\underline{y}_l$
$\hat{\mathbf{S}}(\omega)$	(Defined in eq. (5.10))
$w[n]$	DSTFT window function
$\underline{x}_a$	Vector of available samples
$\underline{x}_m$	Vector of missing samples
$x[n]$ or x_n	Time-domain signal
$\hat{x}[n]$	Signal after inverse MDCT transform
$x_{(p)}[n]$	Sample in an MDCT segment
$X^{MDCT}[k]$	MDCT coefficient
$X^{DSTFT}[k]$	DSTFT coefficient
$\underline{y}_l$	Data snapshot (defined in eq. (5.3))
$\underline{Y}(\omega)$	Average DFT of the data snapshots, $\underline{y}_l$ (defined in eq. (5.8))
$\alpha(\omega)$	A spectral coefficient
δ	Constant time interval
$\delta[\cdot]$	Kronecker delta function
Δ_0	Fractional number, denoting the offset from frequency bin

ϕ_0	Phase parameter
ω	Frequency parameter
$[\cdot]^T$	Transpose
$[\cdot]^H$	Conjugate-transpose

Abbreviations

AAC	Advanced Audio Coder
APES	Amplitude and Phase Estimation
AR	Auto Regressive
CD	Compact Disk
CELP	Code Excited Linear Prediction
CM	Cyclic Maximization
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DSTFT	Discrete Short-Time Fourier-Transform
EM	Expectation Maximization
FEC	Forward Error Correction
GAPES	Gapped-data APES
HILN	Harmonic and Individual Lines plus Noise
HMM	Hidden Markov Model
IP	Internet Protocol
LS	Least-Squares
MAPES	Missing-data APES
MB	Mega Bytes
MDCT	Modified Discrete Cosine Transform
ML	Maximum Likelihood
MPEG	Moving Pictures Expert Group
MP3	MPEG-1 Audio Layer-III

MSE	Mean Square Error
NMN	Noise-Masking-Noise
NMR	Noise to Mask Ratio
NMT	Noise-Masking-Tone
OLA	Overlap and Add
PE	Perceptual Entropy
PEAQ	Perceptual Evaluation of Audio Quality
PLC	Packet Loss Concealment
QoS	Quality of Service
RTP	Real-Time Protocol
RTSP	Real-Time Streaming Protocol
SI	Statistical Interpolation
SMR	Signal to Mask Ratio
SPL	Sound Pressure Level
TCP	Transmission Control Protocol
TMN	Tone-Masking-Noise
UDP	User Datagram Protocol
VoIP	Voice Over Internet Protocol

Chapter 1

Introduction

1.1 Streaming Media and Packet Loss

With the growing popularity of the internet, and the advancement in modem technology there is an increasing interest in using the internet for broadcasting multimedia such as audio and video. This kind of delivery over the internet is referred to as *streaming media*, since the sound and picture data flows in a digital stream from a server computer to the client computer, ready to be heard or viewed in real time, without having to download all of the content before use.

Audio streaming operates by first compressing short segments of a digital audio signal and then gathering them 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 seamless play, the packets are "buffered" so 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 be built up, instead of waiting a long time for the files to be downloaded completely.

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, a gap is created in the streamed media and the client's audio player has nothing to play. Each such gap, 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. However, since typical audio packets correspond to 20-30 msec of audio, the gap created by even a single lost packet is relatively wide (882-1323 samples at 44.1 kHz sampling rate) and is therefore difficult to interpolate.

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. This work focuses on a receiver-based solution.

Previous works on receiver-based audio packet loss concealment [1, 2, 3] 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 modelling [7].

1.2 Research Goals and Main Results

The goal of this research work is to develop a new way to interpolate gaps created by lost packets in a streamed audio signal. The new solutions are designed for streaming applications that use MPEG-audio coders, where the implementations and simulations are performed using an MP3 coder.

The main results of this work include a new algorithm for packet-loss concealment, which reconstructs the missing data in the DSTFT domain, using an algorithm for reconstruction of complex signals: either GAPES [8] (Gapped-data Amplitude and Phase Estimation) or MAPES-CM [9] (Amplitude and Phase Estimation - Cyclic Maximization) algorithms. Also, Since MPEG-audio coders compress the data in the MDCT domain, we developed a new procedure for efficiently converting data from the MDCT domain to the DSTFT domain and vise versa.

The subjective experiments performed in this work show that the reconstruction is almost transparent for 10% loss rate, and yields good quality at higher loss rates, with an acceptable quality achieved even at 30%. Our tests also show that the reconstruction performs better than previously reported advanced concealment techniques such as Statistical Interpolation [6].

1.3 Thesis Outline

The thesis is organized as follows: Chapter 2 introduces the problem of packet loss in internet streaming audio applications, along with some statistics concerning the phenomenon. This chapter also explores some of the previous work in the area of packet loss recovery of audio signals.

Chapter 3 gives a short introduction to MPEG-audio coders, specifically the MP3 coder. This chapter focuses on the main features that were used in this work, such as time-frequency analysis. Readers who wish to learn more about the MP3 coder can find a detailed description of it, along with some general information about psychoacoustics, in Appendix A.

Chapter 4 examines the implications of packet loss on domains other than the MDCT, such as the time domain and the DSTFT domain and explores the limitations and benefits of interpolating the packet-loss gaps in different domains. Finally it explains our decision to conceal the packet loss in the DSTFT domain.

Chapter 5 described the theoretical background behind the APES [10] algorithm (Amplitude and Phase Estimation), that lead to the development of the GAPES [8] and the MAPES-CM [9] algorithms that are used in our concealment algorithm, as described in Chapter 6. Chapter 6 also describes the procedure that was developed in this work, for efficiently converting data between the MDCT and the DSTFT domains.

Chapter 7 describes and compares the results of the subjective tests that were performed on each of the proposed concealment methods, and Chapter 8 concludes the thesis.

Chapter 2

The Packet Loss Problem and Existing Solutions

Streaming media is currently poised to become the de facto global media broadcasting and distribution standard, incorporating all other media, including television, radio, and film. The low cost, convenience, worldwide reach, and technical simplicity of using one global communication standard makes web broadcasting irresistible to media distributers and consumers. There are currently more than a dozen formats for streaming audio over the Web, from widely used formats, such as RealNetworks' RealAudio, streaming MP3, Macromedia's Flash and Director Shockwave, Microsoft's Windows Media, and Apple's QuickTime, to more recent entries that synchronize sounds with events on a web page, such as RealMedia G2 with SMIL and Beatnik's Rich Music Format (RMF). Multimedia Streaming Services (MSS) will also be used in 3rd generation mobile networks.

2.1 Streaming Protocols

According to [11], the big breakthrough that enabled the streaming revolution was the adoption of a new internet protocol called the User Datagram Protocol

(UDP) and new encoding techniques that compressed audio files into extremely small packets of data. UDP made streaming media feasible by transmitting data more efficiently than previous protocols from the host server over the internet to the client player. More recent protocols such as the Real-Time Streaming Protocol (RTSP) are making the transmission of data even more efficient.

UDP is a *connectionless* transport layer protocol. Meaning that the communication occurs between hosts with no previous setup: packets that are sent between two hosts may take different routes. This provides a simple and unreliable message service for transaction-oriented services. Unlike the TCP, it adds no reliability, flow-control, or error-recovery functions to IP. Because of the UDP's simplicity, UDP headers contain fewer bytes and consume less network overhead than TCP and are therefore more suitable for real-time transmissions. UDP protocol is also referred to as *packet switching* and is used for optimization of the use of the bandwidth available in a network and to minimize the latency. Also, UDP and RTSP are ideal for audio broadcasting since they place a high priority on continuous streaming rather than on absolute document security. Unlike TCP transmission, when a UDP audio packet drops out, the server keeps sending information, causing only a brief glitch instead of a huge gap of silence. TCP, on the other hand, keeps trying to resend the lost packet before sending anything further, causing greater delays and breakups in the audio broadcast.

The RTSP protocol in specific is designed to support real-time traffic by establishing and controlling either a single or several time-synchronized streams of continuous media such as audio and video, and it also provides services such as sequence numbering, time-stamping and delivery monitoring (in order to facilitate quality-of-service monitoring). The RTSP protocol does not actually deliver the data, but works alongside existing delivery channels such as UDP, TCP, or

IP-multicast.

Regardless of the advances in the transmission protocols, streaming media would not be possible without the rapid innovation in encoding algorithms, that compress and decompress audio and video data. Uncompressed audio files are huge: One minute of playback of a CD-quality (16-bit, 44.1 kHz) stereo audio file requires around 9 MB of data (approximately enough disk space to capture a small library of books or a 200-page web site . . .). Standard modem connections don't have the capacity to deliver pure, uncompressed CD-quality audio in high speed. In order to stream across the limited bandwidth of the web, audio has to be compressed and optimized with lossy codecs, which reduce the file size by discarding some amount of data during the encoding process before it is sent over the internet.

In conclusion, the streaming internet protocols and the encoding procedures enable the fast transmission of media files over the internet. However, they do not ensure safe delivery of the packets nor do they provide other quality of service (QoS) guarantees, but relies on lower-layer services to do so.

2.2 Packet Loss Statistics

As was just mentioned, networks such as the internet do not provide guaranteed resources such as bandwidth or guaranteed performance measures such as maximum delay or maximum loss rate. There are some control mechanisms that support real-time applications over such networks, and they can help reduce packet losses, but they do not prevent it entirely. In fact, the experience accumulated over the internet indicates that audio packet losses are in large part responsible for the second-rate quality of many audio transmissions over the network.

Packet losses can arise from several reasons:

- Congestion on routers which creates buffer overflow, causing the routers to discard some of the packets.
- In real-time applications such as video-conferencing or streaming audio/video, a packet that is delayed too long would have to be discarded in order to meet the application's timing requirements, leading to the appearance of loss.
- Fading and interference in wireless links.

Reference [12] describes some characteristics of packet loss in audio streams sent over the internet. The authors measured the audio loss process over a number of connections (i.e. source-destination pairs) using the standard protocol stack of the internet, i.e. IP/UDP/RTP. The main result of this research was that the number of consecutively lost audio packets, i.e. the length of the gaps, is small, especially when the network load is low or moderate. This result also agrees with previous measurements obtained with non-audio UDP packets over different connections over the internet [13].

Fig. 2.1 shows two of the measurements that were taken during the multicast transmission. The upper graph represents the losses measured at 8:00 am, and the lower graph represents the losses measured at 4:00 pm. Obviously, the total number of losses is higher at 4:00 pm, when the network load is higher. Also, it can be easily seen that most losses are isolated and contain a small number of packets. This last observation is confirmed by the shape of the distribution function of the number of consecutive losses, appearing in Fig. 2.2.

Reference [13] also contains a study of the loss rates in such cases, as a function of the network load, expressed by the parameter δ , which indicates the constant interval between the send times of two successive UDP packets. The source sends probe packets at regular intervals (every δ msec), so naturally smaller δ means

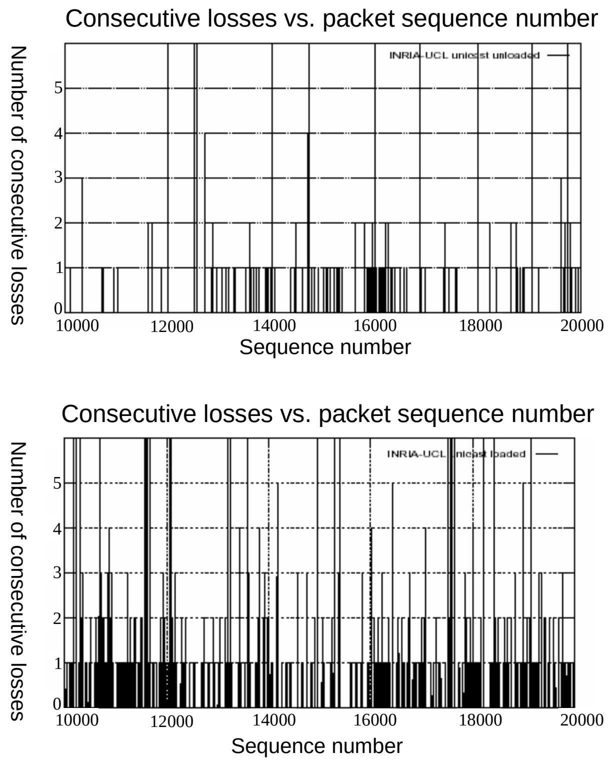
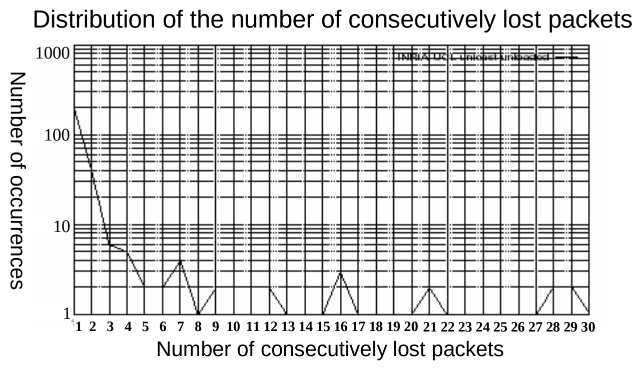


Figure 2.1: Evolutions of the number of consecutively lost packets at 8:00 am (top) and 4:00 pm (bottom).

higher network load and the load decreases as δ increases. The results of these experiments show that the packet losses are essentially random as long as the traffic uses less than 10% of the available capacity of the connection over which the packets are sent. At higher rates, the losses become more and more correlated, since the router is more likely to experience buffer overflows, and given that packet n is lost, (which means that the router's buffer is occupied upon arrival of packet n) there is a good chance that packet $n{+}1$ will be discarded too. Table 2.1 shows these results.

Regarding loss rates in general, reference [14] reports that according to their measurements, 50% of the receivers have a mean loss rate of about 10% or lower, and around 80% of the receivers had **some** interval of the day where no loss was



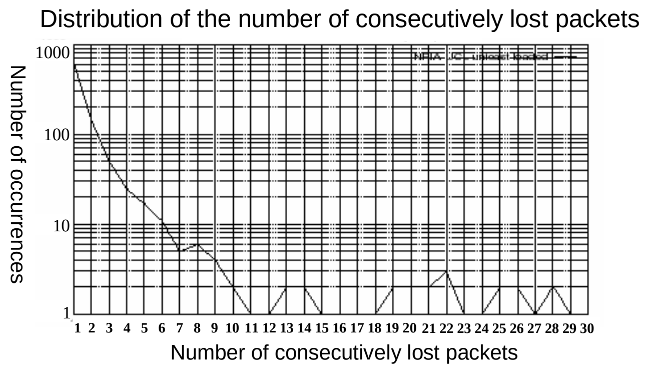


Figure 2.2: Distribution of the number of consecutively lost packets at 8:00 am (top) and 4:00 pm (bottom).

observed. However, 80% of the measured sites reported also some interval during the day when the loss rate was greater than 20%, which is generally regarded as the threshold above which audio (speech in specific) becomes unintelligible and intolerable.

In conclusion, we can safely assume that most of the audio-streaming clients will experience some amount of packet loss during the connection: When the network's load is low it will be a small loss rate (below 10%) with mostly isolated gaps. When the network's load is high the gaps will be more frequent and wide, which can even lead to the all-too-familiar connection drop-out that every internet user has encountered.

Table 2.1: Variations of conditional and unconditional loss probability for different values of δ

δ (msec)	8	20	50	100	200	500
Unconditional loss probability: $Pr\{$ packet n is lost $\}$	0.23	0.16	0.12	0.10	0.11	0.09
Conditional loss probability: $Pr\{$ packet n+1 is lost / packet n is lost $\}$	0.6	0.42	0.27	0.18	0.18	0.09

2.3 Packet Loss Recovery Techniques

Although most packet losses are relatively small, each instance of loss, unless concealed in some way, produces an annoying disturbance. This is because the human hearing system is sensitive to the discontinuity that is created in the waveform by lost packet. Ignoring the gap and splicing the ends of the gap together is not an acceptable solution since it introduces a timing mismatch in the playback buffer, and might interfere with server-client synchronization in real time applications. Therefore, for isolated packet losses and certainly for multiple consecutive packet losses, the gap needs to be filled-in with samples approximating the original waveform.

There are numerous techniques for packet loss recovery [1, 2, 3], mostly designed for speech signals. The various techniques can be roughly divided into two categories: sender-based and receiver-based [2]. All techniques require the help of the decoder at the receiving side, but the sender-based techniques also use side-information from the encoder or change the bitstream format to improve the results of the concealment process. Some of the sender-based techniques perform *error correction*, creating an exact repair of missing packets, with no loss of the signal's audio quality, while all the receiver-based techniques (and also some of the sender-based techniques) perform *error concealment*, where only an approximation of the missing packets is achieved, resulting in some quality degradation: When designing a receiver-based algorithm for error concealment, 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. Another point, which is not treated in this work, is that the reconstructed signal might be used as input for more post processing, as in the case of speech recognition [15].

Since this research focuses on a receiver-based solution, the receiver-based category is explored in details in the next sub-section.

In the sender-based category, one can find the *Forward Error Correction* (FEC) mechanism which is based on the transmission of redundant information alongside with the original information, so that at least some of the lost data can be recovered from the redundant information. The main flaw of this technique is that it increases the network's congestion, leading to a worsening of the problem which it was originally intended to solve. Other techniques in this category are *Layer Encoding*, which is included in the new HILN standard for MPEG-4 parametric audio coding tool [16] and *Interleaving*, where basic data units (packets or even data samples [17]) are reorganized before transmission and then returned to their original order at the receiver. This operation enables better handling of burst-losses, since after the reordering, a wide gap of several lost data units turns into several smaller gaps which are easier to handle.

2.3.1 Receiver-Based Techniques

As was already mentioned, receiver-based techniques are usually less effective than sender-based techniques, since they don't have any information from the encoder to rely on, besides the corrupted bit stream itself. Most of the work made in this area was related to speech signals (concerning VoIP or video conferencing applications), and only recently articles are being published about work related to general audio signals.

The different algorithms can be divided into 3 main categories, ordered with increasing complexity and performance:

Insertion-based techniques: These techniques repair losses by inserting a fill-in packet. The creation of this fill-in is a very simple data-independent procedure, which doesn't involve any signal processing. *Silence* or *Noise Substitution* are very common, as is the *Repetition* of previous packet, which is suggested in the MP3 standard [18]. These schemes are very easy to implement but give relatively poor performance, which makes them appropriate only to very small loss rates (up to 5%) and unacceptable for higher loss rates.

Time-Domain Interpolation-based techniques: The concealment techniques in this category attempt to replace the lost packet by a similar copy, which is achieved using time-domain interpolation based on the packets surrounding the loss. The advantage of this category over the previous one is that it takes into account the content of the lost segment when trying to recover it, thus it achieves better performance (reasonable quality up to 10% loss rate), though at the expense of higher complexity. Among the techniques in this category are *Waveform Substitution* [4], *Pitch Waveform Replication* [19, 20], and *Time-Scale Modification* [21, 22].

Compressed-Domain, Model-based Interpolation techniques: The model-based interpolation techniques usually involve a great deal of signal processing and the use of the statistics and models that exist for audio signals (especially for speech signals). As a result, these techniques are usually higher in complexity but also achieve better performance. In many cases these techniques are applied in the compressed domain, instead of on the waveform samples. Among the techniques in this category are: *Interpolation Using Compressed Domain Parameters* for CELP speech coders [23, 24, 25] and for MPEG audio coders [6], and *Model-based Interpolation* [17, 26, 7]. In work [17], the authors represent speech signals using an AR model and estimate the missing samples based on a multi-rate state-space representation of the AR process. Work [26], which also refers to speech, uses an HMM model to estimate the lost signal, and in [7] general audio signals are represented using a sinusoidal model which is also used for reconstruction of the lost data.

As described in Chapter 7, the algorithm developed in this work was compared with two previously reported concealment techniques: *Packet Repetition* [18] and *Statistical Interpolation* [6], therefore we chose to describe these techniques next:

The packet repetition technique is simply replacing the lost packet with a copy of the packet that has arrived last. This operation usually maintains the statistical characteristics of the waveform (especially in parametric coders, such as CELP speech coders or HILN audio coder) since speech and audio signals are in many cases quasi-stationary, but there are times where it will perform poorly, for example in cases where there is a rapid change in the signal's statistics like in the case of voiced-unvoiced transition in a speech signal, or the appearance of a fast transient in audio signals.

The idea behind the statistical interpolation technique is to apply a samples-restoration algorithm, originally intended for time-domain signals, on the MDCT domain coefficients which are also real-valued. The advantage of this approach is that a gap of thousands of samples in time domain (for one or more lost packets) is translated into a smaller gap, which is easier to handle, of only few samples at each frequency bin in the MDCT domain along time (this concept is further explained in section 3.2.2). Another advantage to this approach is that it is not necessary to estimate all the bins, since in a typical music piece most of the energy is located in the lower frequency bins, where the highest bins are usually very close to zero, and therefore can be ignored. There are also some limitations to this approach: First, an estimation error in a single frequency bin will affect all the time samples in that packet, due to the inverse-MDCT procedure that is performed after the restoration (during the decoding process). The second limitation is the problem of handling different frequency resolutions in consecutive packets, due to the fact that MPEG-Audio coders use different window types (this problem is explained in section 4.1).

The statistical interpolation algorithm (or in short, SI) assumes that the MDCT coefficients along time, for a given frequency bin, are a realization of an auto-regressive (AR) process. For a given frequency bin, a missing coefficient is estimated by a linear combination of the available coefficients at the same frequency bin, taken from surrounding packets.

The restoration algorithm is an iterative algorithm, consecutively repeating the following two steps:

- Estimate the AR coefficients vector from all the set of samples: available and estimated ones (in the first iteration the values of the missing samples are arbitrarily set to zero).

- Estimate the set of lost samples based on the AR coefficients that were calculated in the previous step.

The restoration algorithm is computationally efficient for small loss intervals: The vector of AR coefficients is determined so it minimizes the expected variance of the statistical restoration error using the Levinson-Durbin algorithm, and the rest of the calculations use small matrices. For losses of more than 5 consecutive packets, the authors suggest using simple repetition (with attenuation applied) since the computational overhead becomes high and the poor results of the SI algorithm in that case do not justify it. However, the algorithm has also some limitations: For an AR model of order p, the algorithm requires at least p available samples at the ends of the segment, in order to get a meaningful estimation of the AR coefficients using the covariance or autocorrelation method, which cannot be guaranteed in random packet losses. And finally, there is no evidence that MDCT coefficients of audio signals behave according to AR models, and therefore there is no justification for choosing this model over other possible models.

The complexity of the SI algorithm is $O(L^2) + O(P_m^3)$ where L is the AR order and $P_m = 2Q$ is the number of missing samples.

Chapter 3

MPEG-Audio Coding

In recent years, MP3 audio 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 [27] 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.

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 audio quality. The most interesting among them is MPEG-1 Layer 3, a.k.a. MP3.

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 the characteristics and limitations of the human auditory system, such as 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 by a human listener. Needles to say, that the quality of the psychoacoustic model has a great impact on the quality and efficiency of the encoding process.

This chapter explores the MP3 coder, mainly focusing on the time-frequency mapping, since it is relevant to this work. The interested reader can find more about the MP3 coding process and psychoacoustics in general, in Appendix A.

3.1 The MP3 Coder

3.1.1 Encoder

Figure 3.1 in the next page, is a block-diagram of the MP3 encoding procedure, which is described next:

The signal in the input of the encoder is divided into frames of 576 samples each. Each such frame 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-band signals.

Parallel to this, the psychoacoustic model is applied on the signal in order to calculate, for each frame, the SMR in every critical band of the human auditory system. The model also determines for each frame the type of window to be used

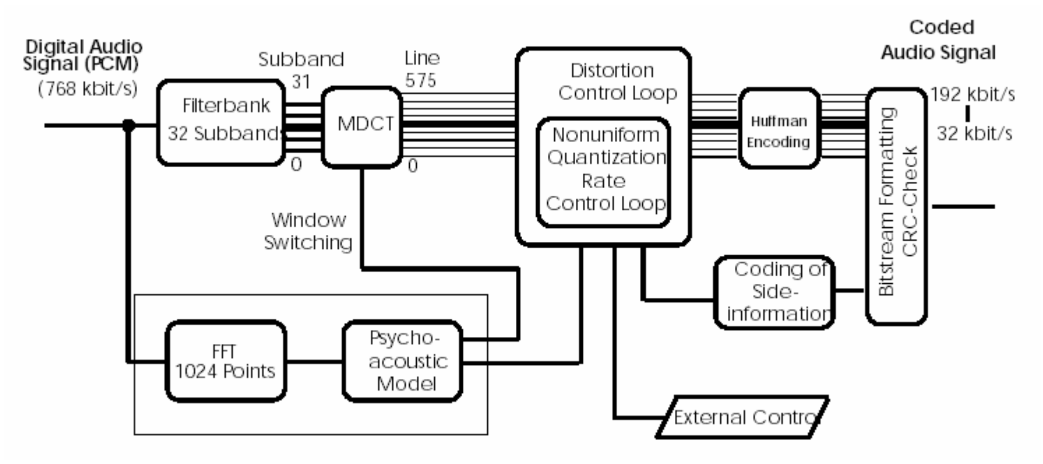


Figure 3.1: Block diagram of an MP3 encoder

in the MDCT process, according to the frame's short- or long-term characteristics (this is further explained in section 3.2.2).

Next, the 576 MDCT coefficients (one for each frequency line) 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 other 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**.

3.1.2 Decoder

The block diagram of a typical MP3 decoder appears in Fig. 3.2. The decoder deciphers the coded bit-stream, restoring the quantized MDCT coefficient values and the side information related to them, such as the window type that was assigned to each frame. After all this data is available, the coefficients are transformed back to the sub-band domain by applying an inverse-MDCT on the coefficients of each of the equal-width sub-bands. Finally, the waveform samples are reconstructed using filter bank summation on all the sub-bands.

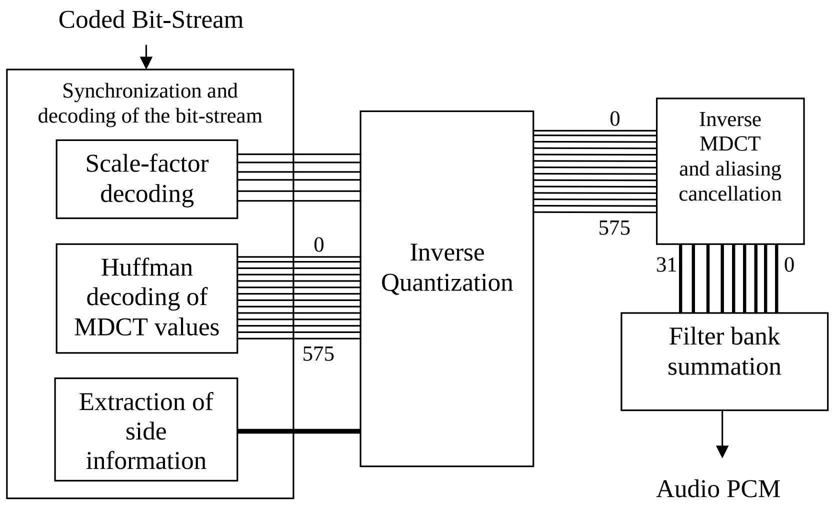

Coded Bit-Stream
Synchronization and decoding of the bit-stream
Scale-factor decoding
Huffman decoding of MDCT values
Extraction of side information
0
575
Inverse Quantization
0
575
Inverse MDCT and aliasing cancellation
31
0
Filter bank summation
Audio PCM

Figure 3.2: Block diagram of the MP3 decoder

3.2 Time-Frequency Mapping in the MP3

The filter bank that is used in the MP3 coder belongs to the class of *hybrid filter banks*. It is built by cascading two different kinds of filter banks: first a polyphase filter bank and then an additional Modified Discrete Cosine Transform (MDCT).

3.2.1 Polyphase Filter Bank

The polyphase filter bank is common for all layers of MPEG-1 audio compression. Polyphase structures are computationally very efficient because a DCT can be used in the filtering process (or DFT, in the complex case), and they are of moderate complexity and low delay (more about the implementation is in appendix A) . The polyphase filter bank divides the audio signal into 32 equal-width frequency sub-bands, where for every frame of 576 samples, the output of the filtering process is 18 new samples for each of the 32 sub-bands.

Each filter originally has 512 coefficients, where the impulse response of each sub-band filter, $h_k(n)$, is obtained by multiplying the impulse response of a single *prototype low-pass filter*, $h_0(n)$, by a modulation function which shifts the low-pass response to the appropriate sub-band frequency range:

$$h_k(n) = h_0(n) \cdot e^{j \cdot \varphi(k,n)}, \quad 0 \leq k \leq 63 \tag{3.1}$$

Where in the case of the MP3 standard, $h_0(n)$ is a simple filter in the shape of a *sinc* function.

This is the way to build a bank of 64 complex filters. In order to get real-valued filters, every two symmetric complex filters are added together. Since every two symmetric complex filters are the complex-conjugate of one another, the imaginary part vanishes and we are left with 32 real-valued filters:

$$\tilde{h}_k(n) = h_k(n) + h_{63-k}(n) = h_0(n) \cdot \cos[\varphi(k,n)], \quad 0 \leq k \leq 31 \tag{3.2}$$

In the case of MPEG-1:

$$\varphi(k,n) = \frac{(2k+1) \cdot (n-16) \cdot \pi}{64}, \quad 0 \leq k \leq 31 \tag{3.3}$$

The resulting filters have center frequencies at odd multiples of $\frac{\pi}{64}f_s$ where f_s is the sampling frequency, and each has a nominal bandwidth of $\frac{\pi}{32}f_s$. Although the prototype filter is relatively simple, it provides good time resolution with reasonable frequency resolution. However, the design of the polyphase filter bank has three notable disadvantages [27]:

- The filter bank and its inverse are not lossless transformations. Even without quantization, the inverse transformation cannot perfectly reconstruct the original signal. However, the design of the prototype filter assures that the error introduced by the filter bank is small and inaudible.
- Consecutive filter bands overlap at the edges, so a signal at a single frequency can affect two adjacent filter bank outputs. This effect can harm the efficiency of the compression since one signal might be "compressed twice". The fact that the filters overlap also introduces aliasing, since the filter outputs are decimated in the critical rate. The MP3 standard specifies a method of post-processing the MDCT coefficients, before the quantization, in order to cancel that aliasing completely.
- The equal widths of the sub-bands do not accurately reflect the human auditory system's critical bands: at lower frequencies a single sub-band covers several critical bands, while at higher frequencies one critical band can spread over a number of sub-bands (see more about this in appendix A). For that reason, Layer 3 extended the frequency partition further by using an MDCT transform in each of the sub-bands. This partition results in 576 frequency bins, which are later grouped according to the critical bands of the human auditory system for the quantization process.

3.2.2 MDCT

The MDCT transform is defined as:

$$X^{MDCT}[k] = \sum_{n=0}^{2N-1} x[n] \cdot h[n] \cdot \cos\left(\tfrac{\pi}{N} \cdot \left(n + \tfrac{N+1}{2}\right) \cdot \left(k + \tfrac{1}{2}\right)\right), \quad 0 \leq k \leq N-1 \quad (3.4)$$

where $x[n]$ is the original signal and $h[n]$ is a window function.
The inverse transform is defined as:

$$\hat{x}[n] = \frac{2}{N} \sum_{k=0}^{N-1} X^{MDCT}[k] \cdot \cos\left(\tfrac{\pi}{N} \cdot \left(n + \tfrac{N+1}{2}\right) \cdot \left(k + \tfrac{1}{2}\right)\right), \quad 0 \leq n \leq 2N-1 \quad (3.5)$$

From the definitions it can be seen that $2N$ samples in the time-domain are transformed into only N real-valued coefficients in the MDCT domain. Hence it is a lossy transform, and the output of the inverse transform includes aliasing in the time-domain:

$$\hat{x}[n] = \begin{cases} x[n] \cdot h[n] - x[N-n-1] \cdot h[N-n-1], & 0 \leq n \leq N-1 \\ x[n] \cdot h[n] + x[3N-n-1] \cdot h[3N-n-1], & N \leq n \leq 2N-1 \end{cases} \quad (3.6)$$

In order to perfectly reconstruct the signal, this aliasing effect has to be cancelled. Aliasing cancellation can be achieved when the MDCT transform is applied along time, on segments that have 50% overlap between them (as shown in Fig. 3.4) and under certain conditions:

1. The analysis window and the synthesis window are the same window, $h[n]$.
2. The window is symmetric: $h[n] = h[2N-1-n]$.
3. $h^2[n] + h^2[n+N] = 1$.

For example, one of the window functions that the MP3 uses, for which the conditions above hold, is the sinusoidal window function in (3.7), also presented in Fig. 3.3.

$$h[n] = \sin\left(\frac{\pi}{2N} \cdot \left(n + \frac{1}{2}\right)\right), \quad 0 \leq n \leq 2N - 1 \tag{3.7}$$

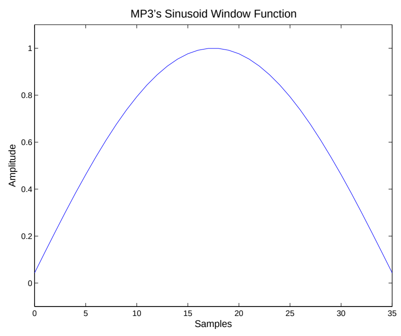


Figure 3.3: MP3's sinusoidal window

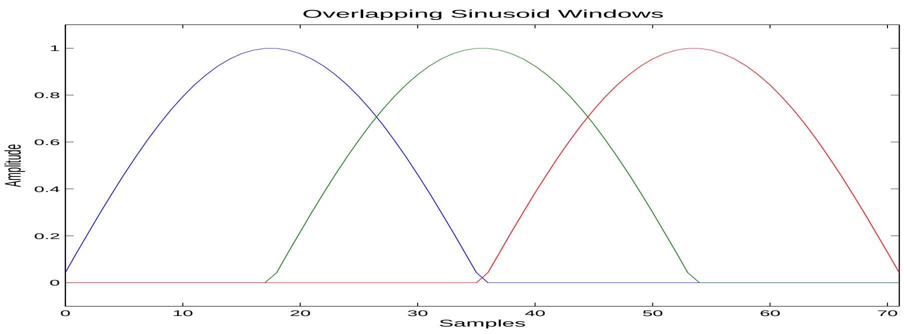


Figure 3.4: 50% Overlapping sinusoidal windows

The aliasing cancellation is achieved in the following manner, also shown in (3.8): Let's denote each time-segment by the index p, where $p \in \mathbb{Z}$. After the inverse-MDCT is applied, the $2N$ aliased samples of segment p are multiplied by the window function, $h[n]$. Then, each half-segment is merged with the overlapping section in the neighboring segment (the time-segments indexed $(p-1)$ and $(p+1)$) using overlap-and-add (OLA) in order to restore the original samples. Hence, we need a total of 3 consecutive MDCT frames in order to perfectly reconstruct a whole segment of $2N$ samples.

$$x_{(p)}[n] = \begin{cases} \hat{x}_{(p-1)}[n+N] \cdot h[N-n-1] + \hat{x}_{(p)}[n] \cdot h[n] & 0 \leq n \leq N-1 \\ \hat{x}_{(p)}[n] \cdot h[2N-n-1] + \hat{x}_{(p+1)}[n-N] \cdot h[n-N] & N \leq n \leq 2N-1 \end{cases} \tag{3.8}$$

The next section explains the logic behind the aliasing cancellation procedure: From the expression in (3.6) it can be seen that the aliasing in the first half of the inverse-transformed samples is **independent** of the aliasing in the second half. This property is important, since it allows to change the shape of the next window, without influencing the values in the current one (this concept is clarified in the sequel).

Because of the 50% overlap, one segment of N samples participates in the MDCT of two consecutive frames. Hence, each such segment is expressed twice after the inverse-MDCT transform: Once as the last N samples of the first window, $\hat{x}_{(p)}[N : 2N-1]$, and thereafter as the first N samples of the next window, $\hat{x}_{(p+1)}[0 : N-1]$. Looking again at (3.6) it can be seen that $\hat{x}_{(p)}[N : 2N-1]$ is the **sum** and $\hat{x}_{(p+1)}[0 : N-1]$ is the **difference** of the same N samples, where in each case the signal is multiplied by different window sections. This concept is illustrated in Fig. 3.5.

Since the window function $h[n]$ is symmetric (condition No. 2), the alias cancellation is achieved by an overlap-and-add procedure: Each aliased output segment is multiplied by its matching half-window and the two overlapping results are added together. The Mirrored parts (see Fig. 3.5), which are identical - only with opposite signs, cancel each other. Condition No. 3 ($h^2[n] + h^2[n + N] = 1$) guarantees that the effect of the window multiplication is cancelled too, leaving the original samples restored.

More information about the MDCT can be found in [28, 29].

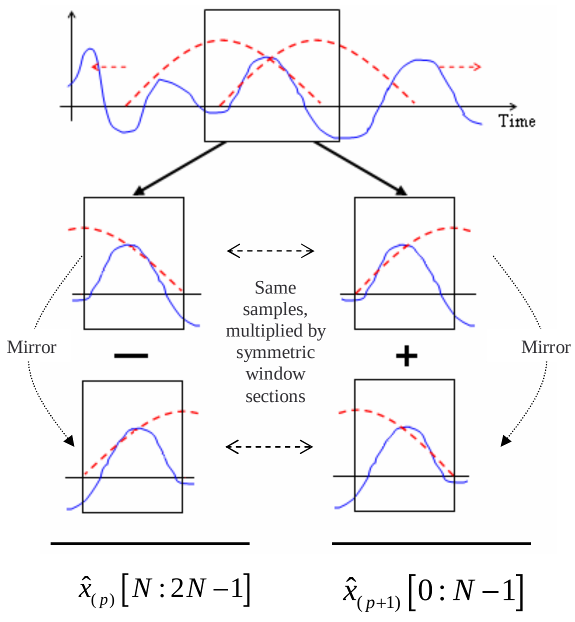


Figure 3.5: Overlapping section represented as aliased samples in the two consecutive windows

Using MDCT With Different Window Functions

As was already mentioned, the fact that the aliasing is independent for each half of the MDCT segment is important: In the case of MP3, this fact enables to dynamically change the time-frequency resolution.

In order to use different window functions for different segments, the conditions for perfect reconstruction are expanded to the general case: Let us assume that there are two consecutive overlapping window functions: $h[n]$ and $g[n]$. The conditions for perfect reconstruction in this case are the following:

1. The analysis window and the synthesis window of a certain segment are the same window.

2. $h[n+N] \cdot h[2N-n-1] = g[n] \cdot g[N-n-1]$.

3. $h^2[n+N] + g^2[n] = 1$.

Note that for the particular case where $h[n] \triangleq g[n]$, the conditions stated earlier agree with the ones defined here.

The MP3 standard defines four different types of window functions, shown in Fig. 3.6, in order to match the different nature of each coded frame:

Long (type 0) a long sinusoidal window, which provides better frequency resolution for stationary frames.

Start (type 1) a transition window, to be used after a Long window and before a Short window.

Short (type 2) provides better time resolution, for frames containing rapidly varying signals or fast transients. This window is actually built of 3 short sub-windows.

Stop (type 3) a transition window, to be used after a Short window and before a Long window.

These four window types are designed to satisfy the last two conditions, so perfect reconstruction is guaranteed, and they enable the coder to change the time-frequency resolution of the windows during the process, in order to get a better representation of the coded signal. However, in order to make sure that the two conditions are satisfied, a transition window must always come between Short and Long windows. The different windows are presented in Fig. 3.6 and a possible ordering of the windows is shown in Fig. 3.7.

The windows of type 0,1 and 3 use a long MDCT transform, turning 36 samples into 18 coefficients. The Short window (type 2) uses 3 short MDCT transforms, where 12 samples are transformed into 6 coefficients: a short transform on each short sub-window. In total, all the windows have a constant length of $2N = 36$ samples and result in $N = 18$ coefficients after the MDCT transform is applied.

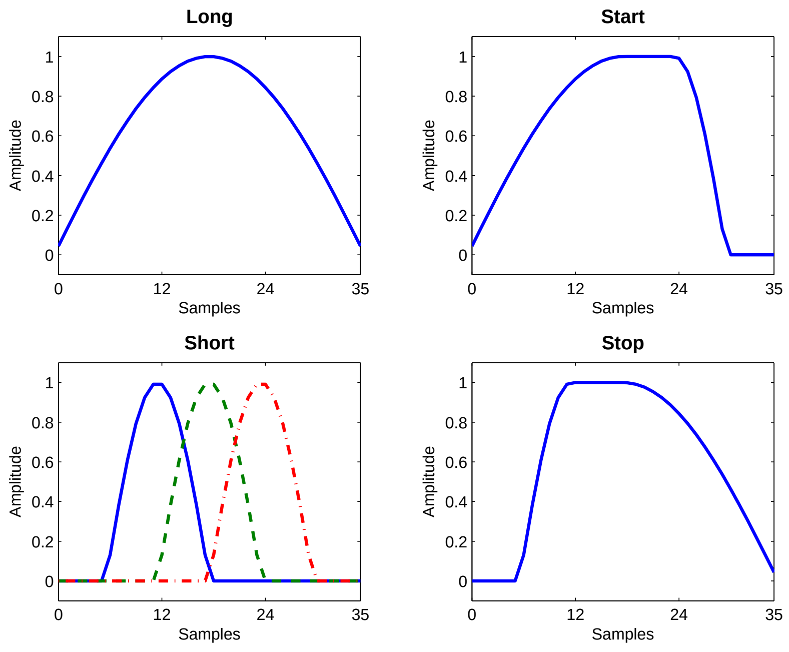


Figure 3.6: The four window types defined by the MP3 standard

As can easily be seen, these four different window functions are actually built from only 3 basic half-window units, described in Fig. 3.8, that are used in direct or reverse form. This fact is exploited in this work, as described in section 6.1. Note that the 3 half-window units are of different lengths: the 'long' and 'short-

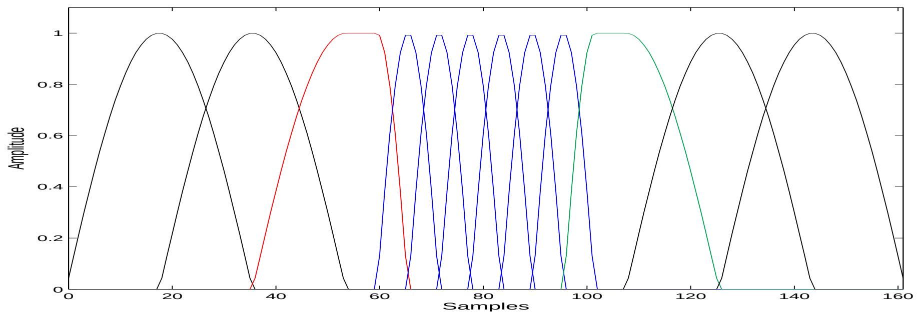


Figure 3.7: A possible ordering of the different windows

to-long' halves are 18 samples long, where the 'short' half is only 6 samples long. The middle section of the 'short-to-long' half is identical to a 'short' half.

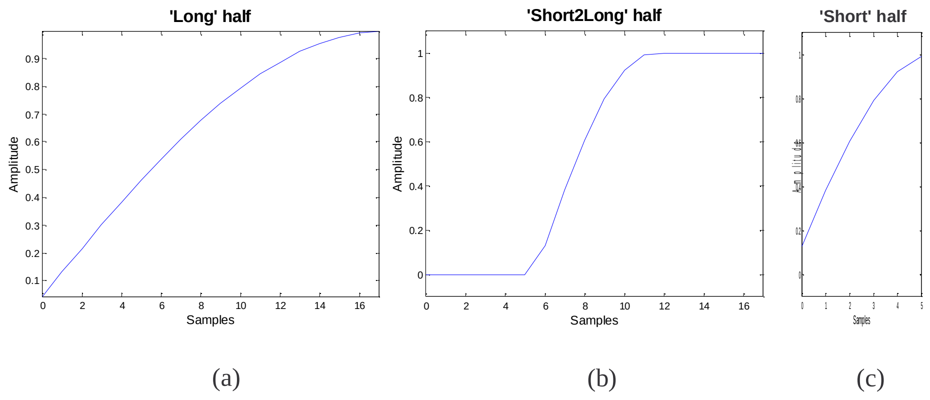


Figure 3.8: The three basic half-window units: (a) long, (b) short-to-long, (c) short

The use of MDCT in the MP3 standard

In the MP3 encoder, the MDCT is performed on the output samples of each of the sub-band filters, taking overlapping segments of 36 samples in the sub-band domain: the 18 outputs of the current analysis frame and another 18 are the output of the previous frame. Every segment of 36 samples is multiplied by its best-suiting window (the decision of the suitable window type is made by the psychoacoustic model, see appendix A) and then it is transformed to the MDCT domain, resulting in 18 frequency coefficients. Since there is a 50% overlap, the total number of MDCT values is hence the same as the number of samples. The MDCT coefficients are then quantized and sent to the decoder.

In the decoder, the values of the MDCT coefficients are restored, and an inverse transform is performed on each sub-band, after which the sub-band samples are restored by using the aliasing-cancellation OLA procedure that was described before. Finally, the waveform samples are reconstructed using filter-bank summation on all the sub-bands.

It's worth noting that since quantization is performed, the reconstruction in the decoder's side is **not** perfect, and the restored signal might suffer from an effect called "pre-echo", due to the fact that it still contains some aliased quantization noise that was not cancelled.

Chapter 4

Choosing the Concealment Domain

As was already described in section 2.3, there are many techniques for packet loss concealment. The most intuitive approach is to use interpolation in the time-domain, i.e., interpolate the waveform samples. However, in some cases there might be a benefit in interpolating the data in other domains, such as in the compressed domain, or the spectral domain. This chapter deals with the subject of choosing the interpolation domain that best suits this work: The first section presents the benefits and limitations of interpolating in each domain. The second section presents an analysis of pure sinusoidal signals in both the MDCT and the DSTFT domains, since our solution is implemented in these domains.

4.1 Different Concealment Domains

In the case of audio coding, two concealment domains come immediately to mind: the time-domain, and the compressed domain. In the case of MPEG-audio coders, the audio signal is compressed 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 a single lost MP3 packet creates a gap of at least 1152 samples: Considering also the fact that due to the loss, the aliasing in the segment preceding the loss cannot be cancelled and therefore this segment is corrupted, then the gap actually becomes of 1728 samples. This is quite a wide gap and is hence very difficult to handle. However, looking at the loss from the point-of-view of the MDCT domain, a loss of a single packet is translated to a gap of only two coefficients per frequency bin. Since such a gap is easier to interpolate, it seems reasonable to prefer doing the interpolation in the MDCT domain, rather than in the time domain, as suggested in [6]. In that work, the missing MDCT coefficients were reconstructed separately for each frequency bin, based on the coefficients available from neighboring packets at the same frequency (see section 2.3.1 for more details about that work).

Working in the MDCT domain also introduces some difficulties, compared to working in the time domain: The main problem arises from the fact that the MPEG-Audio coders use different type of window functions during the MDCT transform, in order to allow for a better representation of the signal's characteristics (as described in section 3.2.2). When a packet is lost, the information about the window type is lost too. Using the incorrect type of window will prevent the aliasing from being cancelled which will result in undesired artifacts, in addition to those introduced by the reconstruction process. However, in some cases this problem can be overcome: As was already explained in section 3.2.2, switching between Long and Short window types is not instantaneous and requires a transition window to come between them. This fact can be utilized 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.

In other cases, where there is an ambiguity regarding the type of the windows that were originally used, it is enough to choose a window type that will comply with the packets available at the edges of the loss. For example: **Long-missing$_1$-missing$_2$-missing$_3$-missing$_4$-Long** could arise from a sequence of **6 consecutive Long** windows, or from the following pattern: **Long-Start-Short-Short-Stop-Long**, or from other possible patterns. Since the first half of the Start window is identical to the first half of the Long window, choosing either one of the possibilities guarantees that the aliasing in the frame preceding the loss will be cancelled correctly (assuming of course, that we achieved perfect reconstruction of the lost MDCT coefficients of the first lost frame). Assuming that the receiver doesn't have any side information regarding the characteristics of the signal that was represented by the lost packets, it is not capable to prefer one possibility over the other, hence its decision will be arbitrary.

There are two more limitations to the MDCT domain which can't be overcome that easily. The first one also results from the use of different window functions: Since different window types have different frequency resolutions, the MDCT coefficients of two consecutive segments at a certain frequency bin 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.

The second limitation is the fact that the MDCT coefficients typically show rapid sign changes from frame to frame, at each frequency bin, which makes it more difficult to interpolate them. These sign changes reflect phase changes in the complex spectral domain [30].

A possible way to cope with the first limitation above 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. A solution to the second limitation would be to work in a domain that has a less fluctuating representation of the signal, thus providing better interpolation results. Interpolation in the Discrete Short-Time Fourier-Transform (DSTFT) domain overcomes both limitations and therefore was chosen as our concealment domain.

In order to work in the DSTFT domain one has to first convert the data from the domain it is compressed in: in this case, the MDCT domain. The simplest way to do this is by first converting the data back to the time domain, using an inverse-MDCT transform and an OLA procedure on the MDCT coefficients in each sub-band, and to then transform the resulting samples in each sub-band to the Fourier-domain using a window function of fixed length.

For the purpose of this work, it makes sense to use the same time-segments originally used by the MDCT transform, so the data is equally represented in both domains. In the case of MP3, this means segments of $2N = 36$ samples from each sub-band, overlapping by 50%. As was mentioned before, the benefits of using this domain are that it has a smoother representation of the signal, and that by using a window function of constant length it is guaranteed that all the coefficients have a fixed resolution. However, the conversion also introduces a few limitations: First of all, the procedure itself adds some complexity to the decoder, which is problematic. For this reason we developed a computationally efficient procedure for the conversion, which is described in section 6.1. Second, due to the overlap between the MDCT segments, a single segment of $2N$ samples is reconstructed from 3 consecutive MDCT frames, as shown in Fig. 4.1. In the case of packet loss this means that a lost frame affects the reconstruction of not only its corresponding segment, but its two closest neighbors too. The result is that the gap at each frequency bin in the DSTFT domain is bigger than the gap in the MDCT domain by two frames, since the neighbors at the edges of the gaps can't be properly restored. This concept is illustrated in Fig. 4.2.

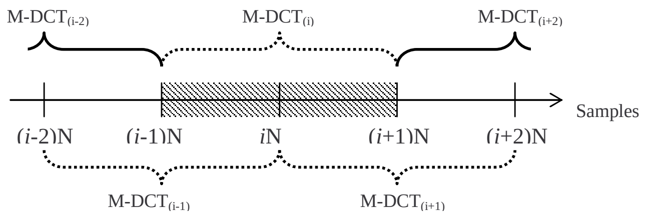


Figure 4.1: One segment of $2N$ samples is reconstructed using the relevant MDCT segment and also the MDCT segments on both its sides (in dotted brackets)

There is one more option that should be mentioned regarding MP3 coding: It is also possible to work in the semi-compressed domain, i.e., in the sub-band domain. The advantage of this domain is the same as in the time domain, only the gaps at each sub-band are shorter. Also, since the signal in each band represents a narrower spectrum, it is spectrally less complicated than the general signal and can be assumed to approximately comply with simpler models, such as the AR model [5]. The disadvantages in this case are that although the gap in each sub-band is smaller than in the time-domain, is still relatively wide for interpolation, and more important - that a concealment algorithm designed for this domain will be suitable only to MP3 coders, and not for its successors, since (as mentioned in the end of appendix A) more advanced standards abandoned the use of hybrid filter-bank and use pure MDCT instead.

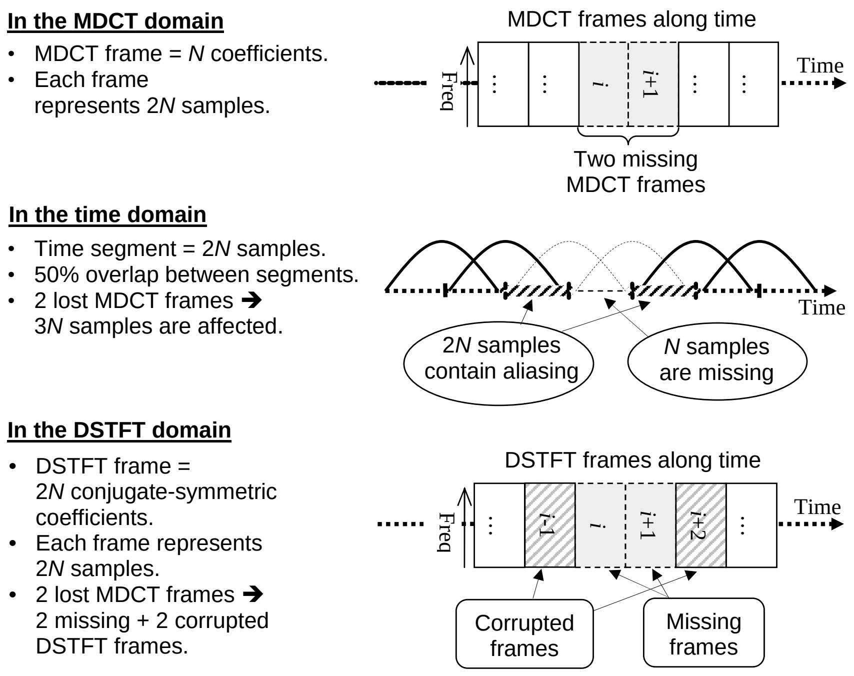

In the MDCT domain
• MDCT frame = N coefficients.
• Each frame represents 2N samples.
MDCT frames along time
Freq
Time
i
i+1
Two missing MDCT frames
In the time domain
• Time segment = 2N samples.
• 50% overlap between segments.
• 2 lost MDCT frames ➔ 3N samples are affected.
Time
2N samples contain aliasing
N samples are missing
In the DSTFT domain
• DSTFT frame = 2N conjugate-symmetric coefficients.
• Each frame represents 2N samples.
• 2 lost MDCT frames ➔ 2 missing + 2 corrupted DSTFT frames.
DSTFT frames along time
Freq
Time
i-1
i
i+1
i+2
Corrupted frames
Missing frames

Figure 4.2: Propagation of error: 2 lost MDCT frames affect the reconstruction of 2+2=4 DSTFT frames

To conclude, here is a summarized list of the benefits and limitations of each concealment domain:

Time Domain

Benefits:

- Interpolation is applied directly to the waveform, regardless of coder type.

Limitations:

- The gap that has to be interpolated is very wide and is hence very difficult to handle: Q consecutive lost packets result in $(2Q+1)\cdot 576$ consecutive missing samples.
- Audio signals can't be described by simple models that might have been used for the interpolation (as opposed to speech signals).

Sub-band Domain

Benefits:

- Time-domain interpolation techniques can be used.
- Since each signal represents a single sub-band, the signal is less complicated and therefore can be assumed to approximately comply with simpler models, such as an AR model.

Limitations:

- Still relatively wide gap: Q consecutive lost packets result in $(2Q + 1) \cdot 18$ consecutive missing samples in each of the 32 sub-bands.
- Interpolation should be performed for each of the sub-bands (32 times).
- Since more advanced MPEG-audio coders don't use hybrid filter-banks anymore, working in this domain is limited only for the MP3 standard.

MDCT Domain

Benefits:

- A small gap which is easier to interpolate: Q consecutive lost packets result in $2Q$ consecutive missing coefficients at each frequency bin.

Limitations:

- Interpolation should be performed for each frequency bin, i.e., 576 times (although higher frequency bins, which contain almost no energy could be ignored).
- The coefficients in this domain has rapid sign changes, which makes it harder to interpolate them.

- Different resolutions for different frames.

DSTFT Domain

Benefits:

- The gap is slightly widened relatively to the MDCT domain, but still small: Q consecutive lost packets result in $2Q + 2$ consecutive missing coefficients at each frequency bin.
- The representation of the signal is smoother than in the other mentioned domains.
- All the frames has the same spectral resolution (as opposed to the MDCT domain).

Limitations:

- As in the MDCT domain, the interpolation process is performed separately for each frequency bin.
- The conversion process adds complexity to the decoder.

4.2 Analysis of Pure Sinusoids in Different Concealment Domains

As was mentioned in the previous section, one of the reasons that the DSTFT domain was chosen as our concealment domain is that its signal representation is less fluctuating. For verification, the algorithm was implemented also in the MDCT domain and indeed showed less satisfying results than the DSTFT implementation (see Chapter 7). Therefore, this section compares the representations of a pure sinusoid, which is a basic signal, in both domains: MDCT vs. DSTFT. Also, since this work uses a reconstruction algorithm that uses spectral estimation, then as a preface to Chapter 5, we examine and compare the spectrum of the coefficients in each domain, as they vary along time.

The development of all the mathematical expressions that are presented here can be found in Appendix C.

4.2.1 Representation of Pure Sinusoids in Different Domains

We define a sinusoidal signal:

$$x[n] = A \cdot \cos(\omega_0 n + \phi_0) \quad , n \in \mathbb{Z} \tag{4.1}$$

The frequency is defined as:

$$\omega_0 \triangleq \frac{2\pi}{2N}(K_0 + \Delta_0) \tag{4.2}$$

Where K_0 is an integer number: $0 \leq K_0 < 2N$ and Δ_0 is a fractional number: $0 \leq \Delta_0 < 1$. Hence, $x[n]$ can be written as:

$$x[n] = A \cdot \cos\left(\tfrac{2\pi}{2N}\left(K_0 + \Delta_0\right) n + \phi_0\right) \quad , n \in \mathbb{Z} \tag{4.3}$$

Without loss of generality we assume that the overlapping $2N$-sample segments to which the MDCT and DSTFT transforms are applied to, are located on the time-axis at integer multiples of N. Therefore, the p'th segment, where $p \in \mathbb{Z}$, will be denoted as:

$$x_{(p)}[n] \triangleq \{x[pN + n]\}_{n=0}^{2N-1} \qquad \forall p \in \mathbb{Z} \tag{4.4}$$

Applying an MDCT transform

For simplicity, let's assume that a single type of window (Long) is used for the

MDCT of all the segments. Using the MDCT's definition in (3.4) and the definition of the Long window in (3.7) we have:

$$\begin{aligned} X_{(p)}^{MDCT}[k] &= \sum_{n=0}^{2N-1} x[pN+n]\cdot h[n]\cdot \cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \quad ,0\le k\le N-1 \\ &= A\sum_{n=0}^{2N-1}\left\{\begin{array}{l} \cos\left(\frac{2\pi}{2N}(K_0+\Delta_0)(pN+n)+\phi_0\right) \\ \cdot\sin\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\cdot\cos\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\end{array}\right\} \end{aligned} \tag{4.5}$$

Applying a DSTFT transform

The window which is used for the DSTFT is an even-length Hann window:

$$w[n] = \sin\left(\tfrac{\pi}{2N}\cdot\left(n+\tfrac{1}{2}\right)\right)^2,\ \ 0\le n\le 2N-1 \tag{4.6}$$

Applying a DSTFT transform to the signal we have:

$$\begin{aligned} X_{(p)}^{DSTFT}[k] &= \sum_{n=0}^{2N-1} x[pN+n]\cdot w[n]\cdot e^{j\frac{2\pi}{2N}nm} \quad ,0\le k\le 2N-1 \qquad (4.7) \\ &= A\sum_{n=0}^{2N-1}\cos\left(\tfrac{2\pi}{2N}(K_0+\Delta_0)(pN+n)+\phi_0\right)\cdot\sin\left(\tfrac{\pi}{2N}\left(n+\tfrac{1}{2}\right)\right)^2\cdot e^{j\frac{2\pi}{2N}nm} \end{aligned}$$

Next, we explore the behavior of the coefficients along time for each transform and different values of Δ_0:

I. $\Delta_0 = 0$

In this case the frequency is located exactly in the middle of one of the frequency bins defined by the transforms.

In this case the MDCT coefficients get the following form:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} = \frac{NA}{2}(-1)^{p\cdot K_0}\cdot \begin{bmatrix} \sin\left(\frac{\pi(N+1)}{2N}k+\frac{\pi(N+2)}{4N}+\phi_0\right)\cdot\delta[K_0+k+1] \\ +\sin\left(-\frac{\pi(N+1)}{2N}k-\frac{\pi}{4}-\phi_0\right)\cdot\delta[K_0+k] \\ +\sin\left(\frac{\pi(N+1)}{2N}k+\frac{\pi(N+2)}{4N}-\phi_0\right)\cdot\delta[K_0-k-1] \\ +\sin\left(-\frac{\pi(N+1)}{2N}k-\frac{\pi}{4}+\phi_0\right)\cdot\delta[K_0-k] \end{bmatrix} \tag{4.8}$$

The DSTFT coefficients get the following form:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = \frac{NA}{2}(-1)^{p\cdot K_0}\cdot \begin{bmatrix} e^{j\phi_0}\cdot\delta[K_0-k]+e^{-j\phi_0}\cdot\delta[K_0+k] \\ -\frac{1}{2}e^{j\phi_0+j\frac{\pi}{2N}}\cdot\delta[K_0-k+1]-\frac{1}{2}e^{-j\phi_0+j\frac{\pi}{2N}}\cdot\delta[K_0+k-1] \\ -\frac{1}{2}e^{j\phi_0-j\frac{\pi}{2N}}\cdot\delta[K_0-k-1]-\frac{1}{2}e^{-j\phi_0-j\frac{\pi}{2N}}\cdot\delta[K_0+k+1] \end{bmatrix} \tag{4.9}$$

As can be seen in both cases, for a fixed frequency bin, indexed k, the change in time is reflected by the factor of $(-1)^{p\cdot K_0}$. Hence, in this particular case there is no reason to prefer the DSTFT domain over the MDCT domain.

II. $\Delta_0 \neq 0$

In this case the frequency is not located exactly in the middle of a frequency bin, and therefore we tend to get fluctuations in the envelope of the resulting coefficients along time.

In this case the MDCT coefficients have the general form:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = A_1(k)\sin(\alpha p+\beta_1(k)) + A_2(k)\sin(\alpha p+\beta_2(k)) \tag{4.10}$$

And the DSTFT coefficients have the general form:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0\neq 0} = B_1(k)\,e^{-j(\alpha p+\gamma_2(k))} + B_2(k)\,e^{j(\alpha p+\gamma_1(k))} \tag{4.11}$$

Where:

$$\begin{aligned}
\alpha &\triangleq \pi(K_0+\Delta_0)\\
A_1(k) &\triangleq \tfrac{A}{4}\sin(\pi\Delta_0)\cdot\left[\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0+k)\right)^{-1}+\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0+k+1)\right)^{-1}\right]\\
A_2(k) &\triangleq \tfrac{A}{4}\sin(\pi\Delta_0)\cdot\left[\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0-k-1)\right)^{-1}+\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0-k)\right)^{-1}\right]\\
B_1(k) &\triangleq \tfrac{A}{4}\sin(\pi\Delta_0)\cdot\left[\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0+k)\right)^{-1}-\tfrac{1}{2}\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0+k+1)\right)^{-1}\right.\\
&\quad\left.-\tfrac{1}{2}\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0+k-1)\right)^{-1}\right]\\
B_2(k) &\triangleq \tfrac{A}{4}\sin(\pi\Delta_0)\cdot\left[\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0-k)\right)^{-1}-\tfrac{1}{2}\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0-k+1)\right)^{-1}\right.\\
&\quad\left.-\tfrac{1}{2}\sin\left(\tfrac{\pi}{2N}(K_0+\Delta_0-k-1)\right)^{-1}\right]
\end{aligned} \tag{4.12}$$

The expressions for $\beta_1(k)$, $\beta_2(k)$, $\gamma_1(k)$ and $\gamma_2(k)$ can be found in Appendix C.

By observing expressions (4.10)-(4.12), we can conclude that the signal's representation in the DSTFT domain is usually less fluctuating than its MDCT representation: Note that all the gain functions (A_i, $B_i, i=1,2$) have the same structure, which defines that when $k\rightarrow K_0$ the values of the gain functions $A_1(k)$ and $B_1(k)$ becomes very high. Same goes for $A_2(k)$ and $B_2(k)$, when $k\rightarrow(2N-K_0)$. When k is far from these values, the values of the corresponding functions decreases quickly towards zero. Hence, in most cases, at a fixed value of k, one of the gain functions (i.e., $i = 1, 2$) will be much more dominant than the other. In these cases the expression can be approximated by only one signal component (a single sine function in case of $X_{(p)}^{MDCT}[k]$, and a single exponent in case of $X_{(p)}^{DSTFT}[k]$) and therefore the gain value will be almost constant along time. This fact is better reflected in the DSTFT domain, where the absolute value of the coefficients along time is almost

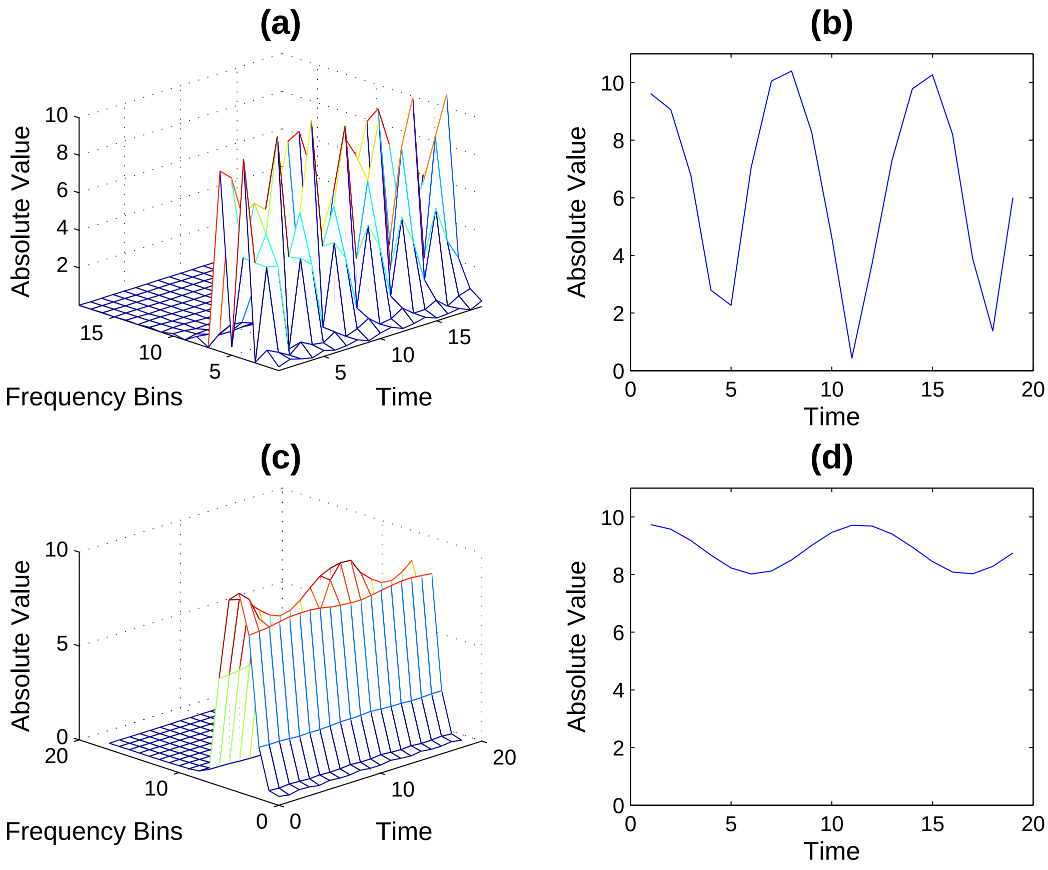


Figure 4.3: An example of MDCT coefficients vs. DSTFT coefficients along time, using a signal that contains 2 sinusoids, at: for $K_0 = 5$, $\Delta_0 = 0.15$ and $\phi_0 = 0.7$, $K_1 = 3$, $\Delta_1 = 0.34$ and $\phi_1 = 0$. (a) The MDCT coefficients, (b) The MDCT amplitude at frequency bin $k = K_0$, (c) The DSTFT coefficients, (d) The DSTFT amplitude at frequency bin $k = K_0$

constant, as opposed to the coefficients in the MDCT domain (where the absolute value of the coefficients along time tends to fluctuate because the gain functions are multiplied by a sine function). Figure 4.3 shows an example of that.
This applies, of course, to most k and K_0 values. However, in the cases where k is positioned at the same distance from K_0 and $(2N - K_0)$, both gain functions have the same effect on the coefficient's value. In this special case, the absolute value of the coefficients along time in both domains have significant fluctuations along time.

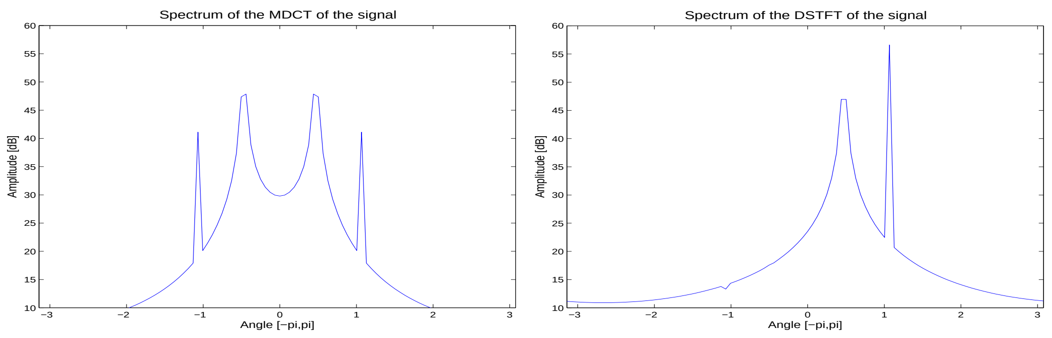
Finally, regarding the parameter ϕ_0 (initial phase of the input sinusoid), all the expressions that were presented so far show clearly that the effect of ϕ_0 on the behavior of the envelope of the coefficients in both domains, along time, is very small.

4.2.2 The Spectrum of the Coefficients in Each Domain

Since the reconstruction algorithm that was chosen in this work (the algorithm is introduced in Chapter 5) uses spectral estimation, we explore the spectral behavior of the coefficients in the MDCT and DSTFT domains, as they vary in time. By that we mean that the coefficients at a fixed frequency bin, k, along the time axis (represented by p), are considered as a time-signal, and we explore the **spectrum of this signal**.

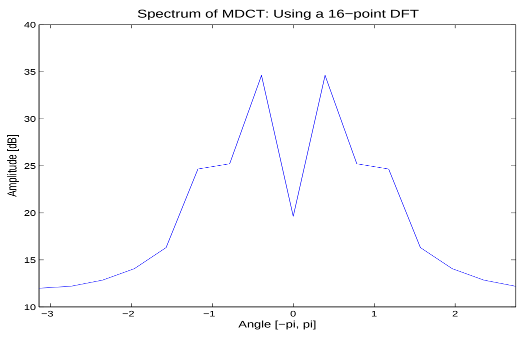
Looking at the expressions in (4.10)-(4.11) it is clear that the spectrum of such signals contains only delta functions. In the case of the MDCT coefficients, each sinusoid is mapped into two delta functions in the spectral domain. In the case of DSTFT coefficients, each exponent is mapped into a single delta function in the spectral domain. The difference becomes more obvious when the original waveform contains more than one sinusoid: a superposition of several sinusoidal signals leads to a complicated spectral image, especially if the frequencies of these sinusoids are close to each other, as is the case presented in Figures 4.4 and 4.5: Figure 4.4 shows the two spectral images, when estimated from a large data sequence. Figure 4.5 shows a more realistic situation, where the spectral images are estimated based on a very short segment of data. Clearly, the images are less clear in this case, however the MDCT spectral image that contains more delta functions looks more like a wide-band spectrum than the DSTFT's spectral image.

The combination of K_0 and Δ_0 determines the location of the delta functions in the spectrum.

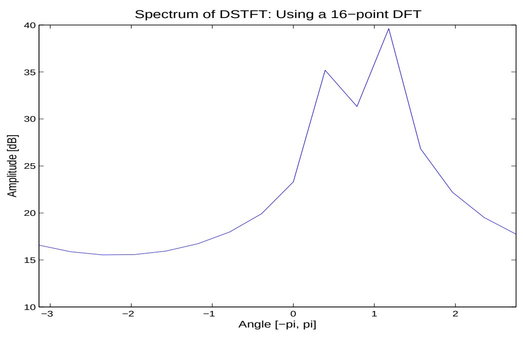


(a) Spectrum of the MDCT coefficients (b) Spectrum of the DSTFT coefficients

Figure 4.4: The spectrum of the coefficients in the MDCT vs. DSTFT domains, at bin $k = 4$: (a) MDCT, (b) DSTFT. The same signal is used as in Figure 4.3.



(a) Spectrum of the MDCT coefficients



(b) Spectrum of the DSTFT coefficients

Figure 4.5: Same as in Figure 4.4, this time using only a 16-point DFT.

Chapter 5

Reconstruction Algorithms for Complex Signals

This chapter introduces the GAPES [8, 31] and MAPES-CM [9] reconstruction algorithms, that are used in this work. The GAPES algorithm was chosen because of several of its qualities: As opposed to the Statistical Interpolation algorithm, suggested in [6], the GAPES algorithm assumes no parametric modelling 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. The MAPES-CM algorithm is a more advanced version published recently, which has all of GAPES's advantages but can handle more loss patterns than GAPES can and has less complexity. Older methods for estimating the spectrum of an incomplete data sequence include [32]: direct spectral calculation, substituting zeroes instead of the lost samples, averaging or using models (such as the AR model).

Since both reconstruction algorithms are derived from the APES [10, 33] algorithm (Amplitude and Phase EStimation), it will be described first. Then, the GAPES algorithm is described and after it the MAPES-CM algorithm. Finally, we compare the two reconstruction algorithms. Note that only the Forward version of GAPES and APES is introduced here, although there exist also a Forward-Backward version of the algorithms. 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 simulations, to justify its complexity.

5.1 The APES Algorithm

This section introduces the APES algorithm for spectral estimation of complete data sequences. We mention the adaptive filter-bank interpretation of this algorithm, which later leads to the derivation of GAPES, and the interpretation as an ML estimator, which later leads to the derivation of MAPES-CM.

Let $\{x_n\}_{n=0}^{P-1} \in \mathbb{C}^P$ denote a discrete-time data sequence of length P. We wish to estimate the spectral component at frequency ω_0, denoted $\alpha(\omega_0)$, of this data sequence. For the frequency of interest, ω_0, we model x_n as:

$$x_n = \alpha(\omega_0) \cdot e^{j\omega_0 n} + e_n(\omega \neq \omega_0) \tag{5.1}$$

where $e_n(\omega \neq \omega_0)$ denotes the residual term which includes the un-modelled noise and the interference from frequencies other than ω_0.

5.1.1 Adaptive Filter-Bank Interpretation

In order to estimate $\alpha(\omega_0)$, a *data-dependent* narrow-band filter, $\underline{h}(\omega_0)$, is designed by requiring that its output will be as close as possible, in the LS sense, to a sinusoid with frequency ω_0. I.e., the filter $\underline{h}(\omega_0)$ should pass the frequency ω_0 without distortion, and at the same time attenuate all the other frequencies as much as possible. The estimated spectral coefficient, $\hat{\alpha}(\omega_0)$, is hence obtained by first filtering the data sequence, $\{x_n\}_{n=0}^{P-1}$ and then calculating the DFT coefficient of the filtered data at frequency ω_0.

Let $\underline{h}(\omega_0)$ denote the impulse response of an M-tap **narrow-band** FIR filter centered at frequency ω_0 ($[\cdot]^T$ denotes the transpose):

$$\underline{h}(\omega_0) \triangleq [h_1(\omega_0), h_2(\omega_0), \ldots, h_M(\omega_0)]^T \tag{5.2}$$

Let $\{\underline{y}_l\}_{l=0}^{L-1}$ denote the $L = P - M + 1$ overlapping sub-vectors (data snapshots) of the full data sequence $\{x_n\}_{n=0}^{P-1}$:

$$\begin{aligned} \underline{y}_l &\triangleq [x_l, x_{l+1}, \ldots, x_{l+M-1}]^T \quad , l = 0, 1, \ldots, L-1 \\ &= \alpha(\omega_0) \cdot \underline{a}(\omega_0) \cdot e^{j\omega_0 l} + \underline{e}_l(\omega_0) \end{aligned} \tag{5.3}$$

where $\underline{a}(\omega_0) \triangleq [1, e^{j\omega_0}, \ldots, e^{j\omega_0(M-1)}]^T$ and $\underline{e}_l(\omega_0) \triangleq \underline{e}_l(\omega \neq \omega_0)$ are the data snapshots of the residual terms: $\underline{e}_l(\omega_0) \triangleq [e_l(\omega_0), e_{l+1}(\omega_0), \ldots, e_{l+M-1}(\omega_0)]^T$. Both vectors are of size $M \times 1$.

Using (5.3) the act of passing the samples of $\{x_n\}_{n=0}^{P-1}$ through the $\underline{h}(\omega_0)$ filter, can be written as:

$$\underline{h}^H(\omega_0) \cdot \underline{y}_l = \alpha(\omega_0) \cdot [\underline{h}^H(\omega_0) \cdot \underline{a}(\omega_0)] \cdot e^{j\omega_0 l} + \underline{e}_l(\omega_0) \quad , l = 0, 1, \ldots, L-1 \tag{5.4}$$

where $[\cdot]^H$ denotes the conjugate-transpose.

Since $\underline{h}(\omega_0)$ is a narrow-band filter centered at ω_0, (5.4) becomes:

$$\underline{h}^H(\omega) \cdot \underline{y}_l \simeq \alpha(\omega) \cdot e^{j\omega l} \quad , l = 0, 1, \ldots, L-1 \tag{5.5}$$

Now, the desired spectral component, $\alpha(\omega_0)$, can be estimated using DFT on the filtered data.

In conclusion to all mentioned above, the process of estimating the spectral coefficient, $\alpha(\omega)$, of any given frequency, ω, can be defined by the following minimization problem:

$$\min_{\underline{h}(\omega),\alpha(\omega)} \quad \frac{1}{L}\sum_{l=0}^{L-1}|\underline{h}^H(\omega)\cdot\underline{y}_l-\alpha(\omega)\cdot e^{j\omega l}|^2 \quad \text{, subject to } \underline{h}^H(\omega)\cdot\underline{a}(\omega)=1 \tag{5.6}$$

where the constraint is added in order to prevent the problem from converging to the trivial solution (where everything is zeroed), while maintaining the demand for passing frequency ω undistorted.

Minimizing the criterion function with respect to $\alpha(\omega)$ gives:

$$\hat{\alpha}(\omega)=\underline{h}^H(\omega)\cdot\underline{Y}(\omega) \tag{5.7}$$

where:

$$\underline{Y}(\omega)\triangleq\frac{1}{L}\sum_{l=0}^{L-1}\underline{y}_l\cdot e^{-j\omega l} \tag{5.8}$$

Substituting (5.7) in (5.6) yields the following minimization problem:

$$\min_{\underline{h}(\omega)} \quad \underline{h}^H(\omega)\hat{\mathbf{S}}(\omega)\underline{h}(\omega) \quad \text{, subject to } \underline{h}^H(\omega)\cdot\underline{a}(\omega)=1 \tag{5.9}$$

where:

$$\hat{\mathbf{S}}(\omega)\triangleq\hat{\mathbf{R}}-\underline{Y}(\omega)\cdot\underline{Y}^H(\omega) \tag{5.10}$$

and $\hat{\mathbf{R}}$ is the sample-covariance matrix:

$$\hat{\mathbf{R}}\triangleq\frac{1}{L}\sum_{l=0}^{L-1}\underline{y}_l\cdot\underline{y}_l^H \tag{5.11}$$

This problem is also known as the problem of finding a *minimum-variance distortionless response beamformer* [34]. The solution to this minimization problem is achieved by using Lagrange multipliers [35], and is given by:

$$\underline{h}(\omega) = \frac{\hat{\mathbf{S}}^{-1}(\omega) \cdot \underline{a}(\omega)}{\underline{a}^H(\omega) \cdot \hat{\mathbf{S}}^{-1}(\omega) \cdot \underline{a}(\omega)} \tag{5.12}$$

5.1.2 Maximum-Likelihood Estimator Interpretation

The APES algorithm can also be explained from another point of view, as an algorithm that mimics a maximum-likelihood (ML) estimator.

Using the same value of M, we return to (5.3) where we assume that $\{\underline{e}_l(\omega)\}_{l=0}^{L-1}$ are zero-mean circularly symmetric complex Gaussian random vectors that are statistically independent of each other and that have the same unknown covariance matrix:

$$E[\underline{e}_i(\omega) \cdot \underline{e}_j^H(\omega)] = \mathbf{Q}(\omega) \cdot \delta[i,j] \tag{5.13}$$

where $E[\cdot]$ denotes the expectation and $\delta[i,j]$ is the kronecker delta. Note that since these vectors contain overlapping data, they are obviously not statistically independent of each other, hence APES is not an **exact** ML estimator, but only an approximate one.

Under these assumptions, and also using (5.3), we can determine the ML estimator of the $\{\underline{y}_l\}_{l=0}^{L-1}$ vectors as:

$$\max_{\mathbf{Q}(\omega),\alpha(\omega)} \quad \log\left(Pr\left\{\{\underline{y}_l\}_{l=0}^{L-1} | \alpha(\omega), \mathbf{Q}(\omega)\right\}\right) \tag{5.14}$$

where the expression of the conditional probability is of L uncorrelated random gaussian vectors of length M. By using the normalized log-likelihood function of these vectors, the following ML criterion can be written:

$$\max_{\mathbf{Q}(\omega),\alpha(\omega)} \quad -\ln|\mathbf{Q}(\omega)| - \frac{1}{L}\sum_{l=0}^{L-1}\left[\underline{y}_l - \alpha(\omega)\cdot\underline{a}(\omega)\cdot e^{j\omega l}\right]^H \cdot \mathbf{Q}^{-1}(\omega)\left[\underline{y}_l - \alpha(\omega)\cdot\underline{a}(\omega)\cdot e^{j\omega l}\right] \tag{5.15}$$

Maximizing this criterion with respect to $\mathbf{Q}(\omega)$ yields that the ML estimate of $\alpha(\omega)$ can be obtained by minimizing the following cost function:

$$F = \left|\hat{\mathbf{Q}}_{ML}(\omega)\right| = \left|\frac{1}{L}\sum_{l=0}^{L-1}[\underline{y}_l - \alpha(\omega)\cdot\underline{a}(\omega)\cdot e^{j\omega l}]\cdot[\underline{y}_l - \alpha(\omega)\cdot\underline{a}(\omega)\cdot e^{j\omega l}]^H\right| \tag{5.16}$$

Minimizing this cost function with respect to $\alpha(\omega)$ yields:

$$\hat{\alpha}(\omega) = \frac{\underline{a}^H(\omega)\cdot\hat{\mathbf{S}}^{-1}(\omega)\cdot\underline{Y}(\omega)}{\underline{a}^H(\omega)\cdot\hat{\mathbf{S}}^{-1}(\omega)\cdot\underline{a}(\omega)} \quad , \tag{5.17}$$

which is same as the solution given by (5.7), where $\underline{a}(\omega)$, $\underline{Y}(\omega)$ and $\hat{\mathbf{S}}(\omega)$ are all defined in the previous sub-section.

5.2 The GAPES Reconstruction Algorithm

This section contains a description of the GAPES algorithm according to [8, 31], and explains its connection to APES. Then, we give an example to GAPES performance in the MDCT and DSTFT domain, using the numerical example given in Chapter 4.

5.2.1 Description of GAPES

Let $\{x_n\}_{n=0}^{P-1} \in \mathbb{C}^P$ denote again a discrete-time data sequence of length P, only this time some of the samples are missing. Let $\underline{x}_m$ denote the vector of P_m missing samples and $\underline{x}_a$ the vector of $P_a = P - P_m$ available samples.

By adapting the spectral estimation problem using the filter-bank approach described in section 5.1.1, to the case of missing data, we reconstruct the missing data assuming that it has the same spectral content as the available data surrounding it. The problem can be described by the following minimization problem:

$$\min_{\underline{x}_m,\{\underline{h}(\omega_k),\alpha(\omega_k)\}_{k=0}^{K-1}} \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} |\underline{h}^H(\omega_k)\cdot \underline{y}(l) - \alpha(\omega_k)\cdot e^{j\omega_k l}|^2 \tag{5.18}$$

$$\text{,subject to } \underline{h}^H(\omega_k)\cdot \underline{a}(\omega_k) = 1$$

where $\{\omega_k\}_{k=0}^{K-1}$ is a pre-defined frequency grid.

The GAPES algorithm suggests an iterative solution to the problem above, where each iteration contains two main steps:

1. Estimate the spectral parameters $\{\underline{h}(\omega), \alpha(\omega)\}$ for a pre-defined frequency grid $\{\omega_k\}_{k=0}^{K-1}$, based both on the available samples and the on estimated samples from the previous iteration, using the APES algorithm (the first initiation of the lost samples will be described in the sequel). At the end of this step we have the set $\{\underline{h}(\omega_k), \alpha(\omega_k)\}_{k=0}^{K-1}$.

2. Reconstruct the missing samples so that their spectral content will approximate the estimated spectrum, described by $\{\underline{h}(\omega_k), \alpha(\omega_k)\}_{k=0}^{K-1}$, in the LS sense. $\underline{x}_m$ is determined by requiring that the output signal, resulting from

filtering the complete data sequence (built from both available and estimated samples) with each of the filters $\underline{h}(\omega_k)$, will be as close as possible (in the least-squares sense) to a sine with a complex amplitude equal to $\alpha(\omega_k)$.

Since the first step (using APES) is already described in the previous section, we continue with the description of the second step:
The expression in (5.18) can be written using matrices in the following way:

$$\min_{\underline{x}_m, \{\underline{h}(\omega_k), \alpha(\omega_k)\}_{k=0}^{K-1}} \sum_{k=0}^{K-1} \left|\mathbf{H}_k \cdot \underline{x} - \alpha(\omega_k) \cdot \underline{b}_k\right|^2 \quad \text{, subject to } \underline{h}^H(\omega_k) \cdot \underline{a}(\omega_k) = 1 \tag{5.19}$$

where $\underline{x} \in \mathbb{C}^{P \times 1}$ is the vector containing all the samples (available and missing), $\underline{b}_k \triangleq [1, e^{j\omega_k}, \ldots, e^{j(L-1)\omega_k}]^T \in \mathbb{C}^{L \times 1}$, and:

$$\mathbf{H}_k \triangleq \begin{bmatrix} \underline{h}^H(\omega_k) & & & & \\ & \underline{h}^H(\omega_k) & & & \\ & & \underline{h}^H(\omega_k) & & \\ & & & \ddots & \\ & & & & \underline{h}^H(\omega_k) \end{bmatrix} \in \mathbb{C}^{L \times P} \tag{5.20}$$

Let's define the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ from $\mathbf{H}_k$ via the following equation:

$$\mathbf{H}_k \cdot \underline{x} = \mathbf{A}_k \cdot \underline{x}_a + \mathbf{B}_k \cdot \underline{x}_m \tag{5.21}$$

where $\mathbf{A}_k \in \mathbb{C}^{L \times P_a}$ and $\mathbf{B}_k \in \mathbb{C}^{L \times P_m}$.
Using these notations, (5.19) becomes:

$$\min_{\underline{x}_m, \{\underline{h}(\omega_k), \alpha(\omega_k)\}} \sum_{k=0}^{K-1} \left|\mathbf{B}_k \cdot \underline{x}_u - \left(\alpha(\omega_k) \cdot \underline{b}_k - \mathbf{A}_k \cdot \underline{x}_a\right)\right|^2 \quad \text{,subject to } \underline{h}^H(\omega_k) \cdot \underline{a}(\omega_k) = 1 \tag{5.22}$$

Minimizing it with respect to $\underline{x}_m$ gives:

$$\hat{\underline{x}}_m = \left(\sum_{k=0}^{K-1} \mathbf{B}_k^H \cdot \mathbf{B}_k \right)^{-1} \cdot \left(\sum_{k=0}^{K-1} \left(\mathbf{B}_k^H \alpha\left(\omega_k\right) \underline{b}_k - \mathbf{B}_k^H \mathbf{A}_k \underline{x}_a \right) \right) \tag{5.23}$$

Finally, regarding the first iteration:
Two options are suggested in [8], depending on the number and pattern of the missing samples in the data sequence. The first option is to assume that all the missing samples have zero values. The second option, which according to our simulations performs better, suggests 'ignoring', during the first iteration only, the $\underline{y}_l$ vectors that contain one or more missing samples. These vectors are 'ignored' by simply replacing them with zero vectors and by dividing the sums that appear in (5.12) and in (5.11) by L' instead of L, where $L' \leq L$ is the number of $\underline{y}_l$ containing only available samples (i.e., the ones that weren't ignored). Note that this initialization dictates that for some loss patterns, all (or almost all) of the $\underline{y}_l$ vectors will be replaced by zero vectors, which makes the estimation useless and unstable.
This is the reason why we say that GAPES is more limited than MAPES-CM when it comes to dealing with various loss patterns.

5.2.2 Comparing GAPES's Performance in Different Domains

On simple signals, such as single sinusoids, using GAPES in both the MDCT and the DSTFT domains show good results. However, there are some cases, where the advantages of the DSTFT domain, as were described in section 4.2, make a big difference.

Figure 5.1-5.2 shows such a case: we use the same numerical example presented in section 4.2. The signal was divided to 20 frames, that represent 50% overlapping segments of 36 samples each. The frames were transformed to the MDCT domain and to the DSTFT domain. For each domain, 6 frames were considered missing and then reconstructed by applying GAPES separately on each frequency bin. Then, the waveform was restored by transforming each frame back to the time domain and using overlap and add.

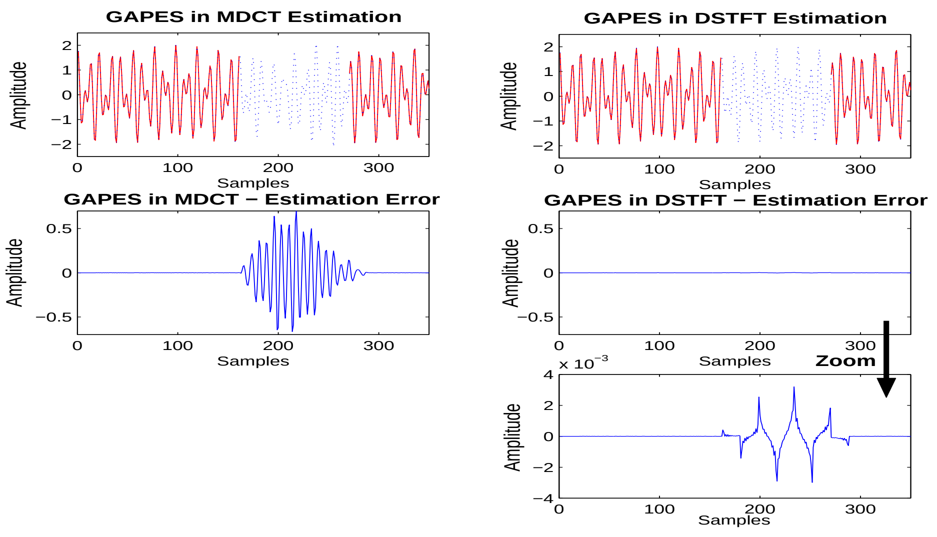


Figure 5.1: GAPES in MDCT vs. GAPES in DSTFT: Example 1: Frames 10-15 are missing. The estimated signal is marked by a dotted line.

Figure 5.1 presents the estimation results when frames 10 to 15 are missing. Figure 5.2 shows the case were frames 3-6 and 15-16 are missing. For reference, applying the same two examples on a simple signal containing only one of the sinusoids ($K_0 = 5$, $\Delta_0 = 0.15$ and $\phi_0 = 0.7$) gives good results in both domains. This case is presented in figure 5.3.

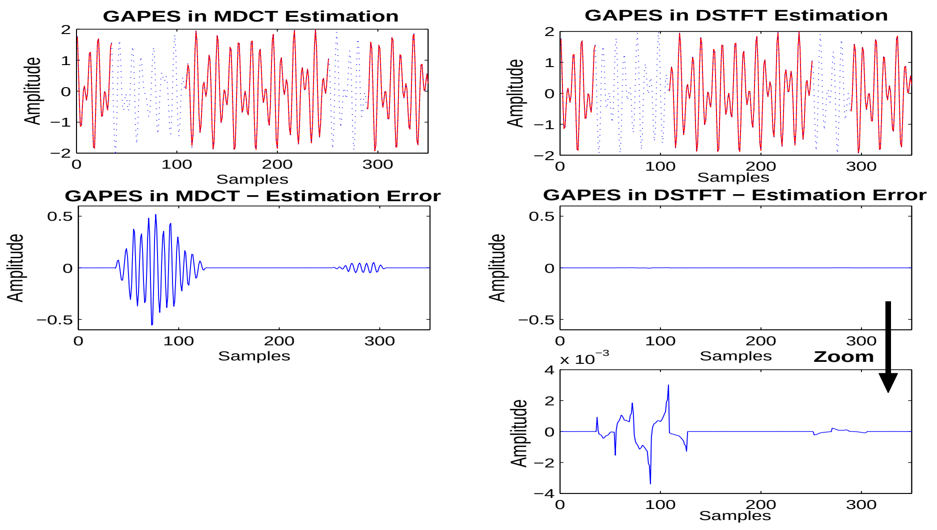


Figure 5.2: GAPES in MDCT vs. GAPES in DSTFT: Example 2: Frames 3-6 and 15-16 are missing. The estimated signal is marked by a dotted line.

5.3 The MAPES-CM Reconstruction Algorithm

Three versions of the MAPES algorithm are described in the literature: MAPES-EM1,2 (Expectation Maximization) [36] and MAPES-CM (Cyclic Maximization) [9]. As in the GAPES algorithm, these three algorithms are designed for spectral estimation of a sequence with missing data, only that these algorithms are derived from the APES algorithm using the maximum-likelihood (ML) interpretation, as described in sub-section 5.1.2. However, among these three MAPES algorithms, only MAPES-CM reconstructs the missing samples as part of the spectral estimation process (as GAPES does) and hence is more suitable for our needs.

The spectral estimation problem is adapted again to the case of a discrete-time sequence with missing samples, as was described in the previous section, only this time using the ML estimator approach. Using this direction, we obtain an extended

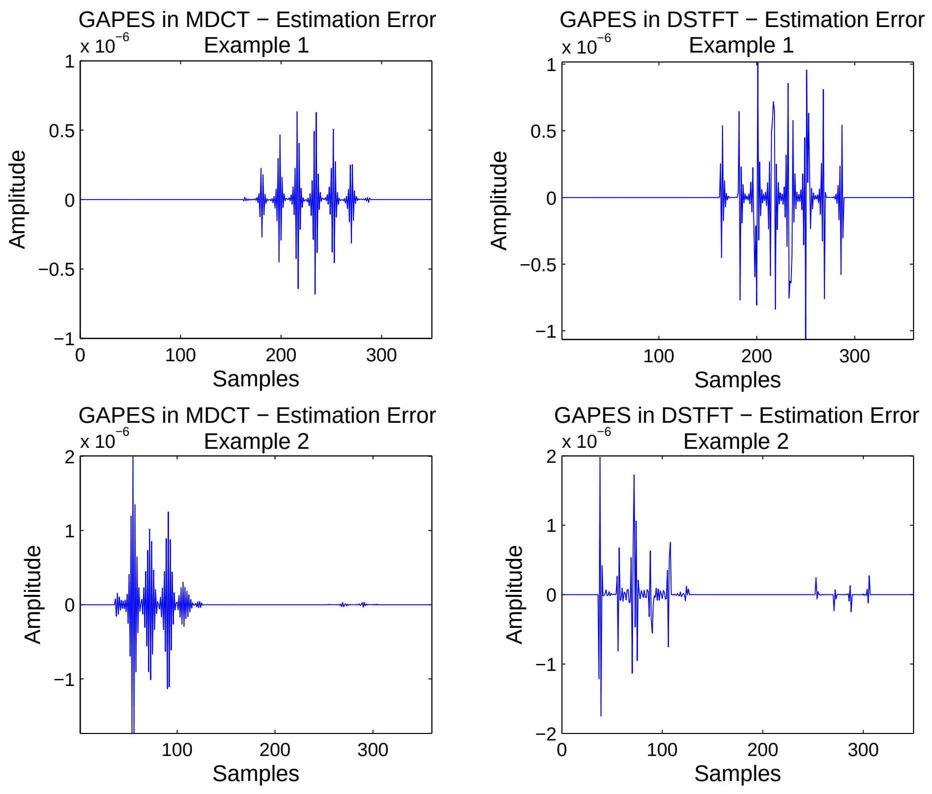


Figure 5.3: GAPES in MDCT vs. GAPES in DSTFT, this time with a signal containing a single sinusoid.

version of the ML criterion that was presented in (5.15):

$$\max_{\underline{x}_m, \{\mathbf{Q}(\omega_k),\alpha(\omega_k)\}_{k=0}^{K-1}} \sum_{k=0}^{K-1} \left\{ \begin{array}{l} -\ln|\mathbf{Q}(\omega_k)| \\ \\ -\frac{1}{L}\sum_{l=0}^{L-1}\left[\underline{y}_l - \alpha(\omega_k)\underline{a}(\omega_k)e^{j\omega_k l}\right]^H \mathbf{Q}^{-1}(\omega_k)\left[\underline{y}_l - \alpha(\omega_k)\underline{a}(\omega_k)e^{j\omega_k l}\right] \end{array} \right\} \tag{5.24}$$

Similar to GAPES, the MAPES-CM algorithm suggests an iterative solution to the problem above, where each iteration contains two main steps:

1. Estimate the spectral parameters $\{\mathbf{Q}(\omega), \alpha(\omega)\}$ for a pre-defined frequency grid $\{\omega_k\}_{k=0}^{K-1}$, based both on the available samples and the on estimated samples from the previous iteration, using the APES algorithm (the values of the missing samples are set to zero before the first iteration). At the end of this step we have the set $\{\mathbf{Q}(\omega_k), \alpha(\omega_k)\}_{k=0}^{K-1}$.

2. Reconstruct the missing samples so that they will comply with the maximum likelihood criterion, assuming that the set of spectral parameters, $\{\mathbf{Q}(\omega), \alpha(\omega)\}_{k=0}^{K-1}$, is available.

Again, since the first step is already described in section 5.1, only the second step is described here:

The expression in (5.24) can be written using matrices in the following way:

$$\max_{\substack{\underline{x}_m, \\ \{\mathbf{Q}(\omega_k),\alpha(\omega_k)\}_{k=0}^{K-1}}} \sum_{k=0}^{K-1} \left\{ \begin{array}{l} -\ln|\mathbf{D}_k| \\ \\ -\frac{1}{L}\left[\underline{\tilde{y}} - \alpha(\omega_k)\underline{\rho}(\omega_k)\right]^H \mathbf{D}_k^{-1}\left[\underline{\tilde{y}} - \alpha(\omega_k)\underline{\rho}(\omega_k)\right] \end{array} \right\} \tag{5.25}$$

where $\underline{\tilde{y}}$ is an $LM \times 1$ column-stack vector obtained by concatenating all the data snapshots, $\{\underline{y}_l\}_{l=0}^{L-1}$:

$$\underline{\tilde{y}} \triangleq \begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \\ \vdots \\ \underline{y}_{L-1} \end{bmatrix} \in \mathbb{C}^{LM\times 1} \tag{5.26}$$

and:

$$\mathbf{D}_k \triangleq \begin{bmatrix} \mathbf{Q}(\omega_k) & & & & \\ & \mathbf{Q}(\omega_k) & & & \\ & & \mathbf{Q}(\omega_k) & & \\ & & & \ddots & \\ & & & & \mathbf{Q}(\omega_k)) \end{bmatrix} \in \mathbb{C}^{LM \times LM} \tag{5.27}$$

$$\underline{\rho}(\omega_k) \triangleq \begin{bmatrix} \underline{a}(\omega_k) \\ e^{j\omega_k}\underline{a}(\omega_k) \\ \vdots \\ e^{j(L-1)\omega_k}\underline{a}(\omega_k) \end{bmatrix} \in \mathbb{C}^{LM \times 1} \tag{5.28}$$

Assuming that the set of spectral parameters, $\{\mathbf{Q}(\omega_k), \alpha(\omega_k)\}_{k=0}^{K-1}$, is available, the criterion in (5.25) can be re-written as:

$$\min_{\underline{x}_m} \sum_{k=0}^{K-1} \left\{ \left[\underline{\tilde{y}} - \alpha(\omega_k)\underline{\rho}(\omega_k)\right]^H \mathbf{D}_k^{-1} \left[\underline{\tilde{y}} - \alpha(\omega_k)\underline{\rho}(\omega_k)\right] \right\} \tag{5.29}$$

Since the data snapshots $\{\underline{y}_l\}_{l=0}^{L-1}$ overlap, a single sample might appear in $\underline{\tilde{y}}$ more than once. The locations of each sample's instances in $\underline{\tilde{y}}$ can be described by the following equation:

$$\underline{\tilde{y}} = \mathbf{S}_a \cdot \underline{x}_a + \mathbf{S}_m \cdot \underline{x}_m \tag{5.30}$$

where $\mathbf{S}_a \in \{0,1\}^{LM \times P_a}$ and $\mathbf{S}_m \in \{0,1\}^{LM \times P_m}$ are the corresponding selection matrices for the available and missing data vectors, respectively.
Using these notations, (5.29) becomes:

$$\min_{\underline{x}_m} \sum_{k=0}^{K-1} \left\{ \left[\mathbf{S}_m \cdot \underline{x}_m - (\alpha(\omega_k)\underline{\rho}(\omega_k) - \mathbf{S}_a \cdot \underline{x}_a)\right]^H \mathbf{D}_k^{-1} \left[\mathbf{S}_m \cdot \underline{x}_m - (\alpha(\omega_k)\underline{\rho}(\omega_k) - \mathbf{S}_a \cdot \underline{x}_a)\right] \right\} \tag{5.31}$$

Note that this expression has a structure similar to (5.22), which was received in the GAPES development, only here we have also a weight matrix.
Minimizing it with respect to $\underline{x}_m$ gives:

$$\underline{\hat{x}}_m = [\mathbf{S}_m^T \mathbf{D}_s \mathbf{S}_m]^{-1} \cdot [\mathbf{S}_m^T \mathbf{D}_v - \mathbf{S}_m^T \mathbf{D}_s \mathbf{S}_a \underline{x}_a] \tag{5.32}$$

where:

$$\mathbf{D}_s \triangleq \sum_{k=0}^{K-1} \mathbf{D}_k^{-1} \tag{5.33}$$

$$\mathbf{D}_v \triangleq \sum_{k=0}^{K-1} \mathbf{D}_k^{-1} \alpha(\omega_k) \underline{\rho}(\omega_k) \tag{5.34}$$

5.4 Comparing GAPES and MAPES-CM

The two algorithms are compared according to three categories: capabilities, complexity and performance. The next sub-sections describe each category, where the performance are demonstrated using the numerical example from Chapter 4.

5.4.1 Capabilities

The two algorithms are both iterative algorithms, derived from the APES algorithm and both of them use this algorithm as the first step of each iteration.

According to [36], since GAPES is an adaptive-filtering based method, it can deal much better with consecutively lost samples (i.e., gaps) than with scattered losses. MAPES-CM, on the other hand, should work well for both gaps and arbitrary loss patterns.

5.4.2 Complexity

The complexity was calculated according to our C implementation of both GAPES and MAPES-CM algorithms, where in the case of MAPES-CM, several of the algorithm's steps were efficiently implemented.

For example, by using $\mathbf{D}_k^{-1} = I_{L\times L} \otimes \mathbf{Q}\left(\omega_k\right)^{-1}$ it turns out that we can invert an $M \times M$ matrix instead of an $LM \times LM$ matrix. $\mathbf{Q}\left(\omega_k\right)^{-1}$ is obtained using the matrix inversion lemma [37] which leaves the algorithm with the need to invert only the $M \times M$ sample-covariance matrix, $\mathbf{R}$, once every iteration (this lemma is applied in a similar way also in the GAPES implementation). Also, multiplying $\mathbf{S}_m$ and $\mathbf{S}_a$ matrices by other matrices is implemented using only additions, since these matrices contain only zero and one values.

We refer here to the complexity per a steady-state iteration (i.e., not the first iteration), assuming the samples are real-valued:

GAPES takes a total of

$M^2L + M^3 + M(K\cdot\log[K]) + K(4M^2 + 20M + 12 + 2LP + 2LP_m + 4LP_m^2 + 4L) + 4P_m^3 + 4P_m^2$ multiplications and $2K$ complex divisions per iteration.

MAPES-CM takes a total of

$M^2L + M^3 + M(K\cdot\log[K]) + K(22M^2 + 20M + 12 + 4LM) + 2LMP_m + 4P_m^3 + 4P_m^2$ multiplications and $3K$ complex divisions per iteration.

To refresh one's memory: P is the length of the data-sequence, containing both available and missing samples, where P_m is the number of missing samples. K is the size of the frequency grid which is used for the spectral estimation, and the data-sequence is partitioned into $L = P - M + 1$ overlapping data snapshots, each containing M samples, as explained in sub-section 5.1.1.

From these expressions it is hard to determine which algorithm is more complicated, and indeed it depends on the values of the different parameters mentioned above. Let's take a typical case of our application as an example, where: $K = P$, $M = \frac{P}{4}$, hence $L \simeq \frac{3}{4}P$. In this case, by comparing the number of multiplications per iteration that GAPES requires to the number required by MAPES-CM, we get:

$$\frac{\textbf{GAPES}}{\textbf{MAPES} - \textbf{CM}} \simeq \frac{29P + 48P_m^2}{35P + 6P_m}$$

So, for example, for $P = 16$ and $P_m = 4$, GAPES requires twice the number of multiplications that MAPES-CM requires. The differences become smaller as $P_m << P$.

5.4.3 Performance

Our simulations show that in the simple cases, where the missing samples are surrounded by plenty of available samples, both algorithms perform well. However, since our application can tolerate only short segments of data (since long segments require a lot of storing space and/or a large delay in the system) we needed to test it under more difficult conditions.

Presented here are results of two simulations, comparing GAPES and MAPES-CM reconstruction capabilities: We use the same signal as in section 4.2 (the one presented in figures 4.3-4.5), containing two sinusoids at close frequencies, this time coded in MP3 format. After going through a channel, with simulated packet-losses, the received MP3 file is decoded and the lost segments are reconstructed in the DSTFT domain, using the proposed concealment algorithm, as described in the next chapter (Chapter 6). The reconstruction was done twice: once by using GAPES and then by using MAPES-CM, in order to compare between them. The figures show the reconstruction results in the MDCT domain, at bin No. 5 (which was also taken as an example in section 4.2).

Figure 5.4 shows a case where 3 consecutive packets (i.e., 6 consecutive MDCT frames) are missing. The estimation is based on a data sequence of length $P = 18$. The filter's length is $M = 5$. Here GAPES is used with its special initialization. As can be seen, MAPES-CM achieves better estimation results in this case, although GAPES had a better starting position (the result of the first iteration was closer to the target signal). The MSE for GAPES is $5.77 \cdot 10^{-4}$ and for MAPES-CM it is $9.66 \cdot 10^{-5}$.

Figure 5.5 shows a different case, where there are 3 short and very close gaps. The estimation is based on a data sequence of length $P = 22$. In this case, GAPES cannot use its special initialization to handle this loss pattern and is forced to use simple zero substitution instead (as MAPES-CM uses). Again, MAPES-CM converges closer to the right values than GAPES does. The MSE for GAPES is $2.02 \cdot 10^{-4}$ and for MAPES-CM it is $6.41 \cdot 10^{-5}$.

It's important to note that although these results agree with previously reported ones [36, 9] (although, without a **direct** comparison between GAPES and MAPES-CM). However, the results of our subjective tests (reported in Chapter 7) show that by dividing the concealment process into several sessions (as described in section 6.2.1), corresponding to loss patterns that GAPES can handle, and using its special initialization (see section 5.2.1), GAPES's average performances are slightly better than those of MAPES-CM, where the differences become smaller as the loss rate increases.

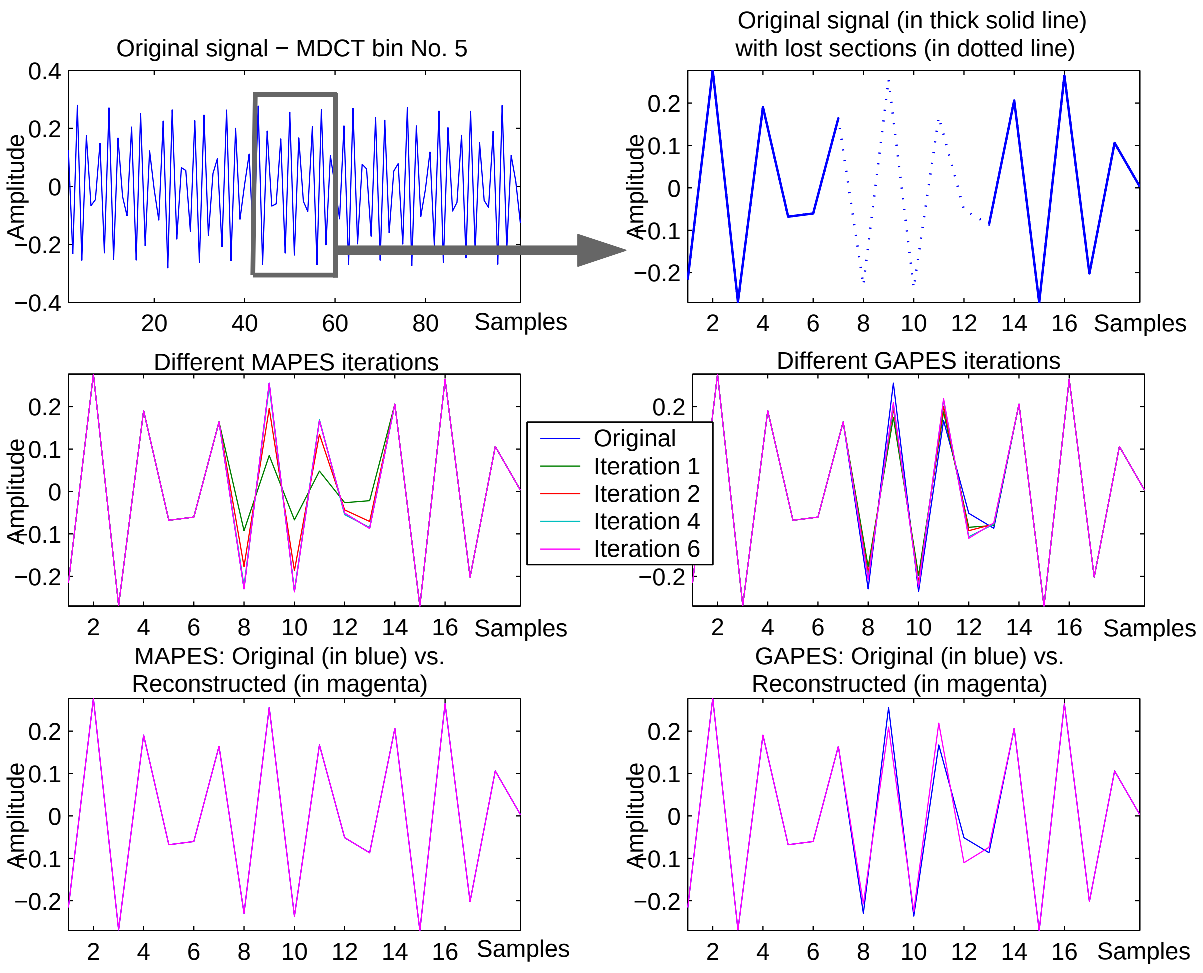


Figure 5.4: MAPES vs. GAPES, Example 1: Samples No. 8-13 are missing.

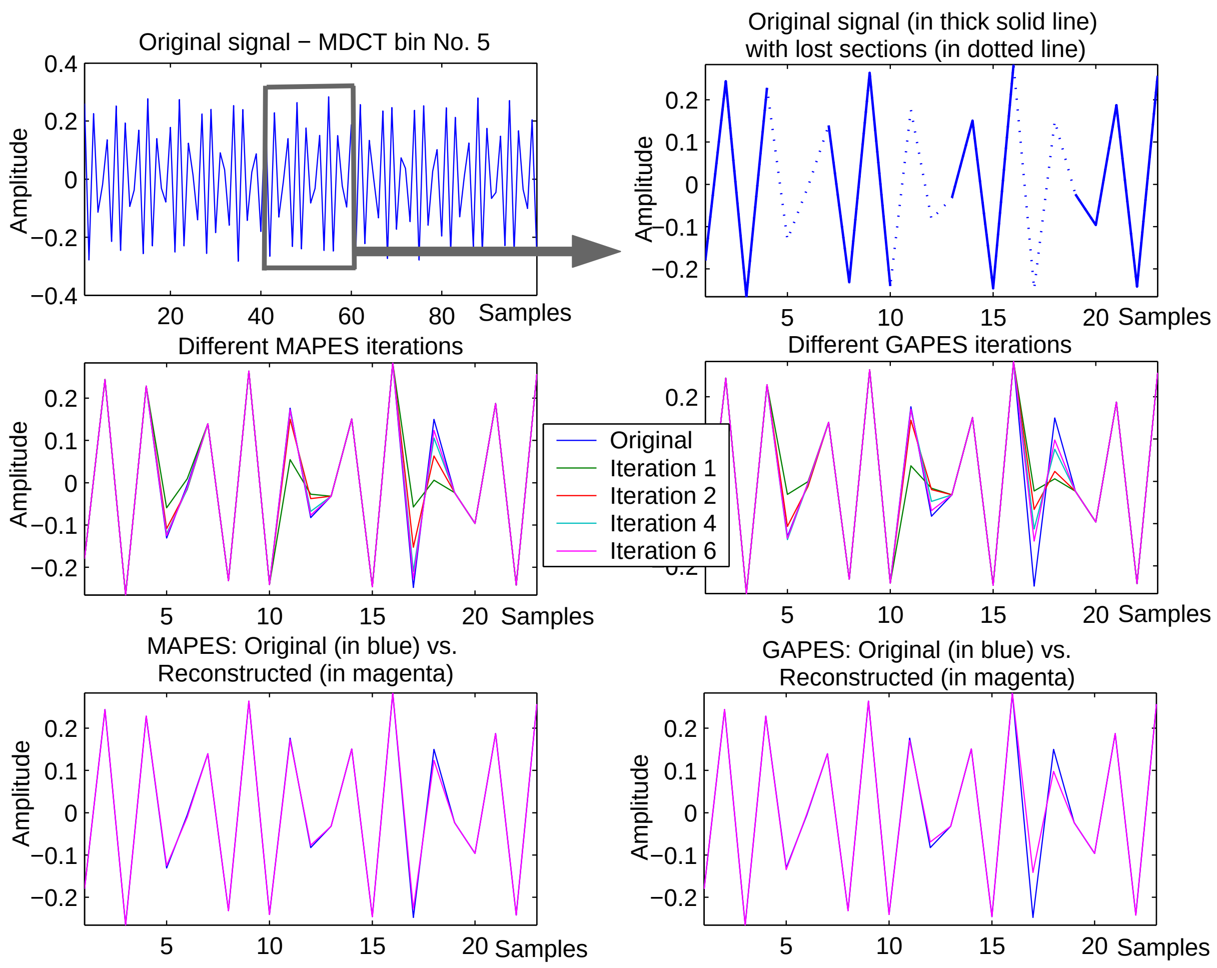


Figure 5.5: MAPES vs. GAPES, Example 2: Samples No. 5-6, 11-12 and 17-18 are missing.

Chapter 6

Proposed Packet Loss Concealment Algorithm

This chapter describes the proposed algorithm for packet loss interpolation in the DSTFT domain. We describe here two versions of the algorithm that are based on the two reconstruction algorithms: GAPES and MAPES-CM, that were introduced in Chapter 5. Also, the possibility of using a similar algorithm in the MDCT domain is explored. The chapter is organized as follows: The first section describes a procedure that was developed in this work, and that is part of the algorithm, for efficiently converting data from the MDCT domain to the DSTFT domain, and vise versa. The Second section describes the algorithm and the location of the concealment block within the MPEG decoder.

6.1 Converting Data from the MDCT to the DSTFT and Vice Versa

In this work we chose to use interpolation in the DSTFT domain. In order to convert the data from MPEG's compression domain to the concealment domain, we developed expressions for direct conversion from MDCT to DSTFT, and vice versa. The conversion procedure is presented here, while the full mathematical development can be found in Appendix B.

As was mentioned in section 4.1, the conversion uses the same time-segments originally used by the MDCT, each of length $2N$ samples with 50% overlap between consecutive segments. Since applying a strait-forward DFT on the signal (i.e., using a rectangular window function) introduces boundary effects, it is better to multiply the signal by a window function that will attenuate the edges of the segment prior to applying the DFT. In general, the only requirement of such a window function is that it has to be a symmetric window that has two halves that complement each other to the value of 1. This requirement arises from the need to make the conversion reversible, i.e., to allow for perfect reconstruction of the signal after the inverse-DFT using an OLA procedure. Our implementation uses a symmetric (even length) Hann window:

$$w[n] = \sin^2\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right), \quad 0 \le n \le 2N-1 \tag{6.1}$$

As Fig. 4.1 showed, a reconstruction of a single whole segment of $2N$ samples requires 3 consecutive MDCT blocks. 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. Since the MPEG standard defines 4 possible window types to be used during the MDCT, and considering the allowed ordering of the windows, we could theoretically have 12 different expressions for the conversion of 3 consecutive MDCT segments into one DSTFT segment. But, after changing the order of summations and applying some algebraic manipulations, all these expressions converge into one general structure. The same happens in the conversion of DSTFT segments back to the MDCT domain.

The expressions for both conversion directions use four functions: g_d^1, g_r^1, g_d^2

and g_r^2, which we refer to as "building-block" functions: Each of these functions is chosen among 3 possible complex functions, according to the window type of the segment that we wish to convert and on the window types of the neighboring segments. These functions are based on the 3 half-window units that form the MP3 window types (presented in Fig. 3.8) and in total there are 12 such possible functions. The assignment of the "building-block" functions for the different cases in the direct and reversed conversions respectively, is given in Table 6.1 and 6.2, and the functions themselves are described in section 6.1.1.

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

$$\begin{aligned} X_{(p)}^{DSTFT}[m] &= \sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k] \cdot (g_d^1[m,k] + (-1)^m \cdot g_r^2[m,k]) \\ &+ \sum_{k=0}^{N-1} X_{(p-1)}^{MDCT}[k] \cdot g_d^2[m,k] \\ &+ \sum_{k=0}^{N-1} X_{(p+1)}^{MDCT}[k] \cdot ((-1)^m \cdot g_r^1[m,k]) \qquad ,0 \leq m \leq N \end{aligned} \tag{6.2}$$

Where $p \in \mathbb{Z}$ is the current block time-index, and $(p-1)$,p,$(p+1)$ denote 3 consecutive MDCT blocks. Since the DSTFT is performed on a $2N$ real-valued sequence, the output is conjugate-symmetric in the frequency domain. Therefore, it suffices to calculate only the first half of the output, as done in (6.2).

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 below, where * denotes a complex-conjugate value, and N_{block} is the length of the MDCT that is applied on the frame (i.e., N in the case of long windows (types 0,1 and 3) and $N_s = \frac{N}{3}$ in the case of a short window (type 2)).

$$
\begin{aligned}
X_{(p)}^{MDCT}[k] &= \\
&\frac{1}{N \cdot N_{block}} \sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m] \cdot ((g_d^1[m,k]^* + g_r^1[m,k]^*) + (-1)^m \cdot (g_d^2[m,k]^* + g_r^2[m,k]^*)) \\
&+ \frac{1}{N \cdot N_{block}} \sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m] \cdot (-1)^m \cdot (g_d^1[m,k]^* + g_r^1[m,k]^*) \qquad (6.3) \\
&+ \frac{1}{N \cdot N_{block}} \sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m] \cdot (g_d^2[m,k]^* + g_r^2[m,k]^*) \qquad , 0 \le k \le N-1
\end{aligned}
$$

Table 6.1: Function assignment for the direct conversion

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 Next window is Short: g_r^1_short	Previous window is Start: g_d^2_short2long Previous window is Short: g_d^2_short	g_r^2_short
Stop	g_d^1_short2long	g_r^1_long	g_d^2_short	g_r^2_long

The fact that there is 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 and stored off-line. The same 12 "building-block" functions are used for both conversion directions. Each function can be represented by $2N^2$ real values that need to be calculated off-line and stored.

Table 6.2: Function assignment for the reverse conversion

Window type \ Function	$g_d^1[m,k]$	$g_r^1[m,k]$	$g_d^2[m,k]$	$g_r^2[m,k]$	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

6.1.1 The "Building-Block" Functions

Introduced here are the expressions for some of the functions that were mentioned in section 6.1 as the "building-block" functions for the direct and reverse conversions, where $0 \le m \le N$ and $0 \le k \le N-1$. The rest of the functions and the full mathematical development can be found in Appendix B.

$$g_d^1\text{\_long}[m,k] = \sum_{n=0}^{N-1} \cos\left[\frac{\pi}{N} \cdot \left(n + \frac{N+1}{2}\right) \cdot \left(k + \frac{1}{2}\right)\right] \cdot h^{long}[n] \cdot w[n] \cdot e^{-j\frac{\pi}{N}nm} \tag{6.4}$$

$$g_r^1\text{\_long}[m,k] = \tag{6.5}$$

$$\sum_{n=0}^{N-1} \cos\left[\frac{\pi}{N} \cdot \left(n + \frac{N+1}{2}\right) \cdot \left(k + \frac{1}{2}\right)\right] \cdot h^{long}[n] \cdot w[N-n-1] \cdot e^{-j\frac{\pi}{N}nm}$$

$$g_d^2\text{\_long}[m,k] = \tag{6.6}$$

$$\sum_{n=0}^{N-1} \cos\left[\frac{\pi}{N} \cdot \left(n + N + \frac{N+1}{2}\right) \cdot \left(k + \frac{1}{2}\right)\right] \cdot h^{long}[N-n-1] \cdot w[n] \cdot e^{-j\frac{\pi}{N}nm}$$

$$g_d^2\text{\_long}[m,k] = \tag{6.7}$$

$$\sum_{n=0}^{N-1} \cos\left[\frac{\pi}{N} \cdot \left(n + N + \frac{N+1}{2}\right) \cdot \left(k + \frac{1}{2}\right)\right] \cdot h^{long}[N-n-1] \cdot w[N-n-1] \cdot e^{-j\frac{\pi}{N}nm}$$

$g^1_{d/r\_\text{short2long}}[m,k]$ and $g^2_{d/r\_\text{short2long}}[m,k]$ are the same as the functions $g^1_{d/r\_\text{long}}[m,k]$ and $g^2_{d/r\_\text{long}}[m,k]$, respectively, except that they use the window function $h^{short2long}$ instead of h^{long}. The expression for $g^1_{d/r\_\text{short}}[m,k]$ and $g^2_{d/r\_\text{short}}[m,k]$ are more cumbersome and are therefore given in appendix B.

6.1.2 Complexity Comparison

Besides the benefit of having a single expression instead of 12 for each conversion direction, our conversion scheme is also more efficient than the traditional procedure of using inverse-MDCT, OLA, and only then applying a DFT on the restored samples. Table 6.3 compares the complexity of each procedure. This calculation takes into account that **fast** fourier transform (FFT) could not have been applied, since the length of the DFT segments is not an integer power of 2. The table shows that our conversion is more efficient than the traditional conversion: From the MDCT domain to the DSTFT our procedure requires 1.6 times less the number of multiplications required by the traditional procedure, and from the DSTFT to the MDCT it requires 3.5 times less, so in total we have a factor of around 5 less.

Table 6.3: Comparing the complexity of conversion schemes

	Efficient Conversion		Traditional Conversion	
	Mults	Adds	Mults	Adds
MDCT to DSTFT	$6N^2$	$8N^2$	$10N^2$	$10N^2$
DSTFT to MDCT	$8N^2$	$20N^2$	$28N^2$	$26N^2$

6.2 The Concealment Algorithm

As was already mentioned, this work focuses on a receiver-based solution. Hence, the concealment algorithm is applied during the decoding process and uses only data available at the receiver. Figure 6.1 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.

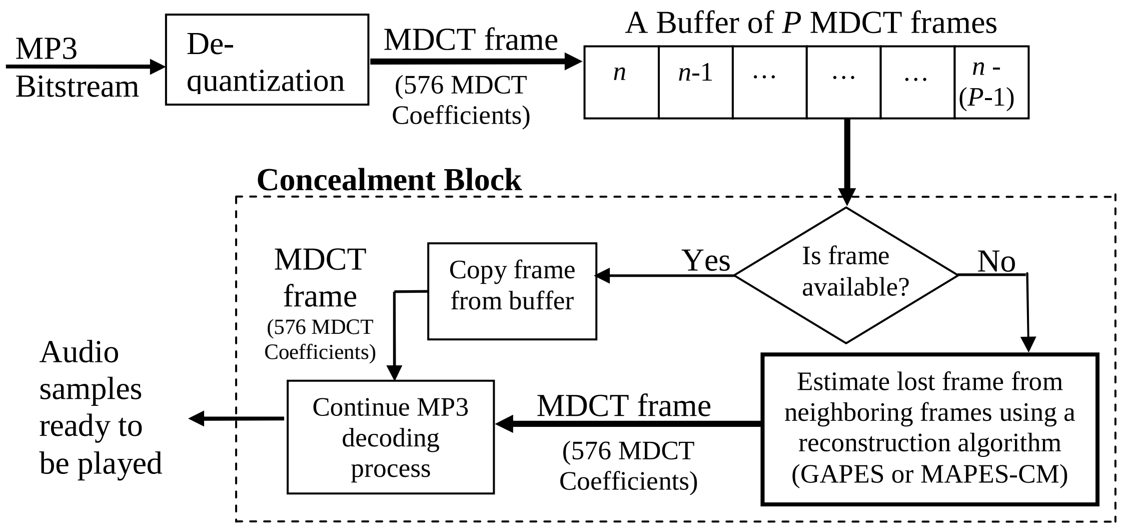


Figure 6.1: A diagram of the proposed decoding process that includes a concealment block

The P most recent MDCT frames are stored in a buffer, indexed from 0 to $P-1$. 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 (as described in section 4.1). Then, the next frame to be played (according to the system's delay, if such delay exists) is copied from the buffer and decoded into waveform samples. In the case where that particular frame is actually a missing frame, we estimate its MDCT coefficients before continuing with the decoding process. Due to packet loss, several MDCT frames in the buffer may be missing, so in order to save the computational overhead that results from activating the concealment block for each missing frame separately, usually several missing frames are concealed together. Hence, we refer to a concealment of one or more MDCT frames at once as a *concealment session*.

Figure 6.2 shows an example to the process that takes place in a concealment session: In this example, 2 MDCT frames are missing in the buffer. The MDCT frames in the buffer are converted to the DSTFT domain, where the data along the time axis at each frequency bin is considered as an independent complex signal, and a single iteration of the reconstruction 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 mentioned in the previous section 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 can improve the performance of the reconstruction algorithms, since the estimation of the lost data will then be based on data both from the past and the future. 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.

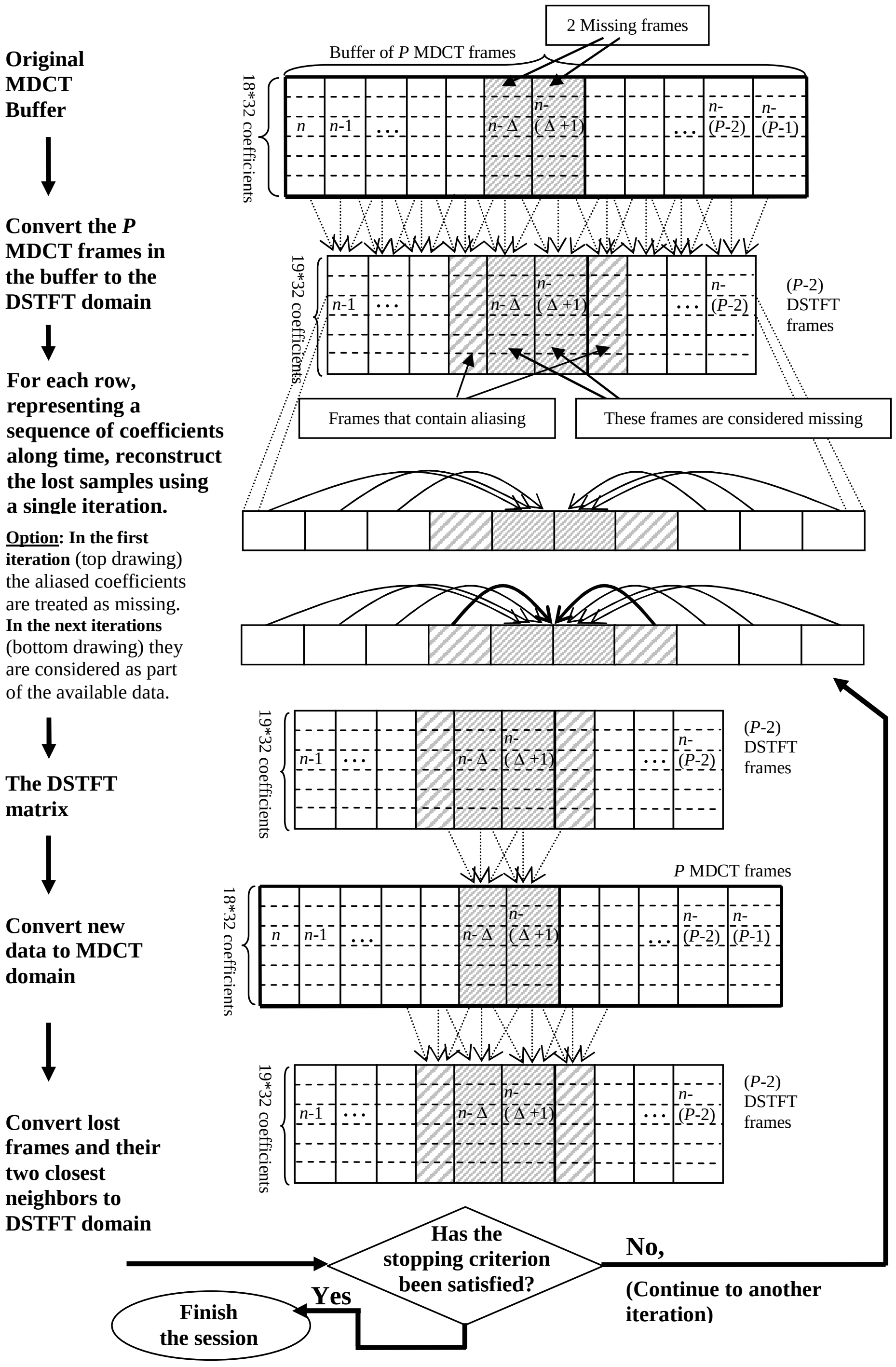

2 Missing frames
Buffer of P MDCT frames
Original MDCT Buffer
18*32 coefficients
n
n-1
n-Δ
n-(Δ+1)
n-(P-2)
n-(P-1)
Convert the P MDCT frames in the buffer to the DSTFT domain
19*32 coefficients
(P-2) DSTFT frames
For each row, representing a sequence of coefficients along time, reconstruct the lost samples using a single iteration.
Frames that contain aliasing
These frames are considered missing
Option: In the first iteration (top drawing) the aliased coefficients are treated as missing. In the next iterations (bottom drawing) they are considered as part of the available data.
The DSTFT matrix
P 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

Figure 6.2: The flow of the proposed algorithm for estimating lost MDCT frames

The buffer of length P, which holds the most recent MDCT frames, should not be too short, so there is enough data for the estimation, and 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 P columns (as shown in Figure 6.2).

Let's assume that there are $P_m \leq P$ 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. As was already mentioned, concealing more than one frame at a time can reduce the computational overhead of the whole concealment process. We can either choose to conceal all the missing frames in the buffer in one concealing session, or a smaller number according to a certain heuristic criterion: for example, lost frames that are gathered closely together will be concealed in one concealing session and distant frames will be concealed in separate sessions. The issue of choosing the number of frames to be concealed in one concealing session also depends on the reconstruction algorithm that is used, since as was stated in chapter 5, GAPES can successfully deal with fewer loss-patterns than MAPES-CM can. Therefore, when using GAPES, it is sometimes better to restrict the concealing session to a selected area inside the buffer which contains a loss pattern that GAPES can deal with by applying its special initialization. This option is discussed in section 6.2.1. Currently, for simplicity, let us assume that all the missing frames in the buffer are concealed in the same concealing session, based on all the available frames in the buffer. Since each MP3 packet contains 2 MDCT frames, one concealment session always contains at least 2 concealed frames (i.e., $2 \leq P_m < P$).

In the first stage of the algorithm, all the MDCT frames in the buffer are converted to the DSTFT domain, since all of them participate in the reconstruction. 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 $(P-2)$ 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 $(P-2)$. 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 bin along the time axis, is considered as an independent complex signal with missing samples. The missing samples of each row vector are estimated from the available samples by applying a single iteration of the GAPES or MAPES-CM algorithm, as described in sections 5.2 and 5.3, respectively. Note that the locations of the missing samples in the vector are actually the indices of the missing MDCT frames in the buffer, since every sample in the row vector corresponds to a single MDCT frame in the buffer. However, due to the fact that the DSTFT frames closest to the lost frames contain aliasing (as was explained in section 4.1) one might consider to treat these frames too as missing data during the first iteration, although this creates a wider gap which is more difficult to handle. Our simulations show that in the case of MAPES-CM this procedure gives a better starting point to the following iterations and therefore better results.

After a single iteration of the reconstruction algorithm is applied on each of the DSTFT matrix's row vectors, 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 smoother 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 closest to the loss will give a better basis for the next 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 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 re-calculating 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 (only DFT and inverse DFT). The limitation of such a conversion would be the need for more memory space, and more data structures in order to hold the time-samples of the frames that require merging.

The final stage of the algorithm is to decide whether the estimation is good enough. As a stopping criterion for the concealment algorithm, we use a threshold over the average squared difference per sub-band, as defined in (6.8). This ratio is calculated over all rows in the DSTFT matrix corresponding to a sub-band (each sub-band contains 19 rows):

$$D_{sb} = \frac{1}{19}\sum_{i=0}^{19}\left(\frac{1}{S}\sum_{j\in S}\frac{\left|\hat{x}_{i,j}^{NEW} - \hat{x}_{i,j}^{OLD}\right|^2}{\left|\hat{x}_{i,j}^{OLD}\right|^2}\right) \tag{6.8}$$

Where $0 \leq sb \leq 31$, S is the set of indices of the P_m lost MDCT frames that are concealed in this session: $|S| = P_m$ and $\hat{x}_{i,j}$ are the reconstructed DSTFT coefficients. D_{sb} is compared to a pre-defined threshold. If it is smaller than the threshold, then the estimation is sufficiently good, so we can stop the reconstruction of this sub-band at this point and continue with the MP3 decoding process. If it is larger than the threshold, then we apply another iteration on this sub-band, as shown in Figure 6.2. 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 sub-band of the uniform filter-bank:

1. Initialization:
   - Store zero values in place of each lost MDCT frame.
   - Determine the parameters for the next concealment session: P, the length of buffer section, P_m, the number of frames to be reconstructed and M (See explanation in section 6.2.1).
   - Convert the MDCT buffer to the DSTFT domain (as described in section 6.1).
2. Estimate the DSTFT coefficients of the lost frames:
   - Treat each sequence of DSTFT coefficients at a certain frequency bin, along the time axis, as a complex signal with missing samples.
   - Estimate the lost samples by applying a single iteration of the reconstruction algorithm (GAPES or MAPES-CM) on each sequence (as de-

scribed in chapter 5).

3. Reconstruct the values of the lost MDCT frames:

   Convert the corresponding DSTFT frames into the MDCT domain (as described in section 6.1).

4. Re-calculate the values in the DSTFT domain:

   Re-calculate the values of the lost frames and their closest neighbors by converting the corresponding MDCT frames to the DSTFT domain (as described in section 6.1).

5. Check the stopping criterion, per sub-band (according to (6.8)):

   If it is satisfied, stop the algorithm for this sub-band and use the MDCT frames that were obtained last. If not, return to (2) for another iteration.

6.2.1 The Differences Between Using GAPES and MAPES-CM

As was already mentioned, the algorithm can be activated using either the GAPES or the MAPES-CM reconstruction algorithms. Since both algorithms use APES as the first step of each iteration all that needs to be done is to apply the second step according to the desired algorithm, as described in sections 5.2-5.3.

However, as explained in the end of section 5.2.1, GAPES performs better with a special initialization that can't fit any loss pattern (as opposed to MAPES-CM, where the lost samples are simply set to zero before the first iteration). Hence, in case that GAPES is the chosen reconstruction algorithm, our solution is to divide the concealment process into several sessions, where in each session the parameters: P, P_m, and M, are redefined to contain only part of the loss, a part that GAPES can handle (the reminder of the lost will be concealed in the next session). This is obtained by selecting a section in the original buffer that contains only part of the missing samples and their closest neighbors and using it during the concealment session. For example, let's assume that we have a buffer of $P = 16$ frames, where frames 7-8 and 13-16 are missing. Choosing a value of the filter length, $M > 2$, results in a large number of zeroed $\underline{y}_l$ vectors in comparison to the total number of $\underline{y}_l$ vectors (which is $L = P - M + 1$). For example, using $M = 4$ results in 9 zeroed vectors out of 13! However, limiting the buffer to only the first 12 frames and estimating only 2 lost frames at once, instead of the whole 6, results in 5 zeroed vectors out of 9, which is more balanced. I.e., the parameters of the concealment session will be redefined to $P = 12$ and $P_m = 2$. This promises better interpolation results, but requires more concealing sessions in order to conceal the remaining frames.

6.2.2 Applying The Concealment Algorithm in the MDCT Domain vs. The DSTFT Domain

Although this work describes an algorithm that is applied in the DSTFT domain, it is important to note that we also considered applying the reconstruction algorithms directly in the MDCT domain. The benefit of such configuration is that it requires fewer calculations since there is no need to convert the data to any other domain, and since the signal is real-valued. The limitations of this configuration, however, are those mentioned in section 4.1: The problem of dealing with windows having different resolutions, and the rapid sign changes. Our subjective tests, reported at Chapter 7, showed that this configuration is inferior to the DSTFT configuration, especially at high loss rates. Next, we compare the complexity of the algorithm when it is implemented in the MDCT domain, versus its implementation in the

DSTFT domain.

The first aspect is the effect of using complex signals in the DSTFT domain, versus using real-valued signals in the MDCT domain. Using the parameters of the proposed algorithm, i.e., P, P_m and M, where $K = P$ and $L = P - M + 1$, and by taking the same typical case from section 5.4.2 as an example, it turns out that using GAPES with a complex signal costs around $\simeq 1.24$ times more multiplications per iteration than using it with a real-valued signal. In the case of MAPES, the ratio is $\frac{\text{complex}}{\text{real}} \simeq \frac{37P+12P_m}{32P+6P_m}$, which for typical parameter values is also $\simeq 1.2$.

The second aspect to be compared is the need to convert the data between the two domains in the DSTFT implementation. This procedure is done once every iteration and involves all the frequency bins inside a single sub-band (which in the case of MP3 is $N = 18$). The conversion from MDCT to DSTFT requires $6N^2$ multiplications per sub-band, and the conversion from DSTFT to MDCT requires $8N^2$ multiplications per sub-band.

In total, we need to compare the two implementations by looking at a single iteration applied on a whole sub-band:
The DSTFT implementation requires to convert the data to the DSTFT domain, then to activate GAPES or MAPES-CM: on $(N-1)$ complex signals and 2 real-valued signals (the DC coefficients), and last, to convert back to the MDCT domain.

$$\begin{aligned}\text{DSTFT implementation} &\simeq 14N^2 + (N-1)\cdot\text{complex GAPES/MAPES}\\ &\qquad\qquad +2\cdot\text{real GAPES/MAPES}\\ &\simeq 14N^2 + 1.2N\cdot\text{real GAPES/MAPES}\end{aligned}$$

On the other hand, using the algorithm in the MDCT domain is simply activating

GAPES or MAPES-CM on N real-valued signals.

$$\text{MDCT implementation} \simeq N \cdot \text{real GAPES/MAPES}$$

Taking into account that the complexity of both GAPES and MAPES-CM at our typical parameter values is in the order of $\mathrm{O}(N^3)$, the addition of $14N^2$ is not very significant, and hence both implementations have similar complexity.

Chapter 7

Description and Analysis of Concealment Results

To evaluate the quality of the proposed algorithm, we had to compare it with previously reported algorithms for packet loss concealment, designed for wide-band audio signals encoded by an MPEG audio coder. In order to do so we considered several measures, both objective and subjective. Since simple objective measures, such as MSE (Mean Squared Error), do not reflect the sensation created in a human listener, we had to look for other alternatives. There exists an algorithm for objective assessment of the perceptual quality of wide-band audio signals (PEAQ - Perceptual Evaluation of Audio Quality [38]). However, after using it to test our files we realized that PEAQ's results didn't correlate with the results of the subjective tests and were in many cases inconsistent, especially at high loss rates. The reason for this behavior is probably since the PEAQ algorithm, as other similar standards such as PESQ (Perceptual Evaluation of Speech Quality [39]), is incapable of accurately assessing the perceptual effect of artifacts caused by packet loss and packet loss concealment applications. Specifically, the PEAQ standard is based on the BS.1116 standard [40], which is designed for detecting small coding impairments and may produce inaccurate results when applied to signals with large and obvious impairments, such as in the case of high-rate packet loss. For this reason, we finally chose to compare the different algorithms, using only subjective listening tests, where the comparison is given in terms of listeners' preference. The first section of this chapter contains some details about the testing environment. The results of the subjective tests are reported next, in section 2.

In addition, the third section compares the performance of the proposed algorithm in different activation modes, i.e., using MAPES-CM as the reconstruction algorithm versus using GAPES. Also, we examine the option of using the reconstruction in the MDCT domain instead of in the DSTFT domain. Finally, we conclude the chapter.

7.1 Testing Environment

The subjective tests were carried out by informal listening. All the participants are inexperienced listeners with normal hearing and in the age range of 24-35 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. The audio files that were used in the tests are specified in Table 7.1. All the files are stereo signals ,15-17 seconds long each, sampled at 44.1 kHz and coded by the LAME MP3 [41] 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.

Table 7.1: 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
4	Jazz6.wav	Jazz music
5	Flute2.wav	Flute and piano

7.2 Comparing the Proposed Algorithm to Previously Reported Algorithms

16 listeners were asked to compare pairs of files, as described in the previous section, where in this test we examine the performance of the proposed algorithm versus previously reported works: *packet repetition*, suggested in the MP3 standard [[18], annex E], and *statistical interpolation* (SI), suggested in [6]. The results are presented in Table 7.2, where in the first three tables the numbers for each pair indicate how many listeners voted in favor of the method. The forth table presents the overall distribution, averaged over all the files. The results clearly show that the proposed algorithm performs better than the two previously reported methods. Moreover, when compared to the uncorrupted signal (i.e., after conventional MP3 decoding) at 10% loss rate, the proposed solution performed so well, that some of the listeners confused the concealed signal with the original one.

It is important to note that the SI method was extended 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. Also, in this test the proposed algorithm was activated using the GAPES reconstruction algorithm.

The packet repetition method introduces no delay since the decoder simply replaces a lost MDCT frame with the previous frame. In the proposed algorithm

Table 7.2: Comparative test results of GAPES-in-DSTFT vs. Previously reported works. The numbers indicate how many listeners voted in favor of each method.

	Proposed Solution vs. Repetition									
	File No. 1		File No. 2		File No. 3		File No. 4		File No. 5	
Loss Rate	Proposed Solution	Rep.	Proposed Solution	Rep.	Proposed Solution	Rep.	Proposed Solution	Rep.	Proposed Solution	Rep.
10%	**14**	2	**15**	1	**16**	0	**10**	6	**16**	0
20%	**16**	0	**16**	0	**15**	1	**14**	2	**15**	1
30%	**16**	0	**15**	1	**12**	4	**14**	2	**14**	2

	Proposed Solution vs. SI									
	File No. 1		File No. 2		File No. 3		File No. 4		File No. 5	
Loss Rate	Proposed Solution	SI	Proposed Solution	SI	Proposed Solution	SI	Proposed Solution	SI	Proposed Solution	SI
10%	**16**	0	**16**	0	**15**	1	**15**	1	**16**	0
20%	**16**	0	**16**	0	**16**	0	**16**	0	**16**	0
30%	**16**	0	**16**	0	**14**	2	**16**	0	**16**	0

	Proposed Solution vs. Uncorrupted Original									
	File No. 1		File No. 2		File No. 3		File No. 4		File No. 5	
Loss Rate	Proposed Solution	Original	Proposed Solution	Original	Proposed Solution	Original	Proposed Solution	Original	Proposed Solution	Original
10%	4	**12**	5	**11**	2	**14**	2	**14**	2	**14**

	Distribution of votes					
	Proposed Solution vs. Repetition		Proposed Solution vs. SI		Proposed Solution vs. Uncorrupted Original	
Loss Rate	Proposed Solution	Repetition	Proposed Solution	SI	Proposed Solution	Original
10%	**88.75%**	11.25%	**97.5%**	2.5%	18.75%	**81.25%**
20%	**95%**	5%	**100%**	0%		
30%	**88.75%**	11.25%	**97.5%**	2.5%		

and in the SI method, a buffer of 9 MP3 packets, i.e., a buffer length of $P = 18$, 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 both methods was fixed to

$D(sb) = 10^{-2}$ (see (6.8)), 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 expressed as:

$$w[n] = \sin^2\left(\frac{\pi}{2N}\left(n + \frac{1}{2}\right)\right), \qquad 0 \leq n \leq 2N - 1 \tag{7.1}$$

which is equal to the square of the 'Long' window type used in the MP3 standard.

7.3 Comparing Different Activation Modes of the Proposed Algorithm

The first test reported in this section compares the performance of the proposed scheme, when used in the MDCT domain, versus using it as described in chapter 6, in the DSTFT domain. This test was done using GAPES as the reconstruction algorithm. 8 listeners were asked to compare pairs of files, each concealed by a different method and to determine which of the two file versions sounds less disturbing. The results are presented in Table 7.3 where in the first table the numbers for each pair indicate how many listeners voted in favor of the method and the second table shows the averaged distribution. The results show clearly that the algorithm performs better in the DSTFT domain, especially at high loss rates.

The next test compares the two possibilities of activating the proposed scheme, using the two reconstruction algorithms that were described in chapter 5: GAPES and MAPES-CM. Similar to the previous test, 8 listeners were asked to compare pairs of files, each concealed using the proposed algorithm but with a different reconstruction method. It is important to note that the listeners who participated in this particular test reported that it was much harder than previous tests, since in many of the cases the two files sounded very much the same. However, the results, presented in Table 7.4, show that in most cases GAPES performs slightly better than MAPES-CM, where the differences become smaller as the loss rate increases.

Table 7.3: Comparative test results of using the algorithm in the MDCT domain vs. using it in the DSTFT domain. The numbers indicate how many listeners voted in favor of each method.

	File No. 1		File No. 2		File No. 3	
Loss Rate	MDCT	DSTFT	MDCT	DSTFT	MDCT	DSTFT
10%	2	**6**	3	**5**	2	**6**
20%	0	**8**	0	**8**	1	**7**
30%	0	**8**	0	**8**	0	**8**

	Distribution of votes	
Loss Rate	MDCT	DSTFT
10%	29.1%	**70.9%**
20%	4.1%	**95.9%**
30%	0%	**100%**

Table 7.4: Comparative test results of GAPES vs. MAPES-CM. The numbers indicate how many listeners voted in favor of each method.

	File No. 1		File No. 2		File No. 3		File No. 4		File No. 5	
Loss Rate	GAPES	MAPES	GAPES	MAPES	GAPES	MAPES	GAPES	MAPES	GAPES	MAPES
10%	**7**	1	**4**	**4**	**8**	0	**5**	3	**6**	2
20%	**8**	0	1	**7**	**8**	0	**5**	3	**7**	1
30%	**6**	2	**4**	**4**	**5**	3	**5**	3	**6**	2

	Distribution of votes	
Loss Rate	GAPES	MAPES
10%	**75%**	25%
20%	**72.5%**	27.5%
30%	**65%**	35%

7.4 Conclusion

Several subjective tests were held in order to test the proposed algorithm: First it was compared to two other concealment methods that were previously reported in the literature. The results of this comparison proves that the proposed solution performs considerably better, even up to 30% loss rate.

Then, we compared the algorithm's different activation modes: The tests showed that although MAPES-CM can handle more loss patterns and has lower complexity demands, using GAPES gives better performance in most examined cases. And finally, the DSTFT domain proved to outperform the MDCT domain as the preferred concealment domain.

Chapter 8

Summary and Future Directions

8.1 Summary

A new algorithm for packet loss concealment was developed in this work. The algorithm is receiver-based and is designed for wide-band audio signals encoded by MPEG-audio coders. The reconstruction of the lost data is done in the DSTFT domain using either GAPES or MAPES-CM reconstruction algorithms. Evaluated by comparative subjective listening tests, the algorithm outperforms previously reported concealment methods, even at high loss rates.

We began by presenting internet audio streaming and the packet loss problem. The main issue was that the length of the gap, created by even a single lost audio packet, is wide (about 1000 samples long) and is therefore difficult to interpolate. The goal of the packet loss concealment process is to interpolate the gap so it won't be noticed by a human listener. However, most of the receiver-based concealment techniques that were reported so far in the literature are suitable only for speech signals, which have a relatively simple model to work by. In the category of general audio signals, such as music, there are very few receiver-based solutions reported. Two of them, which are suitable for signals coded with MPEG-audio coders, are the packet repetition method and the statistical interpolation (SI) method, which we compared our solution to.

After exploring the structure of the MPEG-audio coder and getting familiar with the concept of compression in the MDCT domain, it seemed reasonable to perform the concealment in the MDCT domain, as done in the SI technique. However, there are some flaws in working in the MDCT domain: First, the MDCT coefficients, much like DCT coefficients, show rapid sign changes along time. Second, the use of windows with different frequency resolutions makes estimation based on the gap's closest surroundings unreasonable. The solution we suggested to these two problems was to switch to the DSTFT domain, using windows of constant resolution, and perform the interpolation in this domain. Switching to the DSTFT domain causes a minor expansion of the gap: in case of Q lost packets, we get $2Q$ consecutive lost DSTFT coefficients at each frequency bin plus one corrupted neighbor in each side of the loss, instead of $2Q$ consecutive lost MDCT coefficients.

The GAPES algorithm, which our algorithm is based on, is an iterative algorithm which reconstructs the missing data assuming it has the same spectral content as the available data, using an adaptive filter-bank approach. The algorithm is a gapped-data version of the APES algorithm for spectral estimation. The MAPES-CM algorithm is a newer version that uses an ML-estimator approach to reconstruct the missing data. This version has slightly less complexity and is able to successfully deal with more loss patterns than GAPES does.

The proposed solution includes a concealment block that is situated in the MPEG-audio decoder, right after the de-quantization of the received data into MDCT coefficients: If a packet was lost, the missing MDCT frames are reconstructed and the decoding process continues as usual. The concealment algorithm itself is an iterative algorithm, where at each iteration the MDCT data is converted to the DSTFT domain, where a single iteration of GAPES or MAPES-CM is applied separately in each frequency bin. After all the coefficients are reconstructed they are converted back to the MDCT domain, where a stopping criteria is applied. If more iterations are needed, the data is converted again to DSTFT domain and the whole process is repeated.

As mentioned before, the proposed algorithm was compared to two previously reported concealment methods: Packet Repetition and SI. The comparison was done using comparative subjective listening tests, with various music types and at 10% - 30% loss rates. The tests show that the proposed algorithm performs better than previously reported concealment methods.

8.2 Main Contributions

Using the DSTFT domain for packet loss concealment

Unlike SI, the DSTFT was chosen as the concealment domain. The frequency domain has the benefit of representing well music signals, under the model of sines plus noise. The DSTFT coefficients along time at each frequency bin have a less fluctuating representation of the signal's amplitude, as compared to the MDCT coefficients along time.

Efficient conversion between MDCT and DSTFT domains

A process is developed in this work that enables efficient conversion of the data between the different domains, using a single expression for each conversion direction. The different window types are expressed by building-block functions which can be calculated off-line. This conversion has less complexity than using the straight-forward procedure of inv-MDCT and then DFT, assuming that FFT (radix-2) cannot be applied in this case.

Applying GAPES and MAPES-CM to the packet loss problem

A novel scheme is presented where these two algorithms are adjusted to solve the problem of audio packet loss.

8.3 Future Directions

Applying psychoacoustic rules in the interpolation process

In a perceptual coder, such as MP3, using psychoacoustic rules in the concealment process is called for. One option is to reduce the number of iterations on the basis of psychoacoustic considerations, by setting different stopping thresholds to different critical bands, according to SMR that was determined for each band at the encoder: A smaller SMR value 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 reconstruction, and vice versa.

Assuming that MPEG-audio coders quantize the signal according to the SMR ratios, we can get a rough estimate of these ratios at the decoder, from the lost packet's closest neighbors: 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.

Exploiting frequency bins inter-correlation in the interpolation process

In this work the proposed solution interpolated the lost DSTFT coefficients in each frequency bin separately, exploiting only the correlation along the time axis. However, since the signals in adjacent frequency bins are correlated, this inter-bin correlation can be used to improve the interpolation. Two-dimensional GAPES and MAPES-CM can be considered for this purpose.

Adding side-information at the encoder

Switching to a sender-based solution, by adding side information to the coded signal, can improve the reconstruction results. The side information can be added by means of data-hiding or simply as a separate field in the bit-stream. Such information can include, for example, information about the signs of the MDCT coefficients, or a polynomial approximation of the coefficients (along time, or along frequency).

Making proper adjustments for a Real-Time platform

Since the proposed algorithm is computationally demanding in terms of nowadays technology, certain adjustments are required in order to implement the algorithm on a real-time platform. Such adjustments can include using numerical shortcuts, especially with regard to matrix and vector calculations. Another aspect is to create a sub-optimal version of GAPES and MAPES-CM having less complexity. This can be done by, for example, reducing the size of the pre-defined frequency grid (K), or using the autocorrelation method to estimate the sample-covariance matrix, $\hat{\mathbf{R}}$, and the levinson-durbin algorithm to invert it.

Appendix A

A Guide to Psychoacoustics and MP3 coding

The MPEG-1 standard [18] was defined in 1992 by the ISO/IEC standardization body. The name, which is the first initials of Moving Pictures Experts Group, represents a group that has been set up by the ISO/IEC standardization body in 1988 to develop generic standards for coding video: i.e., moving pictures, associated audio and their combination. Since 1988 ISO/MPEG has been undertaking the standardization of compression techniques for video and audio, and now there exist also MPEG-2, MPEG-4 and soon MPEG-7 and MPEG-21 standards.

MPEG-1 Audio offers a choice of three independent layers of compression, with increasing codec complexity and compressed audio quality: Layer 1 is the simplest, best suiting bit rates above 128 kbps per channel. Layer 2 has an intermediate complexity and targets bit rates around 128 kbps per channel. Layer 3 is the most complex one, but offers the best audio quality, especially for bit rates around 64 kbps per channel. MP3 is the nickname for MPEG-1 layer 3.

MP3 has several qualities that make it flexible and suitable for different application scenarios. First of all, MP3 has four operating modes: single channel (mono), dual channel (two independent channels, for example containing different language versions of the audio), stereo and joint stereo (which is used for more efficient combined coding of the left and right channels of a stereophonic audio signal). Second, MP3 supports compression of different sampling frequencies. MPEG-1 defines audio compression at 32 kHz, 44.1 kHz and 48 kHz. MPEG-2 extends this to half rates, i.e., 16 kHz, 22.05 kHz and 24 kHz. This coding standard has found its way into many different applications, such as digital audio broadcasting (on the internet and in wireless networks), internet streaming (which is discussed in this work), portable audio and for storage and exchange of music files on computers.

This appendix, along with Chapter 3, gives a detailed description of the MP3 coding process and implementation, based on existing literature [42, 27] and on an MP3 source-code example that is available in the net (under the name 'dist10.tgz'). In addition, this appendix gives a background to the theory of psychoacoustics, and describes psychoacoustic model 2, which is the one used in the MP3 coder. Finally, there is also reference to more advanced versions of MPEG-Audio coders.

A.1 MP3 Encoder

The general description of the MP3 encoder is already given in Chapter 3 (see section 3.1.1). Chapter 3 also covers most of the subject of time-frequency mapping in the MP3 (section 3.2), therefore this appendix discusses only the implementation of the polyphase filter-bank. In addition, this section covers the rest of the encoder's parts: The psychoacoustic model, the quantization loop and the bit-stream formatting.

A.1.1 Polyphase Filter-Bank Implementation

As already described in section 3.2.1, the polyphase filter-bank divides the signal into 32 sub-bands of equal width. Each of the filters, $h_k(n)$, is 512-points long and is obtained by multiplying the impulse response of a single prototype low-pass filter, $h_0(n)$, by a modulation function which shifts the low-pass response to the frequency range of the appropriate sub-band, as described in (3.1).

The filtering procedure can be described by a simple convolution:

$$s_k[n] = \sum_{\tau=0}^{511} x[n-\tau] \cdot h_k[\tau] \qquad ,0 \leq k \leq 31 \tag{A.1}$$

Where k is the index of the sub-band. Such a direct implementation will require $32 \cdot 512 = 16384$ multiplies and $32 \cdot 511 = 16352$ additions. Naturally, we can do better than that...

Substituting the explicit expression for $h_k(n)$ we get:

$$s_k[n] = \sum_{\tau=0}^{511} x[n-\tau] \cdot h_0[\tau] \cdot \cos(\tfrac{\pi}{64} \cdot (\tau - 16) \cdot (2k+1)) \qquad ,0 \leq k \leq 31 \tag{A.2}$$

Examining the cosine expression it can be seen that for every k, it has a period of 128 samples, hence that every 64 samples the phase is reversed. This periodicity can be used in the calculations and therefore, only the first 64 values of the cosine function for each sub-band have to be stored. Using $\tau \triangleq 64j + i$ we can write:

$$s_k[n] = \tag{A.3}$$

$$\sum_{j=0}^{7} \sum_{i=0}^{63} x[n-(64j+i)] \cdot h_0[64j+i] \cdot (-1)^j \cdot \cos(\tfrac{\pi}{64} \cdot (i-16) \cdot (2k+1)) \quad ,0 \leq k \leq 31$$

Let's denote:

$$C[n] \triangleq h_0[n] \cdot (-1)^{\lfloor \frac{n}{64} \rfloor} \quad 0 \leq n \leq 511 \tag{A.4}$$

$$M[k][i] \triangleq \cos(\tfrac{\pi}{64} \cdot (i-16) \cdot (2k+1)) \quad ,0 \leq k \leq 31 \quad ,0 \leq i \leq 63$$

Where $M[k][i]$ is a real-valued matrix of dimensions 32×64 and $C[n]$ is a 512-point window function. Then (A.3) can be written as:

$$s_k[n] = \sum_{i=0}^{63} M[k][i] \cdot \sum_{j=0}^{7} x[n-(64j+i)] \cdot C[64j+i] \qquad ,0 \leq k \leq 31 \tag{A.5}$$

Implementing this way requires only $64 \cdot 8 + 32 \cdot 64 = 2560$ multiplies and $7 \cdot 64 + 32 \cdot 63 = 2464$ additions. This is also the place to note that the $M[k][i]$ coefficients are similar to the coefficients used by a 32-point, un-normalized inverse discrete cosine transform (IDCT) and by a simple manipulation of the data, this fact can be used to further optimize the calculations.

A.1.2 The Psychoacoustic Model

The quality of the psychoacoustic model has a great influence on the quality of the encoding process. A lot of additional work has gone into this part of in the encoder since the standard [18] was written. In the next section, some background on the human auditory system will be given and the psychoacoustic principles that can be used for efficient compression will be introduced. Later, the psychoacoustic model 2, that was implemented in ISO/IEC source-code example is explored in details.

Introduction to psychoacoustics

Since it was established, the field of psychoacoustics [43] has made significant progress towards characterizing human auditory perception and particularly the time-frequency analysis process in the inner ear.

Perceptual coders for high quality audio coding have been a research topic since the late 70's, with most activity occurring at late 80's, where J. D. Johnston [44] introduced the theory of *perceptual entropy* (PE), a quantitative estimate of the fundamental limit of transparent audio signal compression. Perceptual audio coders achieve compression by exploiting the fact that the signal contains some amount of irrelevant information that is not detectable even by a well trained or sensitive listener.

The first definition in this context is of the *sound pressure level* (SPL) which is a standard metric that gives the intensity level of sound pressure in decibels (dB) relative to an internationally defined reference level, i.e.:

$$L_{SPL} = 20log_{10}\left(\frac{p}{p_0}\right) \quad [dB] \tag{A.6}$$

Where p is the sound pressure of the acoustic stimulus (in units of Newtons per square meter) and p_0 is the standard reference level. The dynamic range of the intensity for the human auditory system is between 0 to 150 dB-SPL.

The *absolute threshold of hearing* characterizes the amount of energy needed for a pure tone to be detected by a listener in a noiseless environment (see Figure A.1). The SPL reference level is calibrated such that the frequency-dependent absolute threshold of hearing in quiet is measured to be around 0 dB-SPL. When applied to signal compression, the absolute threshold of hearing could be interpreted naively as a maximum allowable energy level for coding distortions, such as quantization noise, introduced in the frequency domain. Unfortunately, on its own, the absolute threshold is of limited value in the coding context: the thresholds are associated with pure tone stimuli, while the quantization noise tends to be spectrally complex rather than tonal. The detection threshold in this case is a

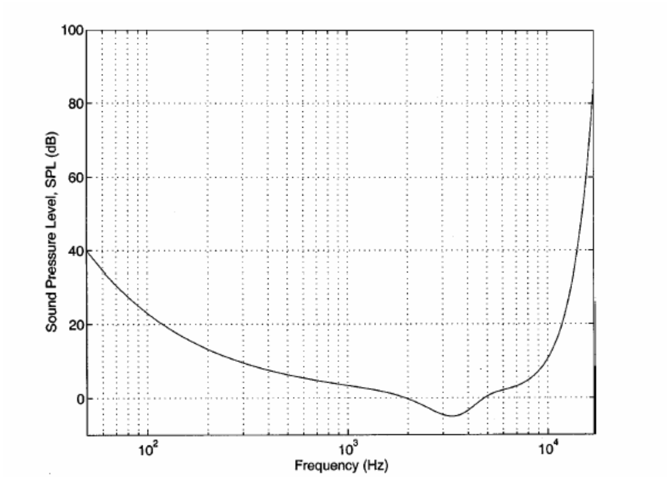


Figure A.1: The absolute threshold of hearing in quiet

modified version of the absolute threshold, also considering the fact that stimuli are in general time-varying, and therefore the detection threshold should also be a time-varying function of the input signal.

In order to estimate this threshold one must consider the way that the human ear performs spectral analysis: The inner ear is filled with fluid and is composed of two canals separated by a membrane, all of which is wound into the form of a spiral. The nerve-endings, scattered along the membrane separating the two canals, are receptors. Since the spiral membrane is long and narrow, we can refer to the position of a small group of nerve-endings as a single coordinate. Very tiny nerve fibers coming from the main auditory nerve are fanned out so as to reach each of these nerve endings. The arrangement of the nerve fibers is similar to the arrangement of wires in a telephone cable which are fanned out and connected to the telephone switch board. Each one of these tiny nerve-endings acts like a telephone transmitter. A sound wave that enters the human ear system is translated into mechanical vibrations that induce travelling waves along the length of the basilar membrane. The travelling waves generate peak responses at frequency specific membrane positions, so different nerve receptors react to different frequency bands according to their locations (see Figure A.2), hence a frequency-to-place transformation is taking place. Once they are triggered by the sound waves coming into the inner ear, the nerve-endings transmit an electrical current which goes to the brain and causes the sensation of hearing. From a signal-processing point

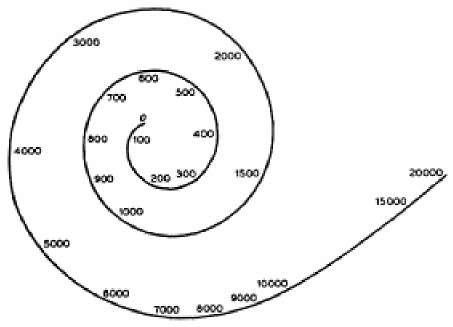


Figure A.2: Position of the response to different frequencies along the membrane according to maximum response to pure tones

of view, this frequency-to-place transformation can be viewed as a bank of highly overlapping band-pass filters, with asymmetric and nonlinear frequency response, and non-uniform bandwidth: the bandwidths of the different filters increase as a direct ratio of the center frequency of each filter. These frequency bands are usually referred to as *critical bands*. The different bands can be viewed in Table A.1. The critical bands are also responsible for the *masking phenomenon*, which is used by modern perceptual audio coders. Masking refers to a process where one sound is rendered inaudible because of the presence of another sound. Regarding the human hearing system, we refer to two kinds of masking: simultaneous and non-simultaneous masking.

Simultaneous masking may occur when two or more stimuli are simultaneously

Table A.1: Critical band filter bank

Band No.	Center Frequency [Hz]	Band Width [Hz]	Band No.	Center Frequency [Hz]	Band Width [Hz]
1	50	0-100	14	2150	2000-2320
2	150	100-200	15	2500	2320-2700
3	250	200-300	16	2900	2700-3150
4	350	300-400	17	3400	3150-3700
5	450	400-510	18	4000	3700-4400
6	570	510-630	19	4800	4400-5300
7	700	630-770	20	5800	5300-6400
8	840	770-920	21	7000	6400-7700
9	1000	920-1080	22	8500	7700-9500
10	1175	1080-1270	23	10500	9500-12000
11	1370	1270-1480	24	13500	12000-15500
12	1600	1480-1720	25	19500	15500-$\sim$20000
13	1850	1720-2000			

presented to the auditory system. From a frequency-domain perspective, the amplitude and shape of the spectrum of these two signals will determine who will be the masker and who the maskee, and to what extent. From a time-domain perspective, phase relationships between stimuli can also affect masking results. A simplified explanation to this phenomenon is that the presence of a strong masker signal creates an excitation on the basilar membrane at the critical band location, which is strong enough to block the detection of the weaker signal. For the purpose of audio coding it is convenient to define three types of simultaneous masking: *Noise-Masking-Tone* (NMT), *Tone-Masking-Noise* (TMN) and *Noise-Masking-Noise* (NMN). The two first ones are presented in Figure A.3.

Noise-Masking-Tone: A case where a narrow-band noise with bigger amplitude masks a weaker tone within the same critical band. The minimum difference between the intensity of the masker and maskee (in dB-SPL units) is called signal-to-mask ratio (SMR) and depends on the locations of the two sound components.

In this case, the minimal SMR is achieved when the frequency of the masked tone is close to the center frequency of the masking noise.

Tone-Masking-Noise: In this scenario, a pure tone with higher intensity occurring at the center of a critical band masks noise located within the same critical band. Again, the minimal SMR is achieved when the frequency of the masker tone is close to the center frequency of the masked noise.

Noise-Masking-Noise: Similarly, in this case one narrow-band noise is masking another narrow-band noise. The SMR here is harder to calculate because of the influence of phase relationships between the masker and maskee [45].

As can be seen in Figure A.3, TMN and NMT are a-symmetric: for tone masker and noise masker of the same amplitude level, the SMR values are different. In fact, knowledge of all three masking types is critical to successfully shape the coding distortion so it will be inaudible. For each analyzed segment in time, the perceptual model should identify, across the frequency spectrum, the noise-like and tone-like components in the audio signal so that the quantization could be held according to the appropriate masking relationships. Another property of the simultaneous masking is that it is not limited to the boundaries of a single critical band. Inter-band masking also occurs: i.e., a masker centered within one critical band has some predictable effect on the detection thresholds in adjacent critical bands. This effect is also known as the *spread of masking* and is often modelled by a triangular spreading function, as illustrated in Figure A.4.

Figure A.4 can also be viewed as an example case: in this case, there is a single masking tone located in the center of a critical band. The masking tone generates an excitation along the basilar membrane that is modelled by a spreading function and a corresponding masking threshold. For the band under consideration, the minimum masking threshold denotes the lowest level of the spreading function within the boundaries of the band. Assuming the masker is quantized using an m-bit uniform scalar quantizer (in the domain of dB-SPL) noise might be introduced at the level m (marked in a dotted line). The SMR (signal-to-mask ratio) and NMR (noise-to-mask ratio) denote the logarithmic distances from the minimum masking threshold to the masker and to the noise level, respectively.

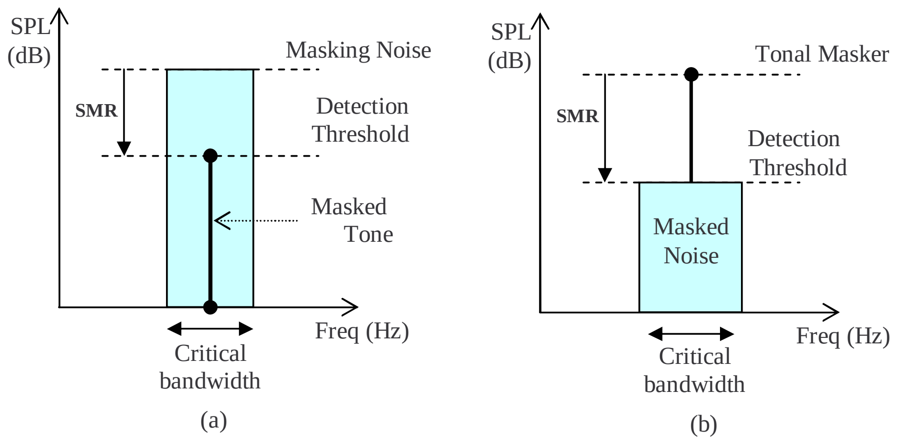


Figure A.3: Simultaneous masking: (a) Noise-Masking-Tone, (b) Tone-Masking-Noise.

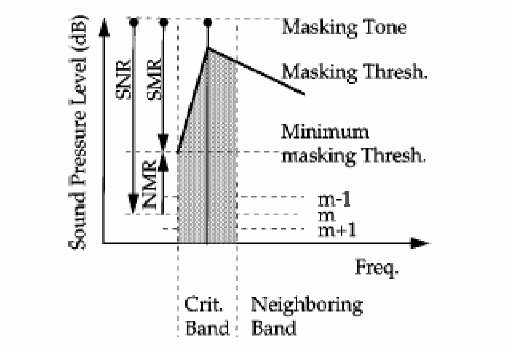


Figure A.4: A schematic representation of simultaneous masking and the spreading function

After the critical band analysis is finished and the spread of masking has been accounted for, the masking thresholds are established, usually in decibel relations:

$$\begin{aligned} &\text{TMN: } Thr = E_T - 14.5 - \text{Band's central bark value} \triangleq E_T - O_{TMN} \\ &\text{NMT: } Thr = E_N - \text{Constant value} \triangleq E_N - O_{NMT} \end{aligned} \tag{A.7}$$

where E_T and E_N are the critical band energies of the tone and noise masker respectively. Each time-frame usually contains a collection of both masking types, which are combined together to form a global masking threshold. The global masking threshold is used, along with the absolute threshold of hearing to create an estimate of the level at which quantization noise becomes "just noticeable".

The other type of masking is non-simultaneous masking, also denoted as *temporal masking* since for a masker of finite duration, this type of making occurs both prior to the masker's onset ("pre-masking") as well as after the masker decayed ("post-masking"), as shown in Figure A.5. Pre-masking tends to be very short (1-2 msec) while the duration of post-masking is longer, depending upon the strength, frequency and duration of the masker (50-300 msec). Temporal masking is used in several audio coding algorithms, including MPEG-1 and MPEG-2 Audio. In layer

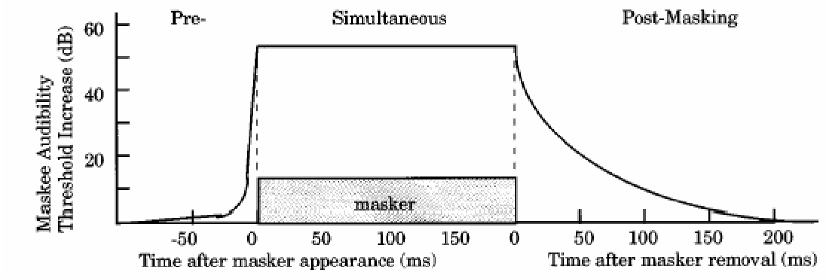


Figure A.5: Non-simultaneous masking

3, the psychoacoustic model also calculates the perceptual entropy (PE) for each analyzed time frame. The PE, defined in (A.18), is a measure of the perceptually relevant information that an audio record contains. It is expressed in bits per sample and represents the theoretical limit on the compressibility of a particular audio signal. Here, the PE is used for deciding the type of window function that will be used for each frame during the MDCT transform.

The psychoacoustic model of MP3

The MPEG-1 standard provides two example implementations of the psychoacoustic model. However, only model 2 includes specific modifications for layer 3 and therefore the standard recommends using it with MP3. A detailed description of model 1 can be found in [45]. Model 2 is described next:

Psychoacoustic model 2 contains a few main steps: spectral analysis, calculation of the unpredictability measure, switching to the perceptual domain and the calculation of the offset and then the masking threshold for each critical band. When used in layer 3 it contains another final stage of PE calculation and window decision.

Step 1 - spectral analysis

This step obtains a high-resolution spectral estimate of the input signal. The input signal is divided into segments of length $N = 1024$ samples each, with an overlap of 75% between consecutive segments (this division is fixed for all the sampling rates supported by MP3). The spectrum is divided in two ways: The first division is according to a long, 1024-point FFT that is performed on the segment after it was multiplied by a 1024-samples Hann window. The frequency domain is divided into 1024 equal width frequency lines, each has a bandwidth of $\frac{f_s}{1024}$. The second division is according to three short, 256-point FFT that are performed on three sub-frames, defined in the middle of the main segment, with an overlap of 50% between them (see Figure A.6), each multiplied by a 256-samples Hann window. In this division the time-resolution is improved at the expense of the frequency-resolution.

Step 2 - calculate the unpredictability measure for each frequency line

The unpredictability measure: upm_j, $0 \leq j \leq 511$, is basically the normalized prediction error, based on linear prediction of a single spectral coefficient from

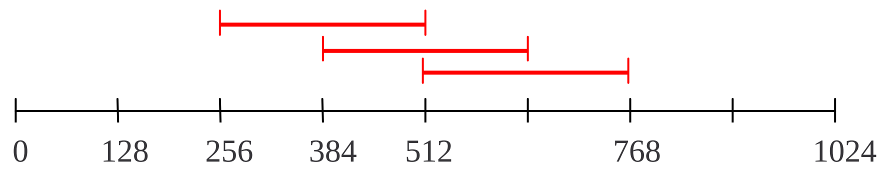


Figure A.6: Schematic description of the locations of the three short sub-frames

the same spectral coefficients in neighboring frames. Since tonal components are more predictable than noise components, their unpredictability measure should be smaller. The unpredictability measure is used for estimating the tonality index, which is later used for determining whether the frequency component is more tone-like or noise-like.

This measure is calculated for each frequency line separately, according to the following procedure (illustrated in Figure A.7):

- For the six lowest frequency lines (index 0-5), where the frequency resolution should be kept high, the long-FFT division is used. For each frequency line, its value is predicted based on the values in the previous two frames using linear prediction.

- In the next two-hundred frequency lines (index 6-205), the frequency resolution could be relaxed, so instead, the short-FFT division is used. The predictability measure for each line is calculated, by predicting the FFT value in the middle sub-frame based on the two neighboring sub-frames. Since each short-division frequency line covers 4 consecutive long-division frequency lines, the resulting value of unpredictability measure is assigned to the corresponding line quartet.

- The remaining high-frequency lines are ignored (assigned with a constant arbitrary value).

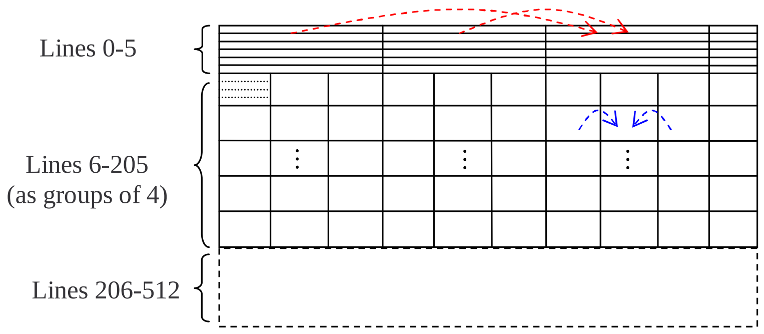


Figure A.7: Frequency division and prediction procedure for different frequency lines

Step 3 - process the spectral values into the perceptual domain

In order to simplify the calculations and to process the data in a perceptual domain, the long-FFT frequency lines are processed in groups.

In the Bark domain, a single bark unit represents a single critical band. The frequency location of each frequency line , $i\Delta f$, $0 \leq i \leq 511$ where $\Delta f = \frac{f_s}{1024}$, is mapped into the Bark domain, b_i ,$0 \leq i \leq 511$, and the frequency lines are then divided into 63 groups, each representing $\sim \frac{1}{3}$ Bark, or in other words - one third critical bandwidth.

The mapping of each frequency line to the bark domain is obtained by using the values in Table A.1: The integer part of b_i is the index of the critical band that contains the frequency $i\Delta f$, and the fractional part of b_i represents its location inside the critical band. for example, $f = 1020$Hz is translated into 9.625 Bark, since the frequency is in the range of critical band No. 9 (that has the bandwidth 920-1080 Hz), and $\frac{f-920}{1080-920} = 0.625$.

For each group, its power is calculated by summing the energy of all the lines in the group:

$$P_i = \sum_{j \in \Delta B_i} p_j \quad , 0 \leq i \leq 62 \tag{A.8}$$

And also the energy-weight unpredictability measure:

$$P_i^{UM} = \sum_{j \in \Delta B_i} upm_j \cdot p_j \quad , 0 \leq i \leq 62 \tag{A.9}$$

Where p_j is the absolute value of the FFT coefficient in frequency line j, upm_j is the unpredictability measure calculated for each line and ΔB_i is the set of frequency lines in the i'th group.

In order to complete the picture in the perceptual domain, the influence of the spreading function is also calculated. The spreading function, as was mentioned before, is used in order to estimate the effects of masking across different critical bands. Here, the value $s_{i,k}$ describes the effect of a tonal component centered in band i on another component in band k. the general shape of the spreading function is presented in Figure A.4. The energies and unpredictability of each group are convolved with the spreading function, in order to get the spread critical band spectrum:

$$Ps_i = \sum_{k=0}^{62} s_{i,k} \cdot P_k, \quad Ps_i^{UM} = \sum_{k=0}^{62} s_{i,k} \cdot P_k^{UM} \tag{A.10}$$

Step 4 - Calculate the offset for each group

For each group a tonality index $0 \leq \alpha_i \leq 1$ is calculated according to A.11:

$$\alpha_i = log_{10} \left(\frac{Ps_i}{2 \cdot Ps_i^{UM}} \right) \tag{A.11}$$

The resulting values are clipped to the limits of $[0,1]$.

The tonality index represents the nature of the frequency components of each band: $\alpha_i \longrightarrow 1$ indicates that the band contains tone-like components, and $\alpha_i \longrightarrow 0$ indicates that the components within that band are more noise-like. The model assumes that when the frequency component is tone-like the prediction error will be very small, resulting in a large value of α_i (which will be clipped to 1), on the other hand, trying to predict a noise-like component gives, in most cases, a prediction error of large scale, of the order of the component itself.

Next, we use the tonality index to calculate the threshold offset for each group, interpolating between TMN offset and NMT offset, according to the tone- or noise-like nature of each band:

$$O_i = O_i^{TMN} \cdot \alpha_i + O_i^{NMT} \cdot (1 - \alpha_i) \tag{A.12}$$

Where O_i^{TMN} and O_i^{NMT} are defined in (A.7).

Since the offset is given in units of dB-SPL, in the power domain it is:

$$P_i^{offset} = 10^{\frac{O_i}{10}} \tag{A.13}$$

Step 5 - calculate the masking threshold for each critical band

The spread critical band spectrum power is divided by the offset power to yield the spread threshold estimate:

$$Ts_i = \frac{Ps_i}{P_i^{offset}} \triangleq 10^{\left(log_{10}(Ps_i) - \frac{O_i}{10}\right)} \tag{A.14}$$

In order to convert the spread threshold back to the bark domain, the convolution of the spreading function must be undone. Since the de-convolution process is numerically unstable, re-normalization is used instead [44]. The spread threshold in each band is divided by the spread function gain:

$$T_i = \frac{Ts_i}{\sum_{j=0}^{62} s_{i,j}} \tag{A.15}$$

Next, and only for layer 3, there is a *pre-echo* control process: Pre-echo might occur when a signal with a sharp attack is introduced near the end of an analysis segment containing mostly regions of low energy. Even when the quantization is performed under the limitations of the masking thresholds, time-frequency uncertainty dictates that when the signal is inverted back to the time domain, the quantization distortion will spread evenly in time throughout the reconstructed block. At the decoder, this results in an unmasked distortion throughout the whole low-energy region preceding the attack. One way to handle this issue is by window switching (see section 3.2.2, on MDCT). Another precaution is to compare the masking threshold in each band to the thresholds that were calculated in preceding time frames, for the same band: if the masking threshold in the previous time-frame is much lower, even considering the incline due to the fast attack, it would be logical to reduce the threshold value for the current frame:

$$T_i'[n] = \min\{T_i[n], 2 \cdot T_i[n-1], 16 \cdot T_i[n-2]\} \tag{A.16}$$

Where n is the time index and i is the group's index.

The final stage is to compare the masking thresholds to the absolute threshold of hearing in quiet, in order to make sure that the thresholds don't introduce too severe demands, since the absolute threshold is considered as the lower bound on the audibility of sound. If the calculated noise threshold in a certain band, is lower than the absolute threshold, this value is replaced by the value of the absolute threshold in that band. At high and low frequencies, where the absolute threshold varies inside the group's band, the average value is used instead.

$$Thr_i = \max\{T_i', T_i^{ABS}\} \tag{A.17}$$

Layer 3 addition

The PE is calculated by measuring the actual number of quantizer levels, given a quantizer's step size that will result in noise energy equal to the audibility threshold. The quantization energy is assumed to spread evenly across all the frequency lines in a certain group.

$$PE = \sum_{j=0}^{62} \max\left\{0, log\left(\frac{1+P_j}{1+Thr_j}\right)\right\} \cdot |\Delta B_j| \tag{A.18}$$

If the PE value exceeds a certain predetermined threshold, then it is assumed that this frame contains an attack and a short window type is decided. The type of the previous time-frame is updated accordingly: for example, if it was a regular Long window, then it becomes a Start window. If the value doesn't exceed the predetermined threshold, one of the other three is used, according to the window type in the previous time-frame. Figure A.8 shows the state-machine of the window switching logic and the window functions are presented in section 3.2.2.

At this point, the bark domain is re-divided into wider non-overlapping bands, called *the scale-factor bands*. Each one of these bands might include one group or more, or a portion of it (among the 63 groups defined ealier in step 3). For each scale-factor band, the energy is re-calculated by a weighted-sum of the energies of the groups that are included (as whole or as part) in its territory. Same goes for the threshold re-calculation. In the case of a long window there are 21 scale-factor bands, and in the case of short window there are 12 scale-factor bands, for each of the three sub-windows (see the window's shape in Figure 3.6). Finally, the threshold-to-energy ratios are calculated for each of the scale-factor bands.

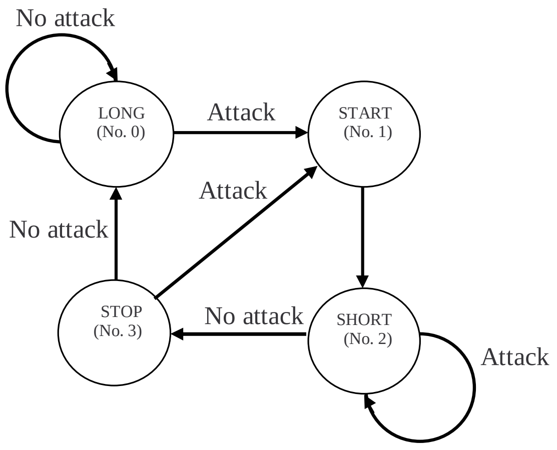


Figure A.8: The window switching state machine

In the case of MP3, the psychoacoustic model returns, for each encoded frame, the following parameters: the perceptual entropy, the window type and the scale-factor band ratios.

A.1.3 Quantization

The quantization and bit-allocation process determine the number of code bits allocated for each scale-factor band, based on the information from the psychoacoustic model. The MP3 standard uses an iterative procedure to determine the quantization step. The non-uniform quantization is done in the MDCT domain. The quantization loop is the most time consuming part of the MP3 encoding algorithm: It depends on the variation of audio signal and doesn't have a fixed execution time. The quantization process can be divided into three levels: The top level of the quantization process is the *iteration loop* subroutine. This subroutine calls the *outer loop* subroutine which controls the distortion level. This subroutine, on its turn, calls the *inner loop* subroutine which controls the bit-rate. The bit allocation procedure is illustrated in Figures A.9 - A.11, and is described next.

Iteration Loop

The iteration-loop has several responsibilities: the calculation of the allowed distortion for each scale-factor band, the calculation of the initial quantizer step-size, the determination of the scale-factor selection information (*scfsi*, in short) and the maintenance of the bit reservoir.

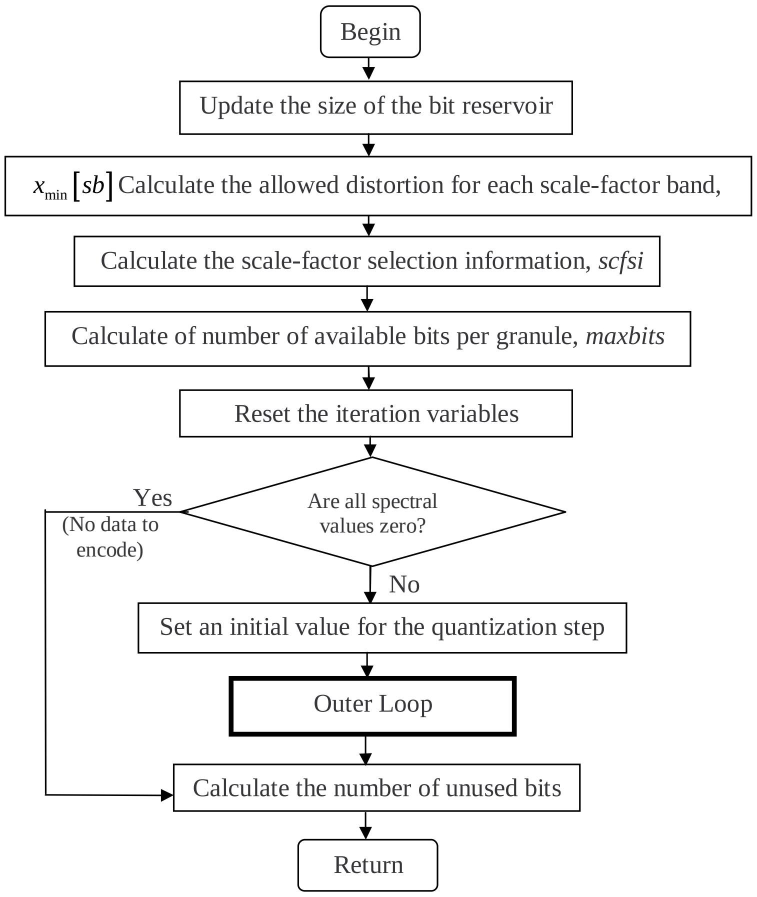

Begin
Update the size of the bit reservoir
$x_{min}[sb]$ Calculate the allowed distortion for each scale-factor band,
Calculate the scale-factor selection information, *scfsi*
Calculate of number of available bits per granule, *maxbits*
Reset the iteration variables
Are all spectral values zero?
Yes
(No data to encode)
No
Set an initial value for the quantization step
Outer Loop
Calculate the number of unused bits
Return

Figure A.9: A flow diagram describing iteration loop

- The calculation of the allowed distortion for each scale-factor band is done using:

$$x_{min}[sb] = \text{ratio}[sb] \cdot \frac{E_{sb}}{\Delta B_{sb}} \tag{A.19}$$

where sb is the index of the scale-factor band, ratio is the value that was calculated by the psychoacoustic model for that band, E_{sb} is the energy within the band and ΔB_{sb} is its bandwidth.

- The scale-factor selection information determines which scale-factor values of the first frame within a packet can be used for the second frame as well. These values are therefore not transmitted, and the spare bits can be used later in the coding process. The information is determined by applying some heuristic rules, regarding the level of energy and masking threshold in both frames.

- The *bit reservoir* mechanism controls the number of unused bits accumulated, since it is not possible for the encoder to always compress the data exactly to a certain number of bits. For example, a frame containing almost quiet signal will need fewer bits compared to a frame containing fast attack, or some other fast transient, so in that case there are a few unused bits left after the coding. The encoder keeps track of the remaining extra bits in case the next frames will need to use them, through the bit reservoir mechanism. However, since this mechanism introduces delay in the encoder, it is restricted: Only a certain number of bits may be accumulated, extra bits above the size limit will be discarded as stuffing bits.

  During the work of the outer-loop, the scale-factors, for each scale-factor band, are increased in small increments until the quantization noise is below x_{min} or until the scale factors cannot be increased any more. After the outer-loop finishes its job, the data is quantized and the remaining bits are added to the reservoir buffer.

Outer Loop - distortion control loop

The outer-loop controls the quantization noise produced by the quantization of the MDCT coefficients inside the inner-loop. The quantization noise is colorized by multiplying the MDCT lines, within each scale-factor band, by the actual scale-factors before the quantization takes place. When it is called, outer-loop calls

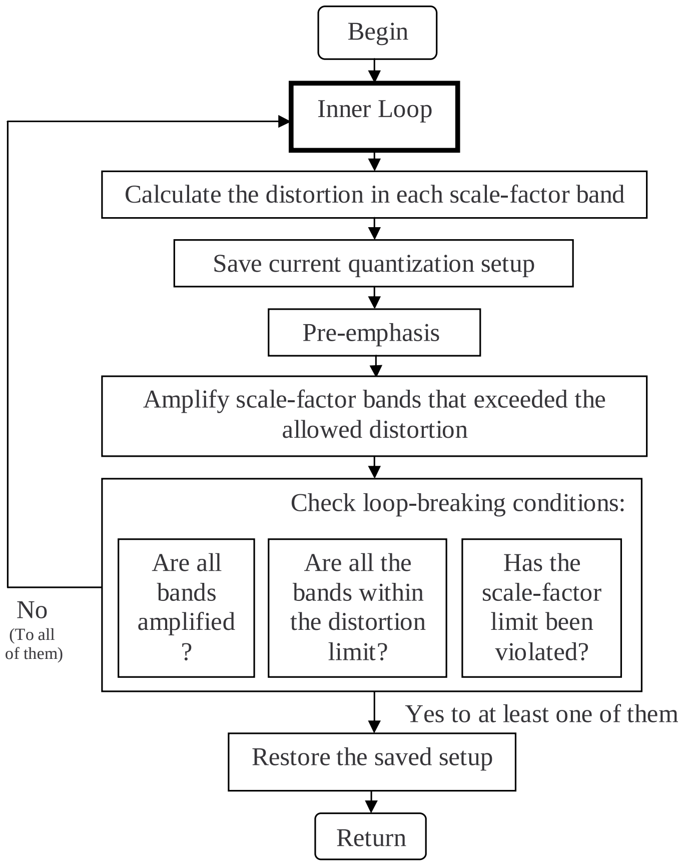

Begin
Inner Loop
Calculate the distortion in each scale-factor band
Save current quantization setup
Pre-emphasis
Amplify scale-factor bands that exceeded the allowed distortion
Check loop-breaking conditions:
Are all bands amplified ?
Are all the bands within the distortion limit?
Has the scale-factor limit been violated?
No (To all of them)
Yes to at least one of them
Restore the saved setup
Return

Figure A.10: A flow diagram describing outer loop

inner-loop in order to quantize the data according to the current parameters. After the quantization is completed, the amount of distortion in each scale-factor band is calculated by comparing the original values with the quantized ones. At this stage, the current quantization setup (parameters and results) is saved, and then

there is an attempt to check if it can further be improved:
First, the Pre-emphasis is switched on if the actual distortion in all the upper four scale-factor bands exceeds the threshold. The pre-emphasis option provides the possibility to amplify the upper part of the spectrum according to the pre-emphasis tables, defined in the standard. This operation enables an additional improvement in the resolution of the quantizer (besides the scale-factors) at higher frequencies.

Second, the scale-factors are amplified: The spectral values of the scale-factor bands which exceeds the allowed distortion are amplified by a constant factor (which can be 2 or $\sqrt{2}$, depending on the implementer's choice). The scale-factor value of each amplified band is increased by one. Section A.2 explains how these values are used in the decoder during the de-quantization process.

The loop will continue to a new iteration, unless one of three conditions occurs:

1. None of the scale-factor bands exceeds the allowed distortion level.
2. All scale-factor bands have already been amplified.
3. The amplification of at least one band exceeds the upper limit which is determined by the transmission format of the scale-factors (more details about that in section A.1.4).

If one of the above conditions is satisfied, the algorithm gives up and returns to the previously-stored quantization setup.

Inner Loop - rate control loop

The inner loop does the actual quantization of the spectral coefficients and calculates the number of bits required for encoding the data into bit-stream format.

The bit-stream formatting is done using entropy coding implemented by Huffman code tables.

The quantization of the vector of MDCT coefficients is done according to:

$$x_{quantized} = \text{sign}(x) \cdot \text{round}\left[\left(\frac{|x|}{2^{\frac{\Delta q}{4}}}\right)^{0.75} - 0.0946\right] \tag{A.20}$$

where Δq is the quantization step's exponent. The quantizer raises its input to the $\frac{3}{4}$ power before quantization to provide a more consistent signal-to-noise ratio over the range of quantizer values. The de-quantizer in the decoder inverses this operation by raising its output to the $\frac{4}{3}$ power.

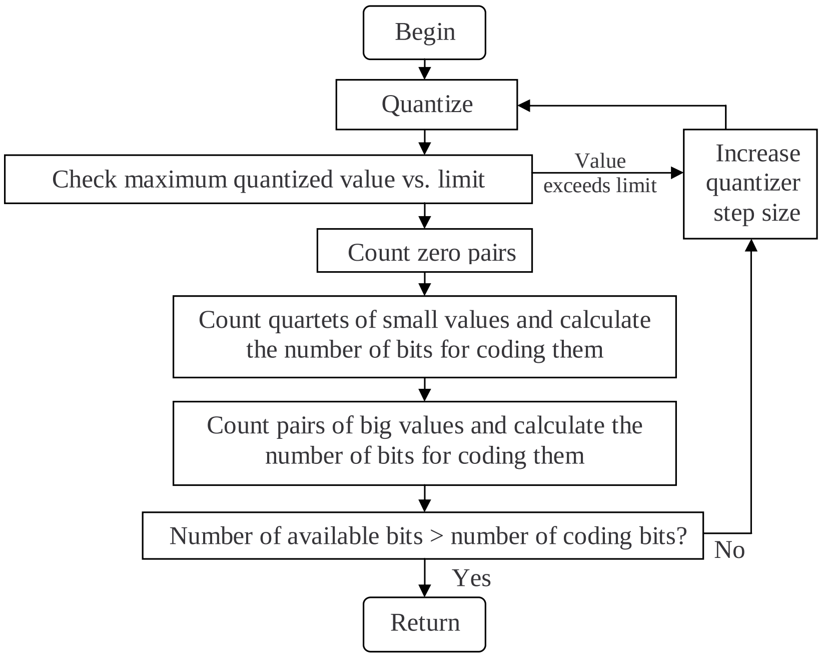

Begin
Quantize
Check maximum quantized value vs. limit
Value exceeds limit
Increase quantizer step size
Count zero pairs
Count quartets of small values and calculate the number of bits for coding them
Count pairs of big values and calculate the number of bits for coding them
Number of available bits > number of coding bits?
No
Yes
Return

Figure A.11: A flow diagram describing inner loop

Next, the number of bits required for the encoding is calculated: the quantized values are divided into three zones: zero values, small values with amplitude ≤ 1, and big values. The zone partition is illustrated in Figure A.12.

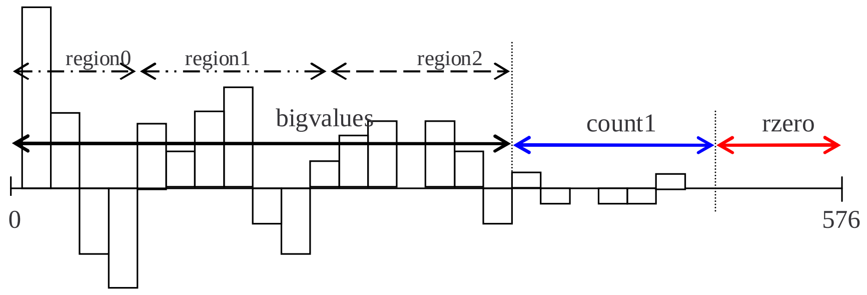


Figure A.12: The different encoding zones

Zero values: The *rzero* counter counts the number of consecutive zero-valued pairs, starting from the higher frequencies, since they usually contain less energy.

Small values: When reaching the first value which is not zero, the algorithm starts to count the number of consecutive quartets of quantized values with an amplitude equal or less than one (i.e., quantized to 0,1 or -1). The number of the quartets is stored in the counter *count1*. Each quartet is encoded by one Huffman codeword. There are two different Huffman code books with corresponding code length tables, and the one that gives the minimum number of bits is chosen.

Big values: The remaining values represent the area of values with higher amplitudes. The number of pairs in this area is counted by the *bigvalues* counter. The scale-factor bands in this area are further divided into three regions, where the split strategy is up to the implementer. Each region is coded separately and each pair within the region is encoded using a Huffman code table. There are 32 different Huffman code tables available, that differ from each other in the maximum value that can be coded with them and in the signal statistics they are optimized for. The Huffman tables can code values of up to 15. For higher values, the remaining portion is encoded using a linear

PCM codeword (each Huffman table has an option for a corresponding PCM field of limited length).

The total number of bits for the different zones is summed up. If it exceeds the number of available bits, than the quantizer step size is widened and a new iteration takes place. If it is less than the available bit number, the process ends and the subroutine returns to its caller.

A.1.4 Bit-Stream Formatting

The last block of the MP3 encoder is the one that turns the quantized data into MPEG-Audio Layer III compliant bit-stream. An MP3 packet includes data of two consecutive frames, each have 576 encoded MDCT coefficients. The general structure of an MP3 packet includes a header section, side information section and the main data section, as shown in Figure A.13. If error protection is specified, then the header block is immediately followed by a 16-bit CRC check word. Also, it is possible to add an optional ancillary data section, at the expense of the main data section.

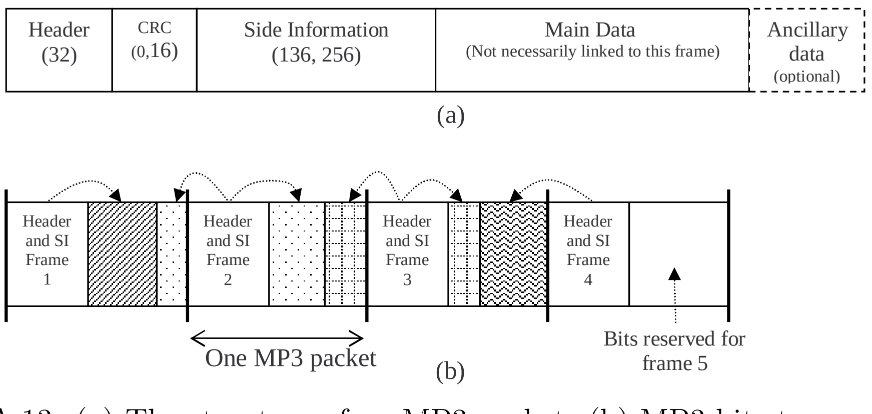

Header (32)
CRC (0,16)
Side Information (136, 256)
Main Data (Not necessarily linked to this frame)
Ancillary data (optional)
(a)
Header and SI Frame 1
Header and SI Frame 2
Header and SI Frame 3
Header and SI Frame 4
One MP3 packet
Bits reserved for frame 5
(b)

Figure A.13: (a) The structure of an MP3 packet. (b) MP3 bit-stream diagram

The purpose of the header section is to allow the receiver to establish synchronization and to determine the basic coding parameters anytime during a broadcast.

It contains key parameters, such as layer number (1-3), version number (MPEG-1/2), sampling frequency, operating mode (mono, stereo etc.) and the existence of error protection codes. The header's length is fixed to 32 bits, and its format is the same to all MPEG-1 layers.

The side information section includes information that is required for the de-quantization and synthesis process of each frame. Among the data coded in this section are: the chosen window type, the indices of the chosen Huffman tables, information about the quantization step and the scale-factors for each band. The length of the side information section is fixed too: 136 bits in case of mono signal and 256 in case of stereo.

The packet's main data section includes the scale-factors and the Huffman encoded MDCT values for each of the two frames (and for each channel, in case of stereo signal). As was already mentioned in section A.1.3, since it's not possible for the encoder to compress each frame exactly to the desired bit rate, there might be some leftover bits in this section. These extra bits can be used as additional space for ancillary data (if such exists), or - if the application allows for some delay in the encoder - for the use of the next frames that are going to be coded, since the MP3 standard allows for the main data section of a single packet to be spread over several of the previous packets, as shown in Figure A.13. If neither of the cases happen, the extra bits are simply stuffed with zeroes.

There are 21 scale-factor bands in the case of long window types (type 0, 1 or 3) and $12 \cdot 3$ scale-factor bands in case of a short window (type 2). Each scale-factor band has a corresponding scale-factor, which is encoded and stored in the "main data" section. The scale-factor bands are divided into two predefined regions, and each region is allocated with a fixed number of bits for each scale-factor values in it, denoted as *slen1* and *slen2*. There are 16 possibilities for the values of the slen-couple, defined in the standard. The scale-factor values are stored simply in their binary representation, hence the value of the maximum scale-factor in the first region is limited to $[0, 2^{slen1}]$ and the same goes for the second region. The outer-loop (see section A.1.3) should make sure that one of the 16 coding options matches the scale-factor values.

The 576 MDCT coefficients of each frame are also quantized and coded according to the Huffman tables that were selected during the work of the bit-allocation loop.

Tables A.2 and A.3 form a small "MP3-dictionary" that specifies the size and meaning of each of the parameters in the header and side information sections, respectively.

Table A.2: Specification of parameters and format of the header section

Parameter's name	Length (in bits)	Possible values	meaning
Sync word	12	0xfff	Indicates the beginning of new packet, used for synchronization.
Version	1	0-1	1 means MPEG-1, 0 means MPEG-2
Layer	2	1-3	The layer number: I, II or III
Error protection	1	0-1	Indicates the existence of a CRC field in the bit-stream.
Bit-rate index	4	0-14	Index of the coding bit-rate, chosen from 15 possible values. For MP3, indices 1-14 indicate varying bit-rates from 32 kbps to 320 kbps. '0' value indicates "free format", which enables using some other user-defined fixed bit-rate.
Sampling frequency	2	1-3	Index of the sampling frequency. For MPEG-1 layer 3 there are three possible values: 32 kHz, 44.1 kHz or 48 kHz.
Padding	1	0-1	This field is active only for signals with sampling rate of 44.1 kHz. Value of '1' indicates that this packet includes an extra slot, necessary for maintaining a fixed bit-rate, otherwise its value is '0'.
Extension	1	0-1	Currently not used.
Mode	2	0-3	Mode of operation: 0 - Stereo, 1 - Joint Stereo, 2 - Dual Channel, 3 - Mono
Mode extension	2	0-3	Active only for Joint Stereo mode. Specifies the coding method used for this special case.
copyright	1	0-1	1 - the file contains copyright protection, 0 - no such protection.
original	1	0-1	Some indication of the originality of the file.
emphasis	2	0,1,3	Indicates the type of de-emphasis that shall be used: 0 - no emphasis, 1 - 50/15 microsec. emphasis, 3 - CCITT J.17

Table A.3: Specification of parameters and format of the side-information section

Parameter's name	Length (in bits)	Possible values	meaning
main_data_begin	9	unsigned	Indicates the start position of the "main data" section of the current frame, as a negative offset from the frame's header.
private_bits	3 or 5		5 bits in case of mono, 3 bits else. For private use, not used by ISO.
Scfsi	4 for each channel	0-1 for each bit	Scale-factor selection information. The scale factor bands are divided into 4 groups, and a value is set for each group. 0 indicates that there are different scale factors for each frame, 1 indicates common values. In case of short window all the values are set to zero.
part2_3_length	12 for each channel and frame	unsigned	Indicates the number of bits in the "main data" section. This information helps to find the starting point of the "ancillary data" section, if such section exists.
big_values	9 bits, same as above	unsigned	As was mentioned before (section A.1.3), the values in the "big values "region are coded as pairs. This field indicates the number of pairs in the "big values" coding area.
global_gain	8 bits, same as above	unsigned	Information about the quantization step.
scalefac_compress	4 bits, same as above	unsigned	Information about the number of bits allocated for the coding of the scale factors.
window_switching _flag	1 bit, same as above	0-1	'1' indicates the use of a window type which is not the usual "long" type (either short, start or stop window types). Otherwise, the value is set to '0'.
The following 4 fields exist only for when "window switching flag" is equal to 1:			
block_type	2 bits, same as above	1-3	Assuming the window type is not 0 ("long"), this field specifies the type of window used in each frame.
mixed_block_flag	1 bit, same as above	0-1	When a short window is selected, there is a possibility to use it only in the higher sub-bands, where in the lower bands a long window is used: 0 - short window in all the spectrum, 1 - long window in lower bands and short in all the rest.
2 x table_select	$2 \cdot 5$ bits, same as above	0-31	This field specifies which Huffman table is used for each region, in case of not-long window type (see details two table-cells down).
3 x subblock_gain	$3 \cdot 3$ bits, same as above	unsigned	Indicates the gain-offset, from the global_gain, for each short-window sub-block.
The following 3 fields exist only for when "window switching flag" is equal to 0:			
3 x table_select	$3 \cdot 5$ bits, same as above	0-31	The three regions that form the "big values" area are coded each separately. This field specifies which Huffman table (among the 32 possible) is used for each region.
region0_count	4 bits, same as above	unsigned	This field specifies the size of the first region within the "big values" coding area.
region1_count	3 bits, same as above	unsigned	This field specifies the size of the last region within the "big values" coding area.
preflag	1 bit, same as above	0-1	Indicates whether the pre-emphasis is used or not: 1 - used, 0 - not used.
scalefac_scale	1 bit, same as above	0-1	Indicates whether the scale-factor amplification is used with base of 2 or $\sqrt{2}$
count1table_select	1 bit, same as above	0-1	This field specifies which Huffman table (among the two possible) is used for the coding of the quartets in the area of small values (values with amplitude).

A.2 MP3 Decoder

A general description of the MP3 decoder with a block diagram are given in section 3.1.2. Here we present a more detailed description of the decoding process:

Synchronization and decoding of the bit-stream:

First, the information from the header section is extracted, and the packet's dimensions are calculated according to it. Next, the side information data is read, and the start position of the main data section is identified according to it. The side information also includes the data necessary for deciphering the main data section. In the end, the scale-factors and the quantization values of the quantized MDCT coefficients are restored from the main data section. If the user added any ancillary data, it is recovered at this stage too.

Inverse quantization:

The MDCT values are reconstructed using the quantization parameters extracted from the side information section and from the main-data section.
For spectral line i, where $0 \leq i \leq 575$, the de-quantization process is done by:

$$\hat{X}^{MDCT}[i] = gain[i] \cdot sign(q[i]) \cdot abs(q[i])^{\frac{4}{3}} \tag{A.21}$$

where $q[i]$ is the quantized value in each line.

The gain value for each MDCT value, is calculated by:

$$gain[i] = 2^{g\_exp(i)} \tag{A.22}$$

Where in the case of a long window (type 0, 1 or 3), assuming that $b(i)$ is the scale-factor band that line i belongs to:

$$\begin{aligned} g\_exp(i) = \tfrac{1}{4} \cdot \Delta Q - \tfrac{1}{2} \cdot (1 + scalefac\_scale) \\ \cdot (scalefactor\,[b\,(i)] + (preflag \cdot preemphasis\,[b\,(i)])) \end{aligned} \tag{A.23}$$

And in the case of short window (type 2) it also considers the sub-block gain for each of the three sub-blocks of a short window type:

$$\begin{aligned} &g\_exp(i) = \\ &\tfrac{\Delta q}{4} - 2 \cdot subblock\_gain[sub\text{-}window(i)] - \tfrac{1}{2} \cdot (1 + scalefac\_scale) \\ &\cdot (scalefactor\,[sub\text{-}window\,(i)]\,[b\,(i)] + preflag \cdot preemphasis\,[b\,(i)]) \end{aligned} \tag{A.24}$$

The Δq parameter, which is the same as the parameter appearing in the quantization formula (A.20), is obtained from the *global_gain* field in the side information section (see table A.3). The parameters *scalefac_scale*, *preflag* and *subblock_gain* also appear as fields in this section. The *global_gain* and *subblock_gain* values affect all values within one time frame. *scalefac_scale* and *preflag* further adjust the gain within each scalefactor band. The pre-emphasis values of the short and long windows are taken from a table defined in the standard, and the scale-factors and quantization indices are taken from the main-data section.

Inverse MDCT and aliasing cancellation:
At this point two things take place: First the filter-bank aliasing reduction that was performed in the encoder (due to the overlapping in the bands of the filter-bank) is reversed, in order to ensure a correct reconstruction of the signal. Then, the 576 frequency lines are re-arranged into their original sub-band form, where there are 18 coefficients for each of the 32 equal-width sub-bands. Every 18 such coefficients are transformed back into the sub-band domain using the Inverse-MDCT transform, resulting in an aliased version of the values. The aliasing is then cancelled by the overlap-and-add procedure that was described in section 3.2.2, resulting in the reconstructed samples of the sub-band domain.

Filter-bank summation:
Finally, 32 samples, one from each sub-band, are applied to the synthesis polyphase filter bank and 32 consecutive audio samples are calculated. This procedure is done 18 times, until the whole frame is reconstructed.

A.3 Advanced MPEG Audio Standards

Since MP3 was defined, research on perceptual audio coding has progressed and codecs with better compression efficiency became available. This section gives a short look on the more advanced versions of MPEG standards.

MPEG-2 denotes the second phase of MPEG, which introduced a lot of new concepts in video coding. The original MPEG-2 Audio standard [46], finalized in 1994, includes only two extensions to MPEG-1 Audio: multi-channel coding and coding at half sampling frequencies: 16 kHz, 22.05 kHz and 24 kHz. Both extensions don't contain new coding algorithms over MPEG-1 Audio, and were backward compatible.

MPEG-2 AAC verification tests in early 1994 showed that using new coding algorithms and giving up on the backward compatibility to MPEG-1 promised a significant improvement in coding efficiency. This led to the definition of MPEG-2 Advanced Audio Coding (AAC) [47], which is considered as the successor of MPEG-1 Audio. The AAC coder has better capabilities: it is able to handle more channels than MP3, and it can handle higher sampling frequencies than MP3 (up to 96 kHz). The AAC coder uses the coding tools already present in MP3, but in a better way: the filter bank is a pure MDCT instead of a hybrid filter bank, the long windows are nearly twice as long as the ones in MP3 providing better frequency resolution, and the short windows are smaller than the ones in MP3, providing better transients handling and less pre-echo. AAC coder also presents some new tools, such as Temporal Noise Shaping [48] (TNS) which is a tool designed to control the location, in time, of the quantization noise by transmission of filtering coefficients, and a prediction tool, designed to enhance compressibility of stationary signals.

MPEG-4 this standard, on its different versions, intends to become the next major standard in the world of multimedia. Unlike MPEG-1 and MPEG-2, the emphasis in MPEG-4 is on new functionalities rather than better compression efficiency. MPEG-4 Audio facilitates a wide variety of applications which could range from intelligible speech to high quality multi-channel audio, and from natural sounds to synthesized sounds. In particular, it supports the highly efficient representation of audio objects consisting of general audio signal, speech signals, synthetic audio and synthesized speech. Coding of general audio ranging from very low bit-rates up to high quality is provided by transform coding techniques, covering a wide range of bit-rates and bandwidths. For higher bit-rates an improved version of the MPEG-2 AAC technology is used. This version is backward compatible to MPEG-2 AAC and includes two new modules: Long Term Prediction (LTP), which replaces the MPEG-2 AAC prediction tool, gives the same performance level with less computational power, and Perceptual Noise Substitution (PNS), which allows replacing coding of noise-like parts of the signal by some noise generated in the decoder side.

Although the focus is on new features, MPEG-4 does introduce two new items for improving audio coding efficiency even further: Bandwidth extension and Parametric coding for low bit-rate coding (HILN [16]).

More details about the various MPEG standards can be found in the MPEG official website [49]. More details about other perceptual coders can be found in [45].

Appendix B

Converting the MDCT to the DSTFT domain and vice versa

This appendix describes in details the mathematical development of the expressions presented in section 6.1.

B.1 Basic Terms and Assumptions

Before describing the mathematical process, there is a need to establish a few basic terms and assumptions that are used in this process:

1. Since this work is implemented using an MP3 coder, both conversion expressions refer to the four window types defined in the MP3 standard and rely on the connections between them (regarding shape and size). Therefore, it is not recommended to use these expressions if the MDCT transform uses other window families. However, a similar conversion for other window families, with similar relationships between them, can easily be created using the same techniques described here.

2. The conversion assumes that the location and size of the time-segments that the DFT is applied on, are the exact same ones that the MDCT uses. Each

such segment contains $2N$ samples and there is a 50% overlap between consecutive segments.

As already mentioned in section 6.1, the window function that is being used during the DSTFT is required to be symmetric, with length of $2N$ samples and its two parts should also complement each other to the value of 1. Since it is symmetric, we refer only to its first half, denoted as $w[n]$ where $0 \leq n \leq N-1$. The second condition can be written as: $w[n] + w[N-n-1] = 1$ for $0 \leq n \leq N-1$.

As described in section 3.2.2, all the MDCT window types have the same length in total: $2N$ samples, same as the segments they are designed to multiply. But, these four windows are built from 3 basic half-window units. Therefore, we prefer here also to refer to these units instead of to the whole window. The three units are denoted $h^{long}[n]$, $h^{short2long}[n]$ and $h^{short}[n]$ where the first two are $N = 18$ samples long and the last one is $N_s = 6$ samples long:

$$h^{long}[n] = \sin\left(\frac{\pi}{2N} \cdot \left(n + \frac{1}{2}\right)\right) \qquad ,0 \leq n \leq N-1 \tag{B.1}$$

$$h^{short}[n] = \sin\left(\frac{\pi}{2N_s} \cdot \left(n + \frac{1}{2}\right)\right) \qquad ,0 \leq n \leq N_s-1 \tag{B.2}$$

$$h^{short2long}[n] = \begin{cases} 0 & 0 \leq n \leq N_s - 1 \\ h^{short}[n - N_s] & N_s \leq n \leq 2N_s - 1 \\ 1 & 2N_s \leq n \leq N-1 \end{cases} \tag{B.3}$$

Comparing Figure 3.6 with Figure 3.8 it can be seen that:

$$\text{Long}[n] = \begin{cases} h^{long}[n] & 0 \leq n \leq N-1 \\ h^{long}[2N-n-1] & N \leq n \leq 2N-1 \end{cases} \tag{B.4}$$

$$\text{Start}[n] = \begin{cases} h^{long}[n] & 0 \leq n \leq N-1 \\ h^{short2long}[2N-n-1] & N \leq n \leq 2N-1 \end{cases} \tag{B.5}$$

$$\text{Stop}[n] = \begin{cases} h^{short2long}[n] & 0 \le n \le N-1 \\ h^{long}[2N-n-1] & N \le n \le 2N-1 \end{cases} \tag{B.6}$$

The Short window is actually built of 3 sub-windows, each of length $2N_s$, where $N_s = \frac{N}{3}$ is the length of the corresponding half-window unit. The locations of the sub-windows, in reference to the long window types, can be described by:

$$\text{Short}_1[n] = \begin{cases} 0 & 0 \le n \le N_s-1 \\ h^{short}[n-N_s] & N_s \le n \le 2N_s-1 \\ h^{short}[N-n-1] & 2N_s \le n \le N-1 \\ 0 & N \le n \le 2N-1 \end{cases} \tag{B.7}$$

$$\text{Short}_2[n] = \begin{cases} 0 & 0 \le n \le 2N_s-1 \\ h^{short}[n-2N_s] & 2N_s \le n \le N-1 \\ h^{short}[N+N_s-n-1] & N \le n \le N+N_s-1 \\ 0 & N+N_s \le n \le 2N-1 \end{cases} \tag{B.8}$$

$$\text{Short}_3[n] = \begin{cases} 0 & 0 \le n \le N-1 \\ h^{short}[n-N] & N \le n \le N+N_s-1 \\ h^{short}[N+2N_s-n-1] & N+N_s \le n \le N+2N_s-1 \\ 0 & N+2N_s \le n \le 2N-1 \end{cases} \tag{B.9}$$

In the next sections, consecutive time segments are marked by consecutive indices, $p \in \mathbb{Z}$, as is the case in the MDCT and DSTFT domains:

An MDCT frame is denoted $X_{(p)}^{MDCT}[k]$, where $0 \le k \le N-1$.

A DSTFT frame is denoted $X_{(p)}^{DSTFT}[m]$, where $0 \le m \le N$ (here only the first part is considered, since the coefficients are conjugate-symmetric).

B.2 Conversion from MDCT to DSTFT

Let's assume that we have a sequence of MDCT frames, each frame contains N real-valued MDCT coefficients that represent a segment of $2N$ time-samples. Our goal is to obtain the DSTFT representation ($2N$ conjugate-symmetric coefficients) of a single time-segment, indexed p, based on the data in the MDCT domain.

B.2.1 Using A Single Window Type

For simplicity we will first describe the procedure assuming all these MDCT frames use a single long window type. The more general case is introduced in the next sub-section. The corresponding half-window will be denoted here as $h[n]\ , 0 \leq n \leq N$.

By using an Inverse-MDCT transform on that frame we get $2N$ time-aliased samples:

$$\hat{x}_{(p)}[n] = \frac{2}{N} \sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k] \cdot \cos\left(\tfrac{\pi}{N} \cdot \left(n + \tfrac{N+1}{2}\right) \cdot \left(k + \tfrac{1}{2}\right)\right), \quad 0 \leq n \leq 2N-1 \quad \text{(B.10)}$$

In order to cancel the aliasing and get the original samples we have to use the OLA procedure, described in section 3.2.2: An inverse-MDCT is applied on each of the frame's two closest neighbors. Then, each of the resulting aliased segments is multiplied by its corresponding window function and the overlapping time segments are added together, to restore the original samples:

$$x_{(p)}[n] = \begin{cases} \hat{x}_{(p-1)}[n+N] \cdot h[N-n-1] + \hat{x}_{(p)}[n] \cdot h[n] & 0 \leq n \leq N-1 \\ \hat{x}_{(p)}[n] \cdot h[2N-n-1] + \hat{x}_{(p+1)}[n-N] \cdot h[n-N] & N \leq n \leq 2N-1 \end{cases} \quad \text{(B.11)}$$

Now that the original samples are restored, a DFT transform of length $2N$ is applied on them, not before the samples are multiplied by DSTFT window function (represented by it's first half: $w[n]$):

$$\begin{aligned} X_{(p)}^{DSTFT}[m] &= \sum_{n=0}^{N-1} x_{(p)}[n]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn} + \sum_{n=N}^{2N-1} x_{(p)}[n]\cdot w[2N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \\ &= \sum_{n=0}^{N-1} x_{(p)}[n]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn} \qquad\qquad ,0\le m\le N \\ &\quad + (-1)^m \sum_{n=0}^{N-1} x_{(p)}[n+N]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \qquad , \end{aligned} \tag{B.12}$$

where after changing the summation variable, one of the exponents could be reduced to $(-1)^m$ and could be taken out of the sum.

Now, by substituting (B.11) into (B.12):

$$\begin{aligned} X_{(p)}^{DSTFT}[m] &= \sum_{n=0}^{N-1} \left(\hat{x}_{(p-1)}[n+N]\cdot h[N-n-1] + \hat{x}_{(p)}[n]\cdot h[n]\right)\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn} \\ &\quad + (-1)^m \sum_{n=0}^{N-1} \left\{ \begin{array}{r} \left(\hat{x}_{(p)}[n+N]\cdot h[N-n-1] + \hat{x}_{(p+1)}[n]\cdot h[n]\right) \\ \cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array} \right\} \end{aligned} \tag{B.13}$$

Thus, by substituting (B.10) into (B.13) we obtain:

$$\begin{aligned}
X_{(p)}^{DSTFT}[m] &= \frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}\left\{\begin{array}{r} X_{(p-1)}^{MDCT}[k]\cdot\cos\left(\frac{\pi}{N}\cdot\left(n+N+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[N-n-1\right]\cdot w\left[n\right]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right\} \\
&+\frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}\left\{\begin{array}{r} X_{(p)}^{MDCT}[k]\cdot\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[n\right]\cdot w\left[n\right]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right\} \qquad \text{(B.14)} \\
&+(-1)^{m}\frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}\left\{\begin{array}{r} X_{(p)}^{MDCT}[k]\cdot\cos\left(\frac{\pi}{N}\cdot\left(n+N\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[N-n-1\right]\cdot w\left[N-n-1\right]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right\} \\
&+(-1)^{m}\frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}\left\{\begin{array}{r} X_{(p+1)}^{MDCT}[k]\cdot\cos\left(\frac{\pi}{N}\cdot\left(n\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[n\right]\cdot w\left[N-n-1\right]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right\}
\end{aligned}$$

Then, after changing the order of the summations:

$$\begin{aligned}
X_{(p)}^{DSTFT}[m] &= \sum_{k=0}^{N-1}X_{(p-1)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\begin{array}{l}\cos\left(\frac{\pi}{N}\cdot\left(n+N+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[N-n-1\right]\cdot w\left[n\right]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right] \\
&+\sum_{k=0}^{N-1}X_{(p)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\begin{array}{l}\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[n\right]\cdot w\left[n\right]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right] \qquad \text{(B.15)} \\
&+(-1)^{m}\sum_{k=0}^{N-1}X_{(p)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\begin{array}{l}\cos\left(\frac{\pi}{N}\cdot\left(n+N\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[N-n-1\right]\cdot w\left[N-n-1\right]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right] \\
&+(-1)^{m}\sum_{k=0}^{N-1}X_{(p+1)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\begin{array}{l}\cos\left(\frac{\pi}{N}\cdot\left(n\frac{N+1}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \\ \cdot h\left[n\right]\cdot w\left[N-n-1\right]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right]
\end{aligned}$$

Each of the expressions in the square brackets will be denoted as a function of two parameters: m, k. The expressions containing the direct form of $w[n]$ are marked with 'd' and those with the reversed form, $w[N-n-1]$, are marked with 'r'. Also, the expressions that use the direct form of $h[n]$ are marked by the number '1' and the ones that use the reverse form, $h[N-n-1]$, are marked by the number '2'.

In total, we have: g_d^1, g_r^1, g_d^2 and g_r^2.

We obtain:

$$\begin{aligned} X_{(p)}^{DSTFT}[m] &= \sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k] \cdot \left(g_d^1[m,k] + (-1)^m g_r^2[m,k]\right) \\ &\quad + \sum_{k=0}^{N-1} X_{(p-1)}^{MDCT}[k] \cdot g_d^2[m,k] + \sum_{k=0}^{N-1} X_{(p+1)}^{MDCT}[k] \cdot (-1)^m \cdot g_r^1[m,k] \end{aligned} \tag{B.16}$$

Where:

$$g_d^1[m,k] = \frac{2}{N} \sum_{n=0}^{N-1} \cos\left(\tfrac{\pi}{N}\left(n + \tfrac{N+1}{2}\right)\left(k + \tfrac{1}{2}\right)\right) \cdot h\,[n] \cdot w\,[n] \cdot e^{-j\frac{2\pi}{2N}mn} \tag{B.17}$$

$$g_r^1[m,k] = \frac{2}{N} \sum_{n=0}^{N-1} \cos\left(\tfrac{\pi}{N}\left(n\tfrac{N+1}{2}\right)\left(k + \tfrac{1}{2}\right)\right) \cdot h\,[n] \cdot w\,[N-n-1] \cdot e^{-j\frac{2\pi}{2N}mn} \tag{B.18}$$

$$g_d^2[m,k] = \frac{2}{N} \sum_{n=0}^{N-1} \left\{ \begin{array}{r} \cos\left(\tfrac{\pi}{N}\left(n + N + \tfrac{N+1}{2}\right)\left(k + \tfrac{1}{2}\right)\right) \cdot h\,[N-n-1] \\ \cdot w\,[n] \cdot e^{-j\frac{2\pi}{2N}mn} \end{array} \right\} \tag{B.19}$$

$$g_r^2[m,k] = \frac{2}{N} \sum_{n=0}^{N-1} \left\{ \begin{array}{r} \cos\left(\tfrac{\pi}{N} \cdot \left(n + N\tfrac{N+1}{2}\right) \cdot \left(k + \tfrac{1}{2}\right)\right) \cdot h\,[N-n-1] \\ \cdot w\,[N-n-1] \cdot e^{-j\frac{2\pi}{2N}mn} \end{array} \right\} \tag{B.20}$$

B.2.2 Using Different Window Types

The main difference in this case is that the type of the MDCT window, $h[n]$, depends on the frame, i.e., it is also a function of p: each MDCT frame can use a different window. However, considering the four window types defined by the MP3, we can observe that some of the cases are easier than others: For example, when converting a frame that uses a Long window there might be an overlap between the Long window and a Start window, or an overlap between a Stop window and the

Long window. In such a case, the overlapping sections in each window function are the direct and reverse form of the same window function, $h^{long}[n]$ (see (B.4)-(B.6)) and therefore coincide with the 'single window' example that was described before.

The cases of converting a frame that uses a Start or a Stop window, which means overlapping with a Short window are less trivial: For example, the case of an overlap between a Start and a Short window, is translated into an overlap between a Start and a Short$_1$ window, as defined in (B.7). A quick look at these windows shows that the overlapping sections in these windows are not symmetric, and therefore this case requires special handling. Same goes for the case of an overlap between a Short and a Stop window, which is translated into an overlap between a Short$_3$ window, as defined in (B.9) and a Stop window.

The most complicated case is converting a frame that uses a Short window type, since it can be followed by either a Stop window or by another Short window, and preceded by either a Start window or by another Short window. Each of the cases defines a different expression.

Another thing which is important to note regarding the short window type, is that in the short window case the MDCT coefficients are organized differently than in a long window, since these coefficients represent 3 short MDCT transforms. In the short window case, the functions also take into account the ordering of the coefficients. Fig. B.1 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).

The rest of this section explores the different cases, and presents the mathematical development of several chosen examples. The full picture, i.e., the assignments for each possible case, are given in Table 6.1.

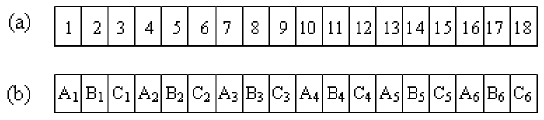


Figure B.1: (a) Long window (type 0,1,3) - coefficients arranged in ascending order: 1-18. (b) Short window (type 2) - interleaving the coefficients of the 3 short MDCT transforms: A-C, according to their ascending order: 1-6.

Converting a Long frame

As already mentioned, this is the easiest case which actually coincides with the particular case described in section B.2.1, since all the overlapping segments use h^{long}. By replacing all the instances of h in equations B.17-B.20 with h^{long} we obtain the functions $g^1_{d/r\_\text{long}}[m,k]$ and $g^2_{d/r\_\text{long}}[m,k]$ that are defined in section 6.1.1.

Converting a Start frame

Following the rules defined in the MP3 standard, in the case of a Start frame the preceding frame uses a Long window type, and the following frame uses a Short window type.

After applying an inverse-MDCT on the current frame and on its two closest neighbors we have the aliased samples. Since the current and previous frames (marked in index p and $p-1$, respectively) use a long MDCT transform, the expression for their inverse transform is given in (B.10). Regarding the next frame, which uses 3 short MDCT transforms, there are overlapping segments within the frame itself:

$$\begin{aligned}
\hat{x}_{(p+1)}[n] &= 0 && , 0 \leq n \leq N_s - 1 \qquad \text{(B.21)}\\
\hat{x}^{\text{Short}_1}_{(p+1)}[n] &= \frac{2}{N_s}\sum_{k=0}^{N_s-1} X^{MDCT}_{(p+1)}[3k]\cdot\cos\left(\tfrac{\pi}{N_s}\left(n - N_s + \tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\\
& && , N_s \leq n \leq N-1\\
\hat{x}^{\text{Short}_2}_{(p+1)}[n] &= \frac{2}{N_s}\sum_{k=0}^{N_s-1} X^{MDCT}_{(p+1)}[3k+1]\cdot\cos\left(\tfrac{\pi}{N_s}\left(n - 2N_s + \tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\\
& && , 2N_s \leq n \leq N+N_s-1\\
\hat{x}^{\text{Short}_3}_{(p+1)}[n] &= \frac{2}{N_s}\sum_{k=0}^{N_s-1} X^{MDCT}_{(p+1)}[3k+2]\cdot\cos\left(\tfrac{\pi}{N_s}\left(n + N + \tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\\
& && , N \leq n \leq N+2N_s-1\\
\hat{x}_{(p+1)}[n] &= 0 && , N+2N_s \leq n \leq 2N-1
\end{aligned}$$

The samples of the Start frame that we wish to convert, are restored using the first half of the Short frame following it, and the second half of the Long frame preceding it, in the following manner:

$$x_{(p)}[n] = \tag{B.22}$$

$$\begin{cases} \hat{x}_{(p-1)}[n+N]\cdot h^{long}[N-n-1] + \hat{x}_{(p)}[n]\cdot h^{long}[n] & ,0 \le n \le N-1 \\ \hat{x}_{(p)}[n]\cdot h^{short2long}[2N-n-1] & ,N \le n \le N+N_s-1 \\ \hat{x}_{(p)}[n]\cdot h^{short2long}[2N-n-1] & \\ \qquad +\hat{x}^{\text{Short}_1}_{(p+1)}[n-N]\cdot h^{short}[n-N-N_s] & ,N+N_s \le n \le N+2N_s-1 \\ \hat{x}_{(p)}[n]\cdot h^{short2long}[2N-n-1] & \\ \qquad +\hat{x}^{\text{Short}_1}_{(p+1)}[n-N]\cdot h^{short}[2N-n-1] & \\ \qquad +\hat{x}^{\text{Short}_2}_{(p+1)}[n-N]\cdot h^{short}[n-N-2N_s] & ,N+2N_s \le n \le 2N-1 \end{cases}$$

Applying the DSTFT on the restored samples gives the following expression:

$$X^{DSTFT}_{(p)}[m] \quad = \tag{B.23}$$

$$\sum_{n=0}^{N-1}(\hat{x}_{(p-1)}[n+N]\cdot h^{long}[N-n-1] + \hat{x}_{(p)}[n]\cdot h^{long}[n])\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+(-1)^m\sum_{n=0}^{N-1}\hat{x}_{(p)}[n+N]\cdot h^{short2long}[N-n-1]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+(-1)^m\sum_{n=N_s}^{2N_s-1}\hat{x}^{\text{Short}_1}_{(p+1)}[n]\cdot h^{short}[n-N_s]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+(-1)^m\sum_{n=2N_s}^{N-1}\left\{\begin{array}{r} (\hat{x}^{\text{Short}_1}_{(p+1)}[n]\cdot h^{short}[N-n-1] + \hat{x}^{\text{Short}_2}_{(p+1)}[n]\cdot h^{short}[n-2N_s]) \\ \cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right\}$$

By substituting the explicit expressions for the inverse-MDCT we obtain:

$$X_{(p)}^{DSTFT}[m] \quad = \tag{B.24}$$

$$\frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} X_{(p-1)}^{MDCT}[k]\cdot\cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{long}[N-n-1]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+\frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k]\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{long}[n]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+(-1)^m\frac{2}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}\left[\begin{array}{l} X_{(p)}^{MDCT}[k]\cdot\cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ \cdot h^{short2long}[N-n-1]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right]$$

$$+(-1)^m\frac{2}{N_s}\sum_{n=N_s}^{2N_s-1}\sum_{k=0}^{N_s-1}\left[\begin{array}{l} X_{(p+1)}^{MDCT}[3k]\cdot\cos\left(\tfrac{\pi}{N_s}\left(n-N_s+\tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ \cdot h^{short}[n-N_s]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right]$$

$$+(-1)^m\frac{2}{N_s}\sum_{n=2N_s}^{N-1}\sum_{k=0}^{N_s-1}\left[\begin{array}{l} X_{(p+1)}^{MDCT}[3k]\cdot\cos\left(\tfrac{\pi}{N_s}\left(n-N_s+\tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ \cdot h^{short}[N-n-1]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right]$$

$$+(-1)^m\frac{2}{N_s}\sum_{n=2N_s}^{N-1}\sum_{k=0}^{N_s-1}\left[\begin{array}{l} X_{(p+1)}^{MDCT}[3k+1]\cdot\cos\left(\tfrac{\pi}{N_s}\left(n-2N_s+\tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ \cdot h^{short}[n-2N_s]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array}\right]$$

The next steps are to change the order of the summations, and to change the variants in the short summations:

$$X_{(p)}^{DSTFT}[m] \quad = \tag{B.25}$$

$$\sum_{k=0}^{N-1} X_{(p-1)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{long}[N-n-1]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}\right]$$

$$+\sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{long}[n]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}\right]$$

$$+(-1)^m\sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k]\left[\frac{2}{N}\sum_{n=0}^{N-1}\left\{\begin{array}{l}\cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\\ \cdot h^{short2long}[N-n-1]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right\}\right]$$

$$+(-1)^m\sum_{k=0}^{N_s-1} X_{(p+1)}^{MDCT}[3k]\left[\begin{array}{l}\dfrac{2}{N_s}\displaystyle\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\cos\left(\tfrac{\pi}{N_s}\left(n+\tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{short}[n]\\ \cdot w[N-(n+N_s)-1]\cdot e^{-j\frac{2\pi}{2N}m(n+N_s)}\end{array}\right\}+\\ \dfrac{2}{N_s}\displaystyle\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\cos\left(\tfrac{\pi}{N_s}\left(n+N_s+\tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1]\\ \cdot w[N-(n+2N_s)-1]\cdot e^{-j\frac{2\pi}{2N}m(n+2N_s)}\end{array}\right\}\end{array}\right]$$

$$+(-1)^m\sum_{k=0}^{N_s-1} X_{(p+1)}^{MDCT}[3k+1]\left[\frac{2}{N_s}\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\cos\left(\tfrac{\pi}{N_s}\left(n+\tfrac{N_s+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\\ \cdot h^{short}[n]\cdot w[N-(n+2N_s)-1]\cdot e^{-j\frac{2\pi}{2N}m(n+2N_s)}\end{array}\right\}\right]$$

Again, as in the previous section, each of the summations in the square brackets will be denoted as a function, besides the short summations of the next frame, $X_{(p+1)}^{MDCT}$, which are gathered up to form a single function, defined for $0 \leq n \leq N-1$, and described by (B.26). The rest of the functions that are used in (B.27) are introduced in section 6.1.1.

$$g_{r\_\mathrm{short}}^{1}[m,k] =$$

$$\begin{cases} \displaystyle\sum_{n=0}^{N_s-1} \left[\begin{array}{l} \cos\left[\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(\lfloor\frac{k}{3}\rfloor+\frac{1}{2}\right)\right]\cdot h^{short}[n]\cdot w[N-1-(n+N_s)]\cdot e^{-j\frac{\pi}{N}(n+N_s)m} \\ +\cos\left[\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(\lfloor\frac{k}{3}\rfloor+\frac{1}{2}\right)\right]\cdot h^{short}[N_s-n-1] \\ \qquad\qquad\qquad\qquad \cdot w[N-1-(n+2N_s)]\cdot e^{-j\frac{\pi}{N}(n+2N_s)m} \end{array} \right] & k=0,3,6,\ldots,N-3 \\ \displaystyle\sum_{n=0}^{N_s-1}\cos\left[\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(\lfloor\frac{k}{3}\rfloor+\frac{1}{2}\right)\right]\cdot h^{short}[n]\cdot w[N-1-(n+2N_s)]\cdot e^{-j\frac{\pi}{N}(n+2N_s)m} & k=1,4,7,\ldots,N-2 \\ 0 & k=2,5,8,\ldots,N-1 \end{cases} \tag{B.26}$$

$$\begin{aligned} X_{(p)}^{DSTFT}[m] &= \sum_{k=0}^{N-1} X_{(p)}^{MDCT}[k]\cdot\left(g_{d\_\mathrm{long}}^{1}[m,k]+(-1)^m g_{r\_\mathrm{short2long}}^{2}[m,k]\right) \\ &+\sum_{k=0}^{N-1} X_{(p-1)}^{MDCT}[k]\cdot g_{d\_\mathrm{long}}^{2}[m,k] \\ &+\sum_{k=0}^{N-1} X_{(p+1)}^{MDCT}[k]\cdot(-1)^m\cdot g_{r\_\mathrm{short}}^{1}[m,k] \end{aligned} \tag{B.27}$$

Converting a Short frame

The most difficult task is to convert an MDCT frame that uses a Short window type. Besides for the number of inverse-MDCT transform that participate in this conversion (since each Short window means 3 short transforms), there is also the aspect of the types of the preceding and following frames: Since in the case of a Short window type, there are several possibilities. By the ordering defined by the

MP3 standard, a Short window type can follow either a Start or a Short window type. Also, the frame following a Short window type can use either a Short or a Stop window type. Hence, in this case there are 4 options for a sequence of 3 consecutive windows.

Since the case of a Start window overlapping a Short window was already covered in the previous subsection, we show here an example of the other two cases: a Short window overlapping a Short window, and a Short window overlapping a Stop window. Hence we chose to describe the development of the conversion procedure for a Short window, assuming it is preceded by a Short window and is followed by a Stop window type: Short→Short→Stop.

Following (B.21) which introduces the aliased samples in the case of a Short window, the original samples in this case are restored by:

$$x_{(p)}[n] = \tag{B.28}$$

$$\begin{cases} \hat{x}_{(p-1)}^{\text{Short}_2}[n+N]\cdot h^{short}[N_s-n-1]+\hat{x}_{(p-1)}^{\text{Short}_3}[n+N]\cdot h^{short}[n] & ,0\le n\le N_s-1 \\ \hat{x}_{(p-1)}^{\text{Short}_3}[n+N]\cdot h^{short}[2N_s-n-1]+\hat{x}_{(p)}^{\text{Short}_1}[n]\cdot h^{short}[n-N_s] & ,N_s\le n\le 2N_s-1 \\ \hat{x}_{(p)}^{\text{Short}_1}[n]\cdot h^{short}[N-n-1]+\hat{x}_{(p)}^{\text{Short}_2}[n]\cdot h^{short}[n-2N_s] & ,2N_s\le n\le N-1 \\ \hat{x}_{(p+1)}[n-N]\cdot h^{short2long}[n-N]+\hat{x}_{(p)}^{\text{Short}_3}[n]\cdot h^{short}[n-N] & \\ \qquad +\hat{x}_{(p)}^{\text{Short}_2}[n]\cdot h^{short}[N+N_s-n-1] & ,N\le n\le N+N_s-1 \\ \hat{x}_{(p+1)}[n-N]\cdot h^{short2long}[n-N] & \\ \qquad +\hat{x}_{(p)}^{\text{Short}_3}[n]\cdot h^{short}[N+2N_s-n-1] & ,N+N_s\le n\le N+2N_s-1 \\ \hat{x}_{(p+1)}[n-N]\cdot h^{short2long}[n-N] & ,N+2N_s\le n\le 2N-1 \end{cases}$$

Applying the DSTFT gives the following expression:

$$X_{(p)}^{DSTFT}[m] \quad = \tag{B.29}$$

$$\sum_{n=0}^{N_s-1}\Big(\hat{x}_{(p-1)}^{\text{Short}_2}[n+N]\cdot h^{short}[N_s-n-1]+\hat{x}_{(p-1)}^{\text{Short}_3}[n+N]\cdot h^{short}[n]\Big)\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+\sum_{n=N_s}^{2N_s-1}\Big(\hat{x}_{(p-1)}^{\text{Short}_3}[n+N]\cdot h^{short}[2N_s-n-1]+\hat{x}_{(p)}^{\text{Short}_1}[n]\cdot h^{short}[n-N_s]\Big)\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+\sum_{n=2N_s}^{N-1}\Big(\hat{x}_{(p)}^{\text{Short}_1}[n]\cdot h^{short}[N-n-1]+\hat{x}_{(p)}^{\text{Short}_2}[n]\cdot h^{short}[n-2N_s]\Big)\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+\sum_{n=N}^{N+N_s-1}\left\{\begin{array}{l}\Big(\hat{x}_{(p+1)}[n-N]\cdot h^{short2long}[n-N]+\hat{x}_{(p)}^{\text{Short}_3}[n]\cdot h^{short}[n-N]\\[2ex] +\hat{x}_{(p)}^{\text{Short}_2}[n]\cdot h^{short}[N+N_s-n-1]\Big)\cdot w[2N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right\}$$

$$+\sum_{n=N+N_s}^{N+2N_s-1}\left\{\begin{array}{l}\Big(\hat{x}_{(p+1)}[n-N]\cdot h^{short2long}[n-N]\\[2ex] +\hat{x}_{(p)}^{\text{Short}_3}[n]\cdot h^{short}[N+2N_s-n-1]\Big)\cdot w[2N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right\}$$

$$+\sum_{n=N+2N_s}^{2N-1}\hat{x}_{(p+1)}[n-N]\cdot h^{short2long}[n-N]\cdot w[2N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}$$

By substituting the explicit expressions for the inverse-MDCT, and changing the variants in some of the summations we obtain:

$$X_{(p)}^{DSTFT}[m] \; = \tag{B.30}$$

$$\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\left(\sum_{k=0}^{N_s-1} X_{(p-1)}^{MDCT}[3k+1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1]\right.\\ \left.+\sum_{k=0}^{N_s-1} X_{(p-1)}^{MDCT}[3k+2]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\right)\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right\}$$

$$+\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\left(\sum_{k=0}^{N_s-1} X_{(p-1)}^{MDCT}[3k+2]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1]\right.\\ \left.+\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\right)\\ \cdot w[n+N_s]\cdot e^{-j\frac{2\pi}{2N}m(n+N_s)}\end{array}\right\}$$

$$+\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\left(\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1]\right.\\ \left.+\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k+1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\right)\\ \cdot w[n+2N_s]\cdot e^{-j\frac{2\pi}{2N}m(n+2N_s)}\end{array}\right\}$$

$$+(-1)^m\sum_{n=0}^{N-1}\sum_{k=0}^{N-1} X_{(p+1)}^{MDCT}[k]\cdot\cos\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short2long}[n]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}$$

$$+(-1)^m\sum_{n=0}^{N_s-1}\left\{\begin{array}{l}\left(\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k+2]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\right.\\ \left.+\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k+1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1]\right)\\ \cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}\end{array}\right\}$$

$$+(-1)^m\sum_{n=0}^{N_s-1}\sum_{k=0}^{N_s-1}\left\{\begin{array}{l}X_{(p)}^{MDCT}[3k+2]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\\ \cdot h^{short}[N_s-n-1]\cdot w[N-(n+N_s)-1]\cdot e^{-j\frac{2\pi}{2N}m(n+N_s)}\end{array}\right\}$$

And after switching the order of the summations and re-arranging the arguments we have:

$$X_{(p)}^{DSTFT}[m] \;=\; \tag{B.31}$$

$$\sum_{k=0}^{N_s-1} X_{(p-1)}^{MDCT}[3k+1] \sum_{n=0}^{N_s-1} \left\{ \begin{array}{l} \cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1] \\ \cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array} \right\}$$

$$+\sum_{k=0}^{N_s-1} X_{(p-1)}^{MDCT}[3k+2] \sum_{n=0}^{N_s-1} \left\{ \begin{array}{l} \cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}mn} \\ +\cos\left(\frac{\pi}{N_s}\left(n+N_s\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1] \\ \cdot w[n+N_s]\cdot e^{-j\frac{2\pi}{2N}m(n+N_s)} \end{array} \right\}$$

$$+\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k] \sum_{n=0}^{N_s-1} \left\{ \begin{array}{l} \cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\cdot w[n+N_s]\cdot e^{-j\frac{2\pi}{2N}m(n+N_s)} \\ +\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[N_s-n-1] \\ \cdot w[n+2N_s]\cdot e^{-j\frac{2\pi}{2N}m(n+2N_s)} \end{array} \right\}$$

$$+\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k+1] \sum_{n=0}^{N_s-1} \left\{ \begin{array}{l} \cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\cdot w[n+2N_s] \\ \cdot e^{-j\frac{2\pi}{2N}m(n+2N_s)}+(-1)^m\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ \cdot h^{short}[N_s-n-1]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn} \end{array} \right\}$$

$$+(-1)^m\sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k+2] \sum_{n=0}^{N_s-1} \left\{ \begin{array}{l} \cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short}[n]\cdot w[N-n-1] \\ \cdot e^{-j\frac{2\pi}{2N}mn}+\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ \cdot h^{short}[N_s-n-1]\cdot w[N-(n+N_s)-1]\cdot e^{-j\frac{2\pi}{2N}m(n+N_s)} \end{array} \right\}$$

$$+(-1)^m\sum_{k=0}^{N-1} X_{(p+1)}^{MDCT}[k] \sum_{n=0}^{N-1} \cos\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot h^{short2long}[n]\cdot w[N-n-1]\cdot e^{-j\frac{2\pi}{2N}mn}$$

Again, as in the previous case, the short summations are gathered up to form a single function, defined for $0 \leq n \leq N-1$.

$$\begin{aligned} X_{(p)}^{DSTFT}[m] &= \sum_{k=0}^{N-1} X_{(p-1)}^{MDCT}[k] \cdot g_{d\_\text{short}}^{2}[m,k] \qquad \text{(B.32)} \\ &+ \sum_{k=0}^{N_s-1} X_{(p)}^{MDCT}[3k] \cdot \left(g_{d\_\text{short}}^{1}[m,k] + (-1)^m \cdot g_{r\_\text{short}}^{2}[m,k] \right) \\ &+ \sum_{k=0}^{N-1} X_{(p+1)}^{MDCT}[k] \cdot (-1)^m \cdot g_{r\_\text{short2long}}^{1}[m,k] \end{aligned}$$

Where $g_{r\_\text{short2long}}^{1}[m,k]$ is introduced in section 6.1.1 and $g_{d/r\_\text{short}}^{2}[m,k]$ and $g_{d\_\text{short}}^{1}[m,k]$ are introduced in (B.33)-(B.35).

$$g_{d\_\text{short}}^{2}[m,k] =$$

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

(B.33)

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

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

(B.34)

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

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

(B.35)

Converting a Stop frame

This conversion is done in the same manner as in the case of the Start frame, using the expressions that were already demonstrated in previous sub-sections.

B.3 Conversion from DSTFT to MDCT

Let's assume that we have a sequence of DSTFT frames, each frame contains $2N$ conjugate-symmetric coefficients that represent a segment of $2N$ time-samples. Our goal is to obtain the MDCT representation (N real-valued coefficients) of a single time-segment, indexed p, based on the data in the DSTFT domain.

As in the conversion from MDCT to DSTFT, the process here is also based on the simple procedure of using an inverse-DFT on the frame that we wish to convert and on its two closest neighbors in order to reconstruct the original samples using a simple overlap-and-add. Then, an MDCT transform is applied on the restored samples, not before they are multiplied with one of the four window types. The length of the MDCT transform is determined by the window type that was chosen (3 short transforms in a Short window type or one long transform for all other types).

Using some algebraic manipulations we obtain the general expressions and the functions that were introduced in section 6.1. Luckily, the functions that were defined for the direct conversion could be applied here as well.

The first step is to restore the $2N$ original samples using an inverse-DFT and an overlap-and-add that involves three consecutive DSTFT frames:

$$x_{(p)}[n] = \tag{B.36}$$

$$\begin{cases} \frac{1}{2N}\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}m(n+N)} + \frac{1}{2N}\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn} & ,0\le n\le N-1 \\ \frac{1}{2N}\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn} + \frac{1}{2N}\sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}m(n-N)} & ,N\le n\le 2N-1 \end{cases}$$

Then, the restored samples are multiplied by the corresponding MDCT window type and an MDCT transform is applied.

Converting into a Long frame

In the case of a Long window type, the window is expressed using two instances of the corresponding half-window unit, $h^{long}[\cdot]$, as shown in (B.4). The MDCT coefficients are therefore given by:

$$\begin{aligned} X_{(p)}^{MDCT}[k] &= \frac{2}{N}\sum_{n=0}^{N-1} x_{(p)}[n]\cdot h^{long}[n]\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ &+\frac{2}{N}\sum_{n=N}^{2N-1} x_{(p)}[n]\cdot h^{long}[2N-n-1]\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \end{aligned} \tag{B.37}$$

By substituting (B.36) into (B.37) and using a change of variants we have:

$$\begin{aligned} X_{(p)}^{MDCT}[k] &= \\ &\frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn}\cdot(-1)^m\cdot h^{long}[n]\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ &+\frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn}\cdot h^{long}[n]\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ &+\frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{2N-1}\left\{\begin{array}{l} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn}\cdot(-1)^m\cdot h^{long}[N-n-1] \\ \qquad\cdot\cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\end{array}\right\} \\ &+\frac{1}{N^2}\sum_{n=0}^{N-1}\sum_{m=0}^{2N-1}\left\{\begin{array}{l} X_{(p+1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn}\cdot h^{long}[N-n-1] \\ \qquad\cdot\cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\end{array}\right\} \end{aligned} \tag{B.38}$$

After changing the order of the summations:

$$X_{(p)}^{MDCT}[k] = \tag{B.39}$$

$$\frac{1}{N^2}\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot(-1)^m \sum_{n=0}^{N-1} \cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{long}[n]\cdot e^{j\frac{2\pi}{2N}mn}$$

$$+\frac{1}{N^2}\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m] \sum_{n=0}^{N-1} \cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)\cdot h^{long}[n]\cdot e^{j\frac{2\pi}{2N}mn}$$

$$+\frac{1}{N^2}\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot(-1)^m \sum_{n=0}^{N-1} \left\{\begin{array}{r} \cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ \cdot h^{long}[N-n-1]\cdot e^{j\frac{2\pi}{2N}mn} \end{array}\right\}$$

$$+\frac{1}{N^2}\sum_{m=0}^{2N-1}\sum_{n=0}^{N-1} X_{(p+1)}^{DSTFT}[m] \left\{\begin{array}{r} \cos\left(\tfrac{\pi}{N}\left(n+N+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \\ \cdot h^{long}[N-n-1]\cdot e^{j\frac{2\pi}{2N}mn} \end{array}\right\}$$

Checking the last equation we can see that the expressions multiplying the DSTFT coefficients are very similar to the functions introduced in section 6.1.1, with one difference: the function in section 6.1.1 are multiplied by the window function $w[\cdot]$. But, since we originally demanded that $w[n] + w[N-n-1] = 1$ than the desired expressions can be expressed by adding together the direct and reverse form of each function. Hence, the conversion expression can be downsized to:

$$
\begin{aligned}
X_{(p)}^{MDCT}[k] \;=\; & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (B.40)\\
& \frac{1}{N^2}\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot\Big[\big(g_{d\_\text{long}}^1[m,k]^* + g_{r\_\text{long}}^1[m,k]^*\big) + (-1)^m\big(g_{d\_\text{long}}^2[m,k]^* + g_{r\_\text{long}}^2[m,k]^*\big)\Big]\\
& + \frac{1}{N^2}\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot(-1)^m\cdot\big(g_{d\_\text{long}}^1[m,k]^* + g_{r\_\text{long}}^1[m,k]^*\big)\\
& + \frac{1}{N^2}\sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m]\cdot\big(g_{d\_\text{long}}^2[m,k]^* + g_{r\_\text{long}}^2[m,k]^*\big)
\end{aligned}
$$

Converting into a Start or a Stop frame

The conversion in this case is very similar to the case of the Long window type. Both the Start and the Stop windows are expressed using one instance of $h^{long}[\cdot]$ and one instance of $h^{short2long}[\cdot]$, as shown in (B.5)-(B.6). By substituting the corresponding half-window units we arrive to the same expressions: using $g_{d/r\_\text{long}}^1[m,k]$ with $g_{d/r\_\text{short2long}}^2[m,k]$ in the case of a Start window, and $g_{d/r\_\text{short2long}}^1[m,k]$ with $g_{d/r\_\text{long}}^2[m,k]$ in the case of a Stop window, as specified in Table 6.2.

Converting into a Short frame

In the case of Short window type, 3 short MDCT transforms are applied instead of one long transform: the samples are multiplied by each of the 3 sub-window functions, located as described in (B.7)-(B.9). Then, a short MDCT transform is applied on each of the resulting, $2N_s$-samples long, windowed segments. The

MDCT coefficients in this case are given by:

$$X_{(p)}^{MDCT}[k] = \tag{B.41}$$

$$\begin{cases} \frac{2}{N_s}\sum_{n=0}^{N_s-1} x_{(p)}[n+N_s]\cdot h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ +\frac{2}{N_s}\sum_{n=0}^{N_s-1} x_{(p)}[n+2N_s]\cdot h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ \qquad ,k=0,3,6,\ldots,N_s-3 \\ \frac{2}{N_s}\sum_{n=0}^{N_s-1} x_{(p)}[n+2N_s]\cdot h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ +\frac{2}{N_s}\sum_{n=0}^{N_s-1} x_{(p)}[n+N]\cdot h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ \qquad ,k=1,4,7,\ldots,N_s-2 \\ \frac{2}{N_s}\sum_{n=0}^{N_s-1} x_{(p)}[n+N]\cdot h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ +\frac{2}{N_s}\sum_{n=0}^{N_s-1} x_{(p)}[n+N+N_s]\cdot h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\ \qquad ,k=2,5,8,\ldots,N_s-1 \end{cases}$$

By substituting (B.36) into (B.41) we have:

$$X_{(p)}^{MDCT}[k] = \tag{B.42}$$

$$\begin{cases}
\frac{1}{N\cdot N_s}\sum_{n=0}^{N_s-1}\left(\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot(-1)^m\cdot e^{j\frac{2\pi}{2N}m(n+N_s)} + \sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}m(n+N_s)}\right) \\
\qquad \cdot h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
+\frac{1}{N\cdot N_s}\sum_{n=0}^{N_s-1}\left(\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot(-1)^m\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)} + \sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)}\right) \\
\qquad \cdot h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
\qquad , k = 0, 3, 6, \ldots, N_s-3 \\
\\
\frac{1}{N\cdot N_s}\sum_{n=0}^{N_s-1}\left(\sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m]\cdot(-1)^m\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)} + \sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)}\right) \\
\qquad \cdot h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
+\frac{1}{N\cdot N_s}\sum_{n=0}^{N_s-1}\left(\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot(-1)^m\cdot e^{j\frac{2\pi}{2N}mn} + \sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn}\right) \\
\qquad \cdot h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
\qquad , k = 1, 4, 7, \ldots, N_s-2 \\
\\
\frac{1}{N\cdot N_s}\sum_{n=0}^{N_s-1}\left(\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot(-1)^m\cdot e^{j\frac{2\pi}{2N}mn} + \sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}mn}\right) \\
\qquad \cdot h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
+\frac{1}{N\cdot N_s}\sum_{n=0}^{N_s-1}\left(\sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m]\cdot(-1)^m\cdot e^{j\frac{2\pi}{2N}m(n+N_s)} + \sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m]\cdot e^{j\frac{2\pi}{2N}m(n+N_s)}\right) \\
\qquad \cdot h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
\qquad , k = 2, 5, 8, \ldots, N_s-1
\end{cases}$$

By changing the order of the summations:

$$X^{MDCT}_{(p)}[k] = \tag{B.43}$$

$$\begin{cases}
\begin{aligned}
&\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p-1)}[m]\cdot(-1)^m \left\{\begin{array}{l}
\displaystyle\sum_{n=0}^{N_s-1} h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}m(n+N_s)} \\
\displaystyle+\sum_{n=0}^{N_s-1} h^{short}[N_s-n-1] \\
\qquad\qquad \cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)}
\end{array}\right\} \\
&+\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p)}[m] \left\{\begin{array}{l}
\displaystyle\sum_{n=0}^{N_s-1} h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}m(n+N_s)} \\
\displaystyle+\sum_{n=0}^{N_s-1} h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
\qquad\qquad\qquad\qquad \cdot e^{j\frac{2\pi}{2N}m(n+2N_s)}
\end{array}\right\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad , k=0,3,6,\ldots,N_s-3 \\[2ex]
&\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p-1)}[m]\cdot(-1)^m \sum_{n=0}^{N_s-1} h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)} \\
&+\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p)}[m] \left\{\begin{array}{l}
\displaystyle\sum_{n=0}^{N_s-1} h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}m(n+2N_s)} \\
\displaystyle+(-1)^m\sum_{n=0}^{N_s-1} h^{short}[N_s-n-1] \\
\qquad\qquad \cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}mn}
\end{array}\right\} \\
&+\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p+1)}[m]\sum_{n=0}^{N_s-1} h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}mn} \\
&\qquad\qquad\qquad\qquad\qquad\qquad , k=1,4,7,\ldots,N_s-2 \\[2ex]
&\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p)}[m]\cdot(-1)^m \left\{\begin{array}{l}
\displaystyle\sum_{n=0}^{N_s-1} h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}mn} \\
\displaystyle+\sum_{n=0}^{N_s-1} h^{short}[N_s-n-1] \\
\qquad\qquad \cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}m(n+N_s)}
\end{array}\right\} \\
&+\frac{1}{N\cdot N_s}\sum_{m=0}^{2N-1} X^{DSTFT}_{(p+1)}[m] \left\{\begin{array}{l}
\displaystyle\sum_{n=0}^{N_s-1} h^{short}[n]\cdot\cos\left(\frac{\pi}{N_s}\left(n+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\cdot e^{j\frac{2\pi}{2N}mn} \\
\displaystyle+\sum_{n=0}^{N_s-1} h^{short}[N_s-n-1]\cdot\cos\left(\frac{\pi}{N_s}\left(n+N_s+\frac{N_s+1}{2}\right)\left(k+\frac{1}{2}\right)\right) \\
\qquad\qquad\qquad\qquad \cdot e^{j\frac{2\pi}{2N}m(n+N_s)}
\end{array}\right\} \\
&\qquad\qquad\qquad\qquad\qquad\qquad , k=2,5,8,\ldots,N_s-1
\end{aligned}
\end{cases}$$

As in the previous cases, the expressions here can also be represented by the functions introduced in (B.26) and (B.33)-(B.35), in the following way:

$$\begin{aligned}
X_{(p)}^{MDCT}[k] \quad = \quad & \frac{1}{N \cdot N_s} \sum_{m=0}^{2N-1} X_{(p-1)}^{DSTFT}[m] \cdot (-1)^m \left(g_d^1\text{\_short}[m,k]^* + g_r^1\text{\_short}[m,k]^* \right) \qquad \text{(B.44)} \\
& + \frac{1}{N \cdot N_s} \sum_{m=0}^{2N-1} X_{(p)}^{DSTFT}[m] \left\{ \begin{array}{l} \left(g_d^1\text{\_short}[m,k]^* + g_r^1\text{\_short}[m,k]^* \right) \\ \\ +(-1)^m \left(g_d^2\text{\_short}[m,k]^* + g_r^2\text{\_short}[m,k]^* \right) \end{array} \right\} \\
& + \frac{1}{N \cdot N_s} \sum_{m=0}^{2N-1} X_{(p+1)}^{DSTFT}[m] \left(g_d^2\text{\_short}[m,k]^* + g_r^2\text{\_short}[m,k]^* \right)
\end{aligned}$$

Appendix C

Analysis of pure sinusoids in the MDCT and DSTFT domains

This Appendix presents the mathematical development of the expressions presented in section 4.2.1.
We start from a single sinusoidal signal:

$$x[n] = A \cdot \cos\left(\tfrac{2\pi}{2N}\left(K_0 + \Delta_0\right) n + \phi_0\right) \quad , n \in \mathbb{Z} \tag{C.1}$$

Where K_0 is an integer number: $0 \leq K_0 < 2N$, Δ_0 is a fractional number: $0 \leq \Delta_0 < 1$ and ϕ_0 is the initial phase.

As described in section 4.2.1, we assume that both the MDCT transform and the DSTFT transform are applied on 50% overlapping segments, each $2N$ samples-long. The segments are indexed by $p \in \mathbb{Z}$, hence, the p'th segment will be denoted as:

$$x_{(p)}[n] \triangleq \{x[pN + n]\}_{n=0}^{2N-1} \tag{C.2}$$

C.1 Applying the MDCT Transform

The MDCT transform is applied with a window function:

$$h[n] = \sin\left(\tfrac{\pi}{2N}\left(n + \tfrac{1}{2}\right)\right) \quad , 0 \leq n \leq 2N - 1 \tag{C.3}$$

Applying the MDCT transform on the signal we have:

$$\begin{aligned} X_{(p)}^{MDCT}[k] &= \sum_{n=0}^{2N-1} x[pN+n]\cdot h[n]\cdot \cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right) \quad ,0\le k\le N-1 \\ &= A\sum_{n=0}^{2N-1}\left\{\begin{array}{l} \cos\left(\frac{2\pi}{2N}(K_0+\Delta_0)(pN+n)+\phi_0\right) \\ \cdot\sin\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)\cdot\cos\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)\end{array}\right\} \end{aligned} \tag{C.4}$$

C.1.1 The case of $\Delta_0 = 0$

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} = \tag{C.5}$$

$$A\sum_{n=0}^{2N-1}\cos\left(\tfrac{2\pi}{2N}K_0(pN+n)+\phi_0\right)\cdot\sin\left(\tfrac{\pi}{2N}\left(n+\tfrac{1}{2}\right)\right)\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)$$

Next, the sine and cosine functions are replaced by the explicit exponent expressions:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} = \tag{C.6}$$

$$\frac{A}{8j}\sum_{n=0}^{2N-1}\left[e^{j\left(\frac{\pi}{N}K_0(pN+n)+\phi_0\right)}+e^{-j\left(\frac{\pi}{N}K_0(pN+n)+\phi_0\right)}\right]\cdot\left[e^{j\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)}-e^{-j\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)}\right]$$

$$\cdot\left[e^{j\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)}+e^{-j\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)}\right]$$

By opening the brackets inside the exponent expressions we have:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} = \tag{C.7}$$

$$\frac{A}{8j}\sum_{n=0}^{2N-1}\left[e^{j\left(\pi K_0 p+\frac{\pi}{N}K_0 n+\phi_0\right)}+e^{-j\left(\pi K_0 p+\frac{\pi}{N}K_0 n+\phi_0\right)}\right]\cdot\left[e^{j\left(\frac{\pi n}{2N}+\frac{\pi}{4N}\right)}-e^{-j\left(\frac{\pi n}{2N}+\frac{\pi}{4N}\right)}\right]$$

$$\cdot\left[e^{j\left(\frac{\pi}{N}nk+\frac{\pi n}{2N}+\frac{\pi(N+1)k}{2N}+\frac{\pi(N+1)}{4N}\right)}+e^{-j\left(\frac{\pi}{N}nk+\frac{\pi n}{2N}+\frac{\pi(N+1)k}{2N}+\frac{\pi(N+1)}{4N}\right)}\right]$$

Since K_0 is an integer number, then $e^{j\pi K_0 p} = e^{-j\pi K_0 p} = (-1)^{pK_0}$. Opening some more brackets:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} \;=\; \tag{C.8}$$

$$(-1)^{pK_0}\,\frac{A}{8j}\sum_{n=0}^{2N-1}\left[e^{j\left(\frac{\pi n}{2N}+\frac{\pi}{4N}\right)} - e^{-j\left(\frac{\pi n}{2N}+\frac{\pi}{4N}\right)}\right]\cdot\left[\begin{array}{l} e^{j\left(\frac{\pi}{N}(K_0+k)n+\frac{\pi n}{2N}+\frac{\pi(N+1)k}{2N}+\frac{\pi(N+1)}{4N}+\phi_0\right)} \\ +e^{j\left(\frac{\pi}{N}(K_0-k)n-\frac{\pi n}{2N}-\frac{\pi(N+1)k}{2N}-\frac{\pi(N+1)}{4N}+\phi_0\right)} \\ +e^{-j\left(\frac{\pi}{N}n(K_0-k)-\frac{\pi n}{2N}-\frac{\pi(N+1)k}{2N}-\frac{\pi(N+1)}{4N}+\phi_0\right)} \\ +e^{-j\left(\frac{\pi}{N}(K_0+k)n+\frac{\pi n}{2N}+\frac{\pi(N+1)k}{2N}+\frac{\pi(N+1)}{4N}+\phi_0\right)} \end{array}\right]$$

Opening more brackets:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} \;=\; \tag{C.9}$$

$$(-1)^{pK_0}\,\frac{A}{8j}\sum_{n=0}^{2N-1}\left[\begin{array}{l} e^{j\left(\frac{\pi}{N}(K_0+k+1)n+\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)} - e^{-j\left(\frac{\pi}{N}(K_0+k+1)n+\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)} \\ +e^{j\left(\frac{\pi}{N}(K_0-k)n-\frac{\pi(N+1)k}{2N}-\frac{\pi}{4}+\phi_0\right)} - e^{-j\left(\frac{\pi}{N}(K_0-k)n-\frac{\pi(N+1)k}{2N}-\frac{\pi}{4}+\phi_0\right)} \\ +e^{-j\left(\frac{\pi}{N}(K_0-k-1)n-\frac{\pi(N+1)k}{2N}-\frac{\pi(N+2)}{4N}+\phi_0\right)} - e^{j\left(\frac{\pi}{N}(K_0-k-1)n-\frac{\pi(N+1)k}{2N}-\frac{\pi(N+2)}{4N}+\phi_0\right)} \\ +e^{-j\left(\frac{\pi}{N}(K_0+k)n+\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)} - e^{j\left(\frac{\pi}{N}(K_0+k)n+\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)} \end{array}\right]$$

Taking the expressions that don't depend on n out of the summation, we get:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} \;=\; (-1)^{pK_0}\,\frac{A}{8j} \tag{C.10}$$

$$\cdot\left[\begin{array}{l} e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)}\sum_{n=0}^{2N-1} e^{j\frac{\pi}{N}(K_0+k+1)n} - e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)}\sum_{n=0}^{2N-1} e^{-j\frac{\pi}{N}(K_0+k+1)n} \\ +e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}-\phi_0\right)}\sum_{n=0}^{2N-1} e^{j\frac{\pi}{N}(K_0-k)n} - e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}-\phi_0\right)}\sum_{n=0}^{2N-1} e^{-j\frac{\pi}{N}(K_0-k)n} \\ +e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}-\phi_0\right)}\sum_{n=0}^{2N-1} e^{-j\frac{\pi}{N}(K_0-k-1)n} - e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}-\phi_0\right)}\sum_{n=0}^{2N-1} e^{j\frac{\pi}{N}(K_0-k-1)n} \\ +e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)}\sum_{n=0}^{2N-1} e^{-j\frac{\pi}{N}(K_0+k)n} - e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)}\sum_{n=0}^{2N-1} e^{j\frac{\pi}{N}(K_0+k)n} \end{array}\right]$$

And by using the fact that $\sum_{n=0}^{2N-1} e^{j\frac{\pi}{N}kn} = \sum_{n=0}^{2N-1} e^{-j\frac{\pi}{N}kn} = 2N \cdot \delta[k]$ we have:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} \quad = \tag{C.11}$$

$$(-1)^{pK_0}\frac{NA}{4j}\left[\begin{array}{l} \left(e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)} - e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)}\right)\cdot\delta\left[K_0+k+1\right] \\ +\left(e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}-\phi_0\right)} - e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}-\phi_0\right)}\right)\cdot\delta\left[K_0-k\right] \\ +\left(e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}-\phi_0\right)} - e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}-\phi_0\right)}\right)\cdot\delta\left[K_0-k-1\right] \\ +\left(e^{-j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)} - e^{j\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)}\right)\cdot\delta\left[K_0+k\right] \end{array}\right]$$

By combining every pair of exponents into one sine function we have the final expression, presented in (4.8):

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0=0} = (-1)^{pK_0}\frac{NA}{2}\left[\begin{array}{l} \sin\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}+\phi_0\right)\cdot\delta\left[K_0+k+1\right] \\ -\sin\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}-\phi_0\right)\cdot\delta\left[K_0-k\right] \\ +\sin\left(\frac{\pi(N+1)k}{2N}+\frac{\pi(N+2)}{4N}-\phi_0\right)\cdot\delta\left[K_0-k-1\right] \\ -\sin\left(\frac{\pi(N+1)k}{2N}+\frac{\pi}{4}+\phi_0\right)\cdot\delta\left[K_0+k\right] \end{array}\right] \tag{C.12}$$

C.1.2 The case of $\Delta_0 \neq 0$

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} \quad = \tag{C.13}$$

$$A\sum_{n=0}^{2N-1}\cos\left(\tfrac{2\pi}{2N}\left(K_0+\Delta_0\right)\left(pN+n\right)+\phi_0\right)\cdot\sin\left(\tfrac{\pi}{2N}\left(n+\tfrac{1}{2}\right)\right)\cdot\cos\left(\tfrac{\pi}{N}\left(n+\tfrac{N+1}{2}\right)\left(k+\tfrac{1}{2}\right)\right)$$

The sine and cosine functions are replaced by the explicit exponent expressions:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} \;= \tag{C.14}$$

$$\frac{A}{8j}\sum_{n=0}^{2N-1}\left[e^{j\left(\frac{\pi}{N}(K_0+\Delta_0)(pN+n)+\phi_0\right)}+e^{-j\left(\frac{\pi}{N}(K_0+\Delta_0)(pN+n)+\phi_0\right)}\right]\cdot\left[e^{j\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)}-e^{-j\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)}\right]$$

$$\cdot\left[e^{j\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)}+e^{-j\left(\frac{\pi}{N}\left(n+\frac{N+1}{2}\right)\left(k+\frac{1}{2}\right)\right)}\right]$$

By opening the brackets we get:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0}=\frac{A}{8j}\sum_{n=0}^{2N-1}\left[\begin{array}{l}
e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k+1)n+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\\
+e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k)n+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\\
-e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k)n+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\\
-e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k-1)n+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\\
+e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k-1)n+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\\
+e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k)n+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\\
-e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k)n+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\\
-e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k+1)n+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}
\end{array}\right] \tag{C.15}$$

Since some of the exponent parts do not depend on n, they can be taken out of the summation:

$$X_{(p)}^{MDCT}\left[k\right]|_{\Delta_0\neq 0} = \frac{A}{8j}\left[\begin{array}{l}
e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0+k+1)n}\\
-e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0+k+1)n}\\
+e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0-k)n}\\
-e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0-k)n}\\
+e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0+k)n}\\
-e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0+k)n}\\
+e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0-k-1)n}\\
-e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\sum\limits_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0-k-1)n}
\end{array}\right] \tag{C.16}$$

It is known that a sum of a finite geometrical series is: $\sum_{n=0}^{2N-1} a^n = \frac{1-a^{2N}}{1-a}$. And since k and K_0 are integer numbers we have:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{8j}\left[\begin{array}{l} e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0+k+1)}} \\ -e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k+1)}} \\ +e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k)}} \\ -e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0-k)}} \\ +e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k)}} \\ -e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0+k)}} \\ +e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0-k-1)}} \\ -e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k-1)}} \end{array}\right] \tag{C.17}$$

By taking $e^{\pm j\pi\Delta_0}$ outside of the brackets of each nominator, the equation can be written as:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{4}\sin(\pi\Delta_0)\left[\begin{array}{l} -e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\frac{e^{j\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0+k+1)}} \\ -e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\frac{e^{-j\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k+1)}} \\ -e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\frac{e^{j\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k)}} \\ -e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\frac{e^{-j\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0-k)}} \\ +e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\frac{e^{-j\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k)}} \\ +e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\frac{e^{j\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0+k)}} \\ +e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\frac{e^{-j\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0-k-1)}} \\ +e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\frac{e^{j\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k-1)}} \end{array}\right] \tag{C.18}$$

Applying the last step again on each denominator, we have:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{8j}\sin(\pi\Delta_0)\left[\begin{array}{l}
\frac{e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0+k+1)}} \\
-\frac{e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k+1)}} \\
+\frac{e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k)}} \\
-\frac{e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0-k)}} \\
+\frac{e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k)}} \\
-\frac{e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0+k)}} \\
+\frac{e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0-k-1)}} \\
-\frac{e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k-1)}}
\end{array}\right] \tag{C.19}$$

Gathering the pairs together we get:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{8j}\sin(\pi\Delta_0)\cdot \tag{C.20}$$

$$\left[\begin{array}{l}
\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)^{-1}\cdot\left\{\begin{array}{l} e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0+k+1)}} \\ -e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi(N+2)}{4N}+\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k+1)}} \end{array}\right\} \\
+\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)^{-1}\cdot\left\{\begin{array}{l} e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k)}} \\ -e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0-k)}} \end{array}\right\} \\
+\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)^{-1}\cdot\left\{\begin{array}{l} e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k)}} \\ -e^{j\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0+k)}} \end{array}\right\} \\
+\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)^{-1}\cdot\left\{\begin{array}{l} e^{-j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0-k-1)}} \\ -e^{j\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi(N+2)}{4N}-\frac{\pi(N+1)k}{2N}\right)}\cdot\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k-1)}} \end{array}\right\}
\end{array}\right]$$

And since each pair in the parentheses also forms a sine function:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{4}\sin(\pi\Delta_0)\cdot \tag{C.21}$$

$$\begin{bmatrix} \sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)^{-1}\cdot\sin\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi}{2}k+\pi\Delta_0-\frac{\pi}{2N}(K_0+\Delta_0)\right) \\ +\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)^{-1}\cdot\sin\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi}{2}k+\pi\Delta_0-\frac{\pi}{2N}(K_0+\Delta_0)\right) \\ -\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)^{-1}\cdot\sin\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi}{2}k+\pi\Delta_0-\frac{\pi}{2N}(K_0+\Delta_0)\right) \\ -\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)^{-1}\cdot\sin\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi}{2}k+\pi\Delta_0-\frac{\pi}{2N}(K_0+\Delta_0)\right) \end{bmatrix}$$

More gathering up yields the following expression:

$$X_{(p)}^{MDCT}[k]\,|_{\Delta_0\neq 0} = \tag{C.22}$$

$$\frac{A}{4}\sin(\pi\Delta_0)\cdot\begin{bmatrix} \sin\left(\pi(K_0+\Delta_0)p+\phi_0+\frac{\pi}{4}+\frac{\pi}{2}k+\pi\Delta_0-\frac{\pi}{2N}(K_0+\Delta_0)\right) \\ \cdot\left\{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)^{-1}-\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)^{-1}\right\} \\ +\sin\left(\pi(K_0+\Delta_0)p+\phi_0-\frac{\pi}{4}-\frac{\pi}{2}k+\pi\Delta_0-\frac{\pi}{2N}(K_0+\Delta_0)\right) \\ \cdot\left\{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)^{-1}-\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)^{-1}\right\} \end{bmatrix}$$

Which is the expression described by (4.10) in section 4.2.1.

C.2 Applying the DSTFT Transform

The DSTFT transform is applied with a window function:

$$w[n]=\sin\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)^2 \quad ,0\le n\le 2N-1 \tag{C.23}$$

Applying the DSTFT transform on the signal we have:

$$\begin{aligned} X_{(p)}^{DSTFT}[k] &= \sum_{n=0}^{2N-1} x[pN+n]\cdot w[n]\cdot e^{-j\frac{2\pi}{2N}nk} \quad ,0\le k\le 2N-1 \qquad (C.24) \\ &= A\sum_{n=0}^{2N-1}\cos\left(\frac{2\pi}{2N}(K_0+\Delta_0)(pN+n)+\phi_0\right)\cdot\sin\left(\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right)^2\cdot e^{-j\frac{2\pi}{2N}nk} \end{aligned}$$

C.2.1 The case of $\Delta_0 = 0$

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = A\sum_{n=0}^{2N-1}\cos\left(\tfrac{2\pi}{2N}K_0\left(pN+n\right)+\phi_0\right)\cdot\sin\left(\tfrac{\pi}{2N}\left(n+\tfrac{1}{2}\right)\right)^2\cdot e^{-j\frac{2\pi}{2N}nk} \tag{C.25}$$

Next we use: $\sin^2(\alpha) = \frac{1}{2} - \frac{1}{2}\cos(2\alpha)$

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = \frac{A}{2}\sum_{n=0}^{2N-1}\cos\left(\tfrac{2\pi}{2N}K_0\left(pN+n\right)+\phi_0\right)\cdot\left[1-\cos\left(\tfrac{2\pi}{2N}\left(n+\tfrac{1}{2}\right)\right)\right]\cdot e^{-j\frac{2\pi}{2N}nk} \tag{C.26}$$

The cosine functions are replaced by the explicit exponent expressions:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = \frac{A}{4}\sum_{n=0}^{2N-1}\left\{\begin{array}{l}\left[e^{j\left(\frac{\pi}{N}K_0(pN+n)+\phi_0\right)}+e^{-j\left(\frac{\pi}{N}K_0(pN+n)+\phi_0\right)}\right]\\ \cdot\left[1-\frac{1}{2}e^{j\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right)}-\frac{1}{2}e^{-j\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right)}\right]\cdot e^{-j\frac{\pi}{N}nk}\end{array}\right\} \tag{C.27}$$

Since K_0 is an integer number, then $e^{j\pi K_0 p} = e^{-j\pi K_0 p} = (-1)^{pK_0}$. The expression can be reduced to:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = \frac{A}{4}\cdot(-1)^{pK_0}\sum_{n=0}^{2N-1}\left\{\begin{array}{l}\left[e^{j\left(\frac{\pi}{N}K_0 n+\phi_0\right)}+e^{-j\left(\frac{\pi}{N}K_0 n+\phi_0\right)}\right]\\ \cdot\left(1-\frac{1}{2}e^{j\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right)}-\frac{1}{2}e^{-j\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right)}\right)\cdot e^{-j\frac{\pi}{N}nk}\end{array}\right\} \tag{C.28}$$

By opening the brackets:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = \tag{C.29}$$

$$\frac{A}{4}\cdot(-1)^{pK_0}\sum_{n=0}^{2N-1}\left\{\begin{array}{l} e^{j\left(\frac{\pi}{N}(K_0-k)n+\phi_0\right)}+e^{-j\left(\frac{\pi}{N}(K_0+k)n+\phi_0\right)} \\ -\frac{1}{2}e^{j\left(\frac{\pi}{N}(K_0-k+1)n+\frac{\pi}{2N}+\phi_0\right)}-\frac{1}{2}e^{-j\left(\frac{\pi}{N}(K_0+k-1)n-\frac{\pi}{2N}+\phi_0\right)} \\ -\frac{1}{2}e^{j\left(\frac{\pi}{N}(K_0-k-1)n-\frac{\pi}{2N}+\phi_0\right)}-\frac{1}{2}e^{-j\left(\frac{\pi}{N}(K_0+k+1)n+\frac{\pi}{2N}+\phi_0\right)} \end{array}\right\}$$

And, as in the case of MDCT, using $\sum_{n=0}^{2N-1}e^{j\frac{\pi}{N}kn}=\sum_{n=0}^{2N-1}e^{-j\frac{\pi}{N}kn}=2N\cdot\delta[k]$, we have the final expression, presented in (4.9):

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0=0} = \tag{C.30}$$

$$\frac{AN}{2}\cdot(-1)^{pK_0}\sum_{n=0}^{2N-1}\left\{\begin{array}{l} e^{j\phi_0}\cdot\delta\left[K_0-k\right]+e^{-j\phi_0}\cdot\delta\left[K_0+k\right] \\ -\frac{1}{2}e^{j\left(\frac{\pi}{2N}+\phi_0\right)}\cdot\delta\left[K_0-k+1\right]-\frac{1}{2}e^{j\left(\frac{\pi}{2N}-\phi_0\right)}\cdot\delta\left[K_0+k-1\right] \\ -\frac{1}{2}e^{-j\left(\frac{\pi}{2N}-\phi_0\right)}\cdot\delta\left[K_0-k-1\right]-\frac{1}{2}e^{-j\left(\frac{\pi}{2N}+\phi_0\right)}\cdot\delta\left[K_0+k+1\right] \end{array}\right\}$$

C.2.2 The case of $\Delta_0 \neq 0$

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0\neq 0} = A\sum_{n=0}^{2N-1}\cos\left(\tfrac{2\pi}{2N}\left(K_0+\Delta_0\right)(pN+n)+\phi_0\right)\cdot\sin\left(\tfrac{\pi}{2N}\left(n+\tfrac{1}{2}\right)\right)^2\cdot e^{-j\frac{2\pi}{2N}nk} \tag{C.31}$$

Again, by substituting $\sin^2(\alpha)=\frac{1}{2}-\frac{1}{2}\cos(2\alpha)$, and using the explicit exponent expressions instead of the cosine functions we get:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{4}\sum_{n=0}^{2N-1}\left\{\begin{array}{l} \left[e^{j\left(\frac{\pi}{N}(K_0+\Delta_0)(pN+n)+\phi_0\right)}+e^{-j\left(\frac{\pi}{N}(K_0+\Delta_0)(pN+n)+\phi_0\right)}\right] \\ \cdot\left[1-\frac{1}{2}e^{j\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right)}-\frac{1}{2}e^{-j\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)\right)}\right]\cdot e^{-j\frac{\pi}{N}nk} \end{array}\right\} \tag{C.32}$$

And after opening the brackets:

$$X_{(p)}^{DSTFT}[k]\left|_{\Delta_0\neq 0}\right. = \frac{A}{4}\sum_{n=0}^{2N-1}\left\{\begin{array}{l} e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k)n+\phi_0\right)} \\ +e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k)n+\phi_0\right)} \\ -\frac{1}{2}e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k+1)n+\frac{\pi}{2N}+\phi_0\right)} \\ -\frac{1}{2}e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k-1)n-\frac{\pi}{2N}+\phi_0\right)} \\ -\frac{1}{2}e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0-k-1)n-\frac{\pi}{2N}+\phi_0\right)} \\ -\frac{1}{2}e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{N}(K_0+\Delta_0+k+1)n+\frac{\pi}{2N}+\phi_0\right)} \end{array}\right\} \tag{C.33}$$

Taking the parts that does not depend on n outside of the summation we have:

$$X_{(p)}^{DSTFT}[k]\left|_{\Delta_0\neq 0}\right. = \frac{A}{4}\cdot\left\{\begin{array}{l} e^{j(\pi(K_0+\Delta_0)p+\phi_0)}\sum_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0-k)n} \\ +e^{-j(\pi(K_0+\Delta_0)p+\phi_0)}\sum_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0+k)n} \\ -\frac{1}{2}e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{2N}+\phi_0\right)}\sum_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0-k+1)n} \\ -\frac{1}{2}e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}+\phi_0\right)}\sum_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0+k-1)n} \\ -\frac{1}{2}e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}+\phi_0\right)}\sum_{n=0}^{2N-1}e^{j\frac{\pi}{N}(K_0+\Delta_0-k-1)n} \\ -\frac{1}{2}e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{2N}+\phi_0\right)}\sum_{n=0}^{2N-1}e^{-j\frac{\pi}{N}(K_0+\Delta_0+k+1)n} \end{array}\right\} \tag{C.34}$$

Again, using sum of a finite geometrical sequence, and since k and K_0 are integer numbers we have:

$$X_{(p)}^{DSTFT}[k]\left|_{\Delta_0\neq 0}\right. = \frac{A}{4}\cdot\left\{\begin{array}{l} e^{j(\pi(K_0+\Delta_0)p+\phi_0)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k)}} \\ +e^{-j(\pi(K_0+\Delta_0)p+\phi_0)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k)}} \\ -\frac{1}{2}e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{2N}+\phi_0\right)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k+1)}} \\ -\frac{1}{2}e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}+\phi_0\right)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k-1)}} \\ -\frac{1}{2}e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}+\phi_0\right)}\frac{1-e^{j2\pi\Delta_0}}{1-e^{j\frac{\pi}{N}(K_0+\Delta_0-k-1)}} \\ -\frac{1}{2}e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{2N}+\phi_0\right)}\frac{1-e^{-j2\pi\Delta_0}}{1-e^{-j\frac{\pi}{N}(K_0+\Delta_0+k+1)}} \end{array}\right\} \tag{C.35}$$

By taking a part of the exponent out of the brackets in each nominator and denominator, as was done in the MDCT, we have:

$$X_{(p)}^{DSTFT}[k]\left|_{\Delta_0\neq 0}\right. = \frac{A}{4}\cdot\sin\left(\pi\Delta_0\right)\cdot\left\{\begin{array}{l} \frac{e^{j(\pi(K_0+\Delta_0)p+\phi_0)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)}\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k)}} \\ +\frac{e^{-j(\pi(K_0+\Delta_0)p+\phi_0)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)}\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k)}} \\ -\frac{1}{2}\frac{e^{j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{2N}+\phi_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k+1)\right)}\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k+1)}} \\ -\frac{1}{2}\frac{e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}+\phi_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k-1)\right)}\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k-1)}} \\ -\frac{1}{2}\frac{e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}+\phi_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)}\frac{e^{j\pi\Delta_0}}{e^{j\frac{\pi}{2N}(K_0+\Delta_0-k-1)}} \\ -\frac{1}{2}\frac{e^{-j\left(\pi(K_0+\Delta_0)p+\frac{\pi}{2N}+\phi_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)}\frac{e^{-j\pi\Delta_0}}{e^{-j\frac{\pi}{2N}(K_0+\Delta_0+k+1)}} \end{array}\right\} \tag{C.36}$$

Grouping all the exponent into a single one, we get:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{4}\sin(\pi\Delta_0)\cdot \tag{C.37}$$

$$\left\{\begin{array}{l} \frac{e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0-k)+\phi_0+\pi\Delta_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)}+\frac{e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0+k)+\phi_0+\pi\Delta_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)} \\ -\frac{1}{2}\frac{e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0-k)+\phi_0+\pi\Delta_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k+1)\right)}-\frac{1}{2}\frac{e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0+k)+\phi_0+\pi\Delta_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k-1)\right)} \\ -\frac{1}{2}\frac{e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0-k)+\phi_0+\pi\Delta_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)}-\frac{1}{2}\frac{e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0+k)+\phi_0+\pi\Delta_0\right)}}{\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)} \end{array}\right\}$$

As can easily be seen, the expressions in each side of the parenthesis share the same nominator. Therefore, the expression can be written as:

$$X_{(p)}^{DSTFT}[k]\,|_{\Delta_0\neq 0} = \frac{A}{4}\sin(\pi\Delta_0)\cdot \tag{C.38}$$

$$\left\{\begin{array}{l} e^{-j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0+k)+\phi_0+\pi\Delta_0\right)}\cdot\left[\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k)\right)^{-1}\right. \\ \left.-\frac{1}{2}\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k-1)\right)^{-1}-\frac{1}{2}\sin\left(\frac{\pi}{2N}(K_0+\Delta_0+k+1)\right)^{-1}\right] \\ +e^{j\left(\pi(K_0+\Delta_0)p-\frac{\pi}{2N}(K_0+\Delta_0-k)+\phi_0+\pi\Delta_0\right)}\cdot\left[\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k)\right)^{-1}\right. \\ \left.-\frac{1}{2}\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k+1)\right)^{-1}-\frac{1}{2}\sin\left(\frac{\pi}{2N}(K_0+\Delta_0-k-1)\right)^{-1}\right] \end{array}\right\}$$

Which is the expression described by (4.11) in section 4.2.1.

Bibliography

[1] X. S. B.W. Wah and D. Lin, "A survey of error-concealment schemes for real-time audio and video transmissions over the internet," in *International Symposium on Multimedia Software Engineering. Proceedings.*, pp. 17 – 24, December 2000.

[2] O. H. C. Perkins and V. Hardman, "A survey of packet loss recovery techniques for streaming audio," *IEEE Network*, vol. 15, no. 5, pp. 40–48, 1998.

[3] G. Carle and E.W.Biersack, "Survey of error recovery techniques for ip-based audio-visual multicast applications," *IEEE Network*, vol. 11, no. 6, pp. 24–36, 1997.

[4] O. D.J.Goodman, G.B.Lockhart and W.Wong, "Waveform substitution techniques for recovering missing speech segments in packet voice communications," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1440–1448, 1986.

[5] P. Lauber and R. Sperschneider, "Error concealment for compressed digital audio," in *AES 111th convention*, (NY), pp. 1–11, September 2001.

[6] S. Quackenbush and P. Driessen, "Error mitigation in mpeg-audio packet communication systems," in *AES 115th convention*, (NY), pp. 1–11, October 2003.

[7] J. Lindblom and P. Hedelin, "Packet loss concealment based on sinusoidal extrapolation," in *ICASSP '02. Proceedings.*, pp. 173–176, May 2002.

[8] P. Stoica and E. Larsson, "Adaptive filter-bank approach to restoration and spectral analysis of gapped data," *The Astronomical Journal by the American Astronomical Society*, vol. 120, pp. 2163–2173, 2000.

[9] J. L. Y. Wang and P. Stoica, "Two-dimensional nonparametric spectral analysis in the missing data case," in *ICASSP '05. Proceedings.*, pp. 397–400, March 2005.

[10] P. S. H. Li and J. Li, "A new derivation of the apes filter," *IEEE Signal Processing Letters*, vol. 6, no. 8, pp. 205–206, 1999.

[11] J. Beggs and D. Thede, *Designing Web Audio, Chapter 5.* CA: O'Reilly & Associates, Inc., 2001.

[12] H. C. J-C Bolot and A. Garcia, "Analysis of audio packet loss in the internet," in *Workshop on Network and Operating Sys. Support for Digital Audio and Video. Proceedings.*, pp. 163–174, April 1995.

[13] J.-C. Bolot, "Characterizing end-to-end packet delay and loss in the internet," *Journal of High-Speed Networks*, vol. 2, no. 3, pp. 305–323, 1993.

[14] M. Handley, "An examination of mbone performance." USC Information of Science Institute Research Report: ISI/RR-97-450, 1997.

[15] B.Milner and A.James, "Robust speech recognition over mobile and ip networks in burst-like packet loss," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, pp. 223–231, 2006.

[16] B. E. H. Purnhagen and N. Meine, "Error protection and concealment for hiln mpeg-4 parametric audio coding," in *AES 110th convention*, pp. 1–7, May 2001.

[17] Y.Chen and B.Chen, "Model-based multi-rate representation of speech signals and its application to recovery of missing speech packets," *IEEE Trans. On ASSP*, vol. 5, pp. 220–231, 1997.

[18] "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.*

[19] C. O.J.Wasem, D.J.Goodman and H.G.Page, "The effects of waveform substitution on the quality of pcm packet communication," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 3, pp. 342–348, 1988.

[20] *ITU-T Recommendation G.711 Appendix I: A high quality low-complexity algorithm for packet loss concealment with G.711.*

[21] R. A.Stenger, K.B.Younes and B.Girod, "A new error concealment technique for audio transmission with packet loss," in *European Signal Processing Conference. Proceedings.*, pp. 1–4, September 1996.

[22] K. H.Sanneck, A.Stenger and B.Girod, "A new technique for audio packet loss concealment," in *IEEE Global Telecommunications Conference*, pp. 48–52, November 1996.

[23] *ITU-T Recommendation G.723.1: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 Kbit/s.*

[24] *ITU-T Recommendation G.728 Annex I: Frame or packet loss concealment for LD-CELP decoder.*

[25] *ITU-T Recommendation G.729: Coding of speech at 8 kbit/s using Conjugate-Structure Algebraic-Code-Exited Linear-Prediction (CS-ACELP).*

[26] S. C.A.Rodbro, M.N.Murthi and S.H.Jensen, "Hidden markov model-based packet loss concealment for voice over ip," *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 1–15, 2005.

[27] D. Pan, "A tutorial on mpeg/audio compression," *IEEE Multimedia*, vol. 2, no. 2, pp. 60–74, 1995.

[28] Y. Wang and M. Vilermo, "Modified discrete cosine transform-its implications for audio coding and error concealment," *AES Journal*, vol. 51, no. 1/2, pp. 52–61, 2003.

[29] M. V. Y. Wang, L. Yaroslavsky and M. Vaananen, "Some peculiar properties of the mdct," in *IEEE 5th International Confrence on Signal Processing. Proceedings.*, pp. 61–64, August 2000.

[30] J. Tribolet, "Frequency domain coding of speech," *IEEE Trans. On ASSP*, vol. 27, no. 5, pp. 512–530, 1979.

[31] P. S. E.G. Larsson and J. Li, "Amplitude spectrum estimation for two-dimentional gapped data," *IEEE Transactions on signal processing*, vol. 50, pp. 1343–1354, 2002.

[32] W.H.Foy, "Comparison of methods for spectral estimation with interrupted data," *IEEE Transactions on ASSP*, vol. 41, pp. 1449–1453, 1993.

[33] P. Stoica and J. Li, “An adaptive filtering approach to spectral estimation and sar imaging,” *IEEE Transactions on Signal Processing*, vol. 44, no. 6, pp. 1469–1484, 1996.

[34] H. C. R. Zeskind and M. Owen, “Robust adaptive beamforming,” *IEEE Transactions on Signal Processing*, vol. 35, no. 10, pp. 1365 – 1376, 1987.

[35] S. Haykin, *Adaptive Filter Theory.* New Jersey: Prantice-Hall, third ed., 1986.

[36] J. L. Y. Wang, P. Stoica and T. Marzetta, “Nonparametric spectral analysis with missing data via the em algorithm,” *Digital Signal Processing*, no. 15, pp. 191–206, 2005.

[37] P. Stoica and R. Moses, *Introduction to Spectral Analysis.* NY: Prentice-Hall, 1997.

[38] *ITU-R Recommendation BS.1387.1: Method for Objective Measurments of Perceived Audio Quality.*

[39] *ITU-T Recommendation P.862: Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs.*

[40] *ITU-R Recommendation BS.1116-1: Methods for the Subjective Assessment of Small Impairments in Audio Systems Including Multichannel Sound Systems.*

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

[42] K. Brandenburg, “Mp3 and aac explained,” in *AES 117th International Conference on High Quality Audio Coding*, pp. 1–17, September 1999.

[43] H. Fletcher, "Auditory patterns," *Reviews of Modern Physics*, vol. 12, pp. 47–65, 1940.

[44] J. Johnston, "Transform coding of audio signals using perceptual noise criteria," *IEEE Journal on selected areas in communications*, vol. 6, no. 2, pp. 314–323, 1988.

[45] T. Painter and A. Spanias, "Perceptual coding of digital audio," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 451–515, 2000.

[46] "Information technology - generic coding of moving pictures and associated audio, part 3: Audio.," *International Standard IS 13818-3, ISO/IEC JTC1/SC29 WG11, 1994*.

[47] "Mpeg-2 advanced audio coding, aac," *International Standard IS 13818-7, ISO/IEC JTC1/SC29 WG11, 1997*.

[48] J. Herre, "Temporal noise shaping, quantization and coding methods in perceptual audio coding: A tutorial introduction," in *AES 17th Int. Conf. on High Quality Audio Coding*, pp. 1–14, August 1999.

[49] "The official mpeg site: http://www.chiariglione.org/mpeg/,"