Transmission of multimedia data over lossy networks

Martin Isenburg[1)]

TR-96-048

August 1996

1) International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704-1198, Suite 600, Fax: 510-643-7684. E-mail: isenburg@icsi.berkeley.edu . On leave from: Fraunhofer Institut Graphische Datenverarbeitung, Wilhelminenstrasse 7, D-64283 Darmstadt, Germany

E-mail: isenburg@igd.fhg.de.

Acknowledgements

Working in the excellent research environment of the International Computer Science Institute at Berkeley has been an unforgettable experience and I am grateful to everybody who supported my studies abroad.
First of all, I would like to thank Hartmut Chodura, my research advisor, for organizing my stay at the International Computer Science Institute and for his encouragement and advice throughout my studies. Without his guidance and patience during the final days this thesis would have never been completed.
For the support in the United States I want to express my appreciation to Dr. Rainer Storn of Siemens. In many fruitful discussions he provided me with all the necessary informations about signal processing and contributed with ideas and critical reviews to my work.
I want to express my appreciation to all members of the ICSI who I really enjoyed working with. Especially Dr. Andres Albanese, Prof. Michael Luby, Prof. Bernd Wolfinger, Rolf Sigle and Roy Chua supported me through advice and discussions.
Many thanks must go to Prof. Mladen Victor Wickerhauser for writing such great papers about wavelets and time-frequency analysis and to Prof. Jean-Chrysostome Bolot for his papers about packet audio.
For their contribution in making my stay at the ICSI possible I want to express my utmost gratitude to Prof. Dr.-Ing. José Luis Encarnação, Dr.-Ing. Stefan Noll and Norbert Schiffner.
Special thanks go to Kornel Knöpfle for carefully reviewing a draft of this thesis, to Joy Hollenback for showing me how to make cappuccino, to Rainer Klisch for providing shelter during my first Berkeley days, to the immigration police officer at Los Angeles airport for letting me enter the US, to Thu Nguyen for the motor bike rides through San Francisco, to Charn Pramuanvorachat for the three sets of Porsche rims in our kitchen and to Steven McKearney for the English lessons and the fun at tea time.
I also want to thank my parents whose time and effort helped my through the organizational difficulties prior to my departure to America.

Most of all I want to say ‚muchas gracias‘ to Jessica. It was her generous renunciation of a three month vacation to Peru - planned already a year ago - that allowed me to take this opportunity without pricks of conscience. Her love and her letters gave me energy and confidence during the time of separation.

for Jessica

Contents

I Introduction 9
1 Motivation 9
2 Overview 9
3 Channel requirements for real-time audio communication 10
4 Channel characteristics of packet switched networks 14
5 Graceful degradation for packet switched networks 15
6 Structure of the thesis 16

II Background 17
1 Digital audio signal processing 17
1.1 Filtering and Sampling 18
1.2 Quantizing 18
1.3 Digital filters 19
2 Standard audio codecs 21
3 Transmission over the Internet 23
4 Transmission problems 24
5 Packet loss statistics 25

III Scenarios 27
1 Simple transmission 27
2 Piggyback protected transmission 29
3 Priority encoded transmission 32

IV Time-frequency analysis 35
1 Continuous signals 35
2 Discrete signals 36
3 The time-frequency plane 38

V Wavelets 41
1 The wavelet transform 41
1.1 The discrete wavelet transform 41
1.2 The discrete Fourier transform 42
1.3 The discrete wavelet transform versus the discrete Fourier transform 42
1.4 Example 42
1.5 The mother scaling function and the average spaces 44
1.6 The mother wavelet and the detail spaces 45
1.7 The fast wavelet transform 46
1.8 Computing the discrete wavelet transform 48
2 The Haar wavelet 49
2.1 The Haar scaling function 50
2.2 The Haar wavelet 51
3 The Daubechies wavelet 52
3.1 The Daubechies scaling function 52
3.2 The Daubechies wavelet 54
3.3 Other Daubechies wavelets and scaling functions 55
4 The wave packet transform 58
5 The wave level transform 64
VI Audio encoding schemes 65
1 Transform coding 65
2 Transforming audio signals 67
3 Competing objectively 73
4 Competing subjectively 81
5 Compressing the transform coefficients 86
6 The new audio codecs 88
6.1 One layer wave compression 88
6.2 Multi layer wave compression 90
6.3 High FFT compression 92
7 Auditory testing 94
7.1 The Wave Audio Unit (WAU) 94
7.2 Auditory results 96
VII Résumé 99
1 Conclusion 99
2 Current work 100
3 Future work 100
Appendix - A short tutorial on linear algebra 101
A.1 Vector space 101
A.2 Inner product and orthogonality 101
A.3 Norms and normalization 102
A.4 Linear independence 102
A.5 Basis and dimension 102
A.6 Vector subspace 103
A.7 Basis transformation 103
References 105

I

Introduction

1 Motivation

“A network connects a number of distributed points and enables communication between them...”. With currently 10,000,000 hosts and 100,000 networks [32], the Internet represents a powerful heterogeneous communication network spanning the whole world. The rapid growth of the Internet transformed it from a field for scientific research to a mass medium for everyday use. With increasing bandwidth and decreasing transmission delay researchers have been tempted to exploit the Internet for something it had not been designed for: real-time multimedia communication. Recent developments [24], [36] and [44] demonstrate that environments like video and audio broadcasting, teleconferencing and full duplex multimedia communication over the Internet are basically feasible. However, the obtained results for video and audio communication are still of mediocre quality. All these tools suffer from the fact that the Internet, as the underlying transmission network, does not support real-time data streams.
Audio plays a big role for these environments and is usually the most important component in a multimedia communication. The appearance of numerous ‘Internet Phones‘ [28] that allow bidirectional point to point voice connections reflects the strong demand for this kind of audio applications. Realizing a robust audio transmission scheme for the Internet still remains a major task for research.

2 Overview

Distributing real-time data, and specifically audio data, over networks that do not provide guaranteed resources such as bandwidth or guaranteed performance measures such as maximum delay or maximum loss rate is considered in the following.
Audio communication over packet switched networks - like the Internet - is often degraded due to packet losses and varying packet arrival times. From one point in the network the audio data is sent in packets to another point. One important characteristic of a packet switched network is the delay required to deliver a packet from a source to a destination. Each packet generated by a source is routed to the destination via a sequence of intermediate nodes. Variable processing and queueing delays at each hop on the way to the destination sum up to a varying end-to-end delay. Packets may be rejected at intermediate nodes because of buffer overflow or they may be discarded because of transmission errors. Hence, another important characteristic of a packet switched network is its packet loss rate [4]. It appears that the quality of audio delivered from a source to a destination depends essentially on the number of lost packets and of the delay variations between successive packets [6]. Therefore one often encounters substantially reduced

audio quality. Unprotected streams of audio data are very sensitive towards transmission failures and the drop off in quality is enormous. Our goal is to employ encoding mechanisms that make this quality degradation more graceful.
Two approaches have emerged to support real-time applications over packet switched and lossy networks. One approach is to extend current protocols and switch scheduling disciplines to provide the desired performance guarantees. This approach requires that admission control, policing, reservation, and sophisticated scheduling mechanisms are implemented in the network. The design, analysis, and evaluation of such mechanisms is an active research area [5]. The other approach is to adapt applications to the service provided by the network. The idea is to have audio coders and decoders that adapt to the delay variance and loss characteristics of the network. This amounts to develop mechanisms that eliminate or at least minimize the impact of packet loss and delay jitter on the quality of audio delivered to the destination.
Efficient play-out adjustment mechanisms have been developed that minimize the impact of varying packet arrival times [35]. However, minimizing the impact of packet loss on the audio quality remains an active research task. Measurements of the packet loss process for audio streams over the Internet indicate that loss periods usually involve only a small number of consecutive packets when the Internet load is low to medium. This suggests that open loop error correction schemes based on forward error correction are adequate to reconstruct lost audio packets [6].
Two different scenarios - *broadcasting* like Internet radio and *full duplex communication* like Internet telephony - have to be kept in mind. For broadcasting the encoding procedure is allowed to be time consuming, whereas in full duplex communication processing delay is critical. We are looking into protection schemes like PET [33] and the one used for RAT [36] in the MICE project [30] in order to cope with both situations. We are investigating how efficient these protection schemes provide graceful quality degradation in the presence of packet loss and how much delay they are introducing. In order to allow graceful degradation the encoding of the audio signal has to be layered. Then the FEC mechanisms transmit the more important layers of the codification with redundancy distributed among several consecutive packets. This way packet loss does not affect the signal reconstruction at the destination. A major part of this paper focuses on a hierarchical encoding of audio signals using transform coding. The codecs developed here employ various wavelet transforms, the Fourier transform and the discrete cosine transform for a layered and compressed representation of an audio signal. These coders allow the perfect reconstruction of the original signal from the complete encoding information, whereas a reconstruction from fragments of the encoding yields into a graceful degradation of audio quality depending on the amount of data lost.

The next three sections still have an introductory character. We first investigate the requirements that real-time audio communication imposes on the transmission channel. Then we demonstrate that packet switched networks like the Internet do not accomplish these requirements and finally we outline possible methods to overcome these limitations. The last section is dedicated to give a general view of the structure and the organization of the thesis.

3 Channel requirements for real-time audio communication

Real-time audio communication is something that happens everywhere. How it should happen over a packet switched and lossy network like the Internet can be examined by looking at some real world examples. The user expects real-time audio communication over the Internet to be like real-time audio communication over any other media. At first we determine the vocabulary

that is used in the following since most of these expressions do not have a sharply defined meaning.

Definition 1
Communication is the process of exchanging information between two or more points. A sender transforms information into a communication signal which is transmitted in form of a transmission signal to a receiver. The communication signal and the transmission signal may be identical. The medium which is used for transmission is called channel. Depending on the quality of the channel the transmission signal received by the receiver is more or less identical to the one sent by the sender.
A communication is unidirectional if all points are either senders or receivers and it is bidirectional if they are both. A bidirectional communication can be either full duplex or half duplex. In a full duplex communication each point is able to send and receive simultaneously whereas only one point can send at a time in a half duplex communication.

Definition 2
A communication is real-time when it mimics the properties of a face to face communication between human beings.

The communication signal in audio communication is the sound wave, since this is the only signal that is perceivable by the human ear. Audio signal, sound signal, music signal or speech signal are common synonyms that refer to the sound wave in audio communication. A sound wave is continuous and contains a continuous stream of information. Therefore the channel for a real-time audio communication is required to be isochronous.

Conclusion 1
A channel for real-time audio communication has to assure a continuous and isochronous transmission of the sound wave.

The transmission signal for audio communication is not subject to any restrictions. Besides using the sound wave directly, there are techniques that explore radio waves, electromagnetic waves or even light beams to transmit the (transformed) audio signal. The transmission can be either analog or digital, corresponding to a continuous or a discrete representation of the sound wave (see figure 1). In case of an analog transmission, the transmission signal is a continuous transformation of the sound wave. For a digital transmission the sound signal is sampled in regular time intervals. Then the discrete sample values are encoded (usually blockwise) into the transmission signal.

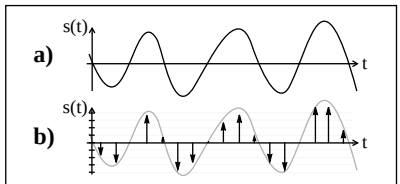


FIGURE 1. (a) analog, (b) digital representation of a sound wave

We look at a real world example for a bidirectional audio communication. When two people are dining in a restaurant, talking with each other, they are having a full duplex communication. This obviously is a real-time communication since a face to face conversation corresponds to our definition of real-time. The communication is full duplex because both people are able to speak and listen simultaneously (see figure 2). Each partner acts as a sender when saying something, and as a receiver when hearing something. The

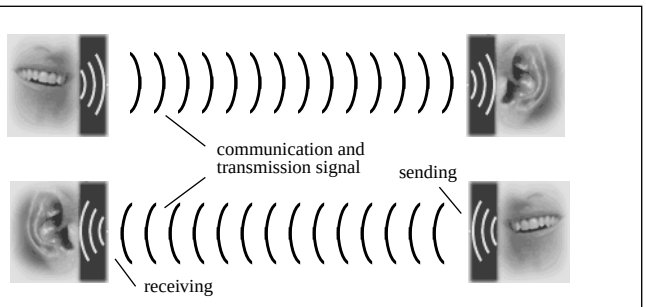


FIGURE 2. bidirectional full-duplex communication

speaker is transforming the information which consists of words, sentences, laughter etc. into a sound wave - the communication signal. Since the two guests are talking directly to each other the transmission signal is the same as the communication signal. The channel that transmits the sound wave is the physical surrounding of the two speakers.

In a usual dinner conversation one understands what the opposite party says, although the communication can be disturbed by a number of reasons. The waitress may interrupt a talk by asking, whether the dessert should be served. Or one tries to explain something while swallowing a huge portion of ‘nachos con guacamole‘. These incidents interfere with the communication either before or while the communication signal is produced. We want to focus on disturbances that affect the transmission signal on its way through the channel. Some mexican musicians may play decent salsa rhythms in the background and impair the conversation this way. The speech signal leaving the mouth of the speaker interferes with the music signal before it arrives at the listeners ear (see figure 3). The transmission signal is altered while passing through the channel.

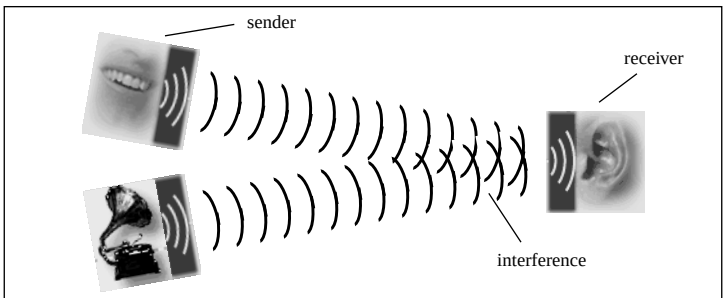


FIGURE 3. interference of a speech signal

Even if no band is playing music and the room is almost quiet, the communication between the two restaurant guests is slightly disturbed because the signal is weakened on its way to the recipient. Talking to a third person that is dining at another table in the restaurant confirms this conclusion. The speech signal leaving the mouth of the speaker is diminished before it arrives at the listeners ear (see figure 4). Again the transmission signal is altered while passing through the channel. The area between the two persons is obviously not an ideal channel. The condition of the channel has a direct impact towards the quality of the transmitted audio signal.

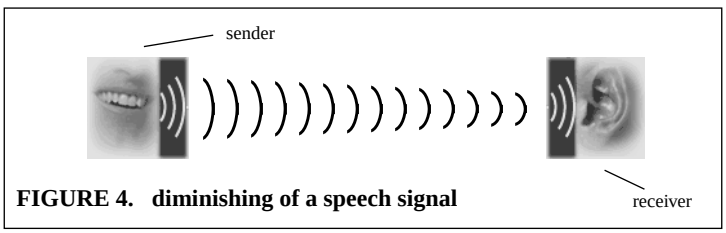


FIGURE 4. diminishing of a speech signal

Radio broadcasts on the other hand are a typical example for unidirectional audio communication. A radio station broadcasts a program which can be received by a large number of people. As in the last example the communication signal is a sound wave. But unlike in the last example the transmission signal is not the same as the communication signal. The station uses a radio wave to transmit the program instead (imagine a radio station using a sound wave!). Prior to sending, the sound wave is transformed into a radio wave and after receiving it is retransformed to a sound wave (see figure 5). Again the physical surrounding is the channel for broadcasting the radio wave. A thunderstorm,

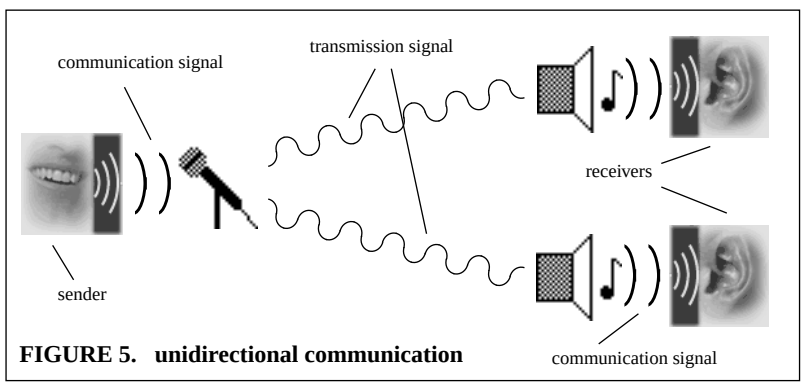


FIGURE 5. unidirectional communication

the neighbours broken hair dryer or a large distance to the radio tower influence the conditions of this channel. Altering the transmission signal - that is the radio wave - directly affects the sound wave.

The quality of the channel has a direct impact on how the transmission signal is altered during transmission. Because the transmission signal encodes the communication signal, a modification of the first results in an alteration of the latter. For understanding the influence that the quality of the channel has on the audio signal, we must observe both, the way the audio signal is encoded into the transmission signal and the way decreasing channel quality modifies the transmission signal.
For analog transmission methods the effects of bad channel conditions are noticeable immediately. Declining (improving) channel conditions directly increase (decrease) the modification of the audio signal. This graceful degradation of the audio signals quality is illustrated in figure 6. For digital transmission methods, like digital ISDN telephone, the behaviour is only slightly different. Whereas small losses in channel quality do not have effects on the audio signal, the distortion increases more rapidly with worse channel conditions than in the analog case.

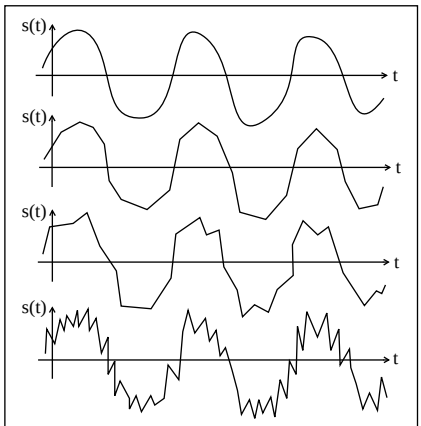


FIGURE 6. graceful degradation of an audio signals quality

The human ear is able to tolerate distortion in an audio signal up to a certain degree. We have to differentiate between noise that is not audible, noise that is audible but leaves the signal intelligible and noise that distorts the audio signal up to total unintelligibility. The transition between these states is gradual. Continuously increasing the amount of distortion results into a graceful degradation of the audio signals intelligibility from perfect to not understandable.
With decreasing channel quality the amount of distortion increases continuously and therefore gracefully degrades the quality of the audio communication. This is a very desired behaviour - the quality of the audio communication adapts to the quality of the channel.
Speaking about the quality of a channel is very general. We want to distinguish between different qualities that a channel offers.

Definition 3
The quality of a channel is determined by the qualities of its properties. These properties are:
- bandwidth
- transmission delay
- correctness of information
- transmission of information

The deterioration of the channel quality in our two real world examples was in fact the worsening of only one of the channels properties. Decreasing channel quality always meant decreasing correctness whereas the other channel properties remained unaltered. The channel guaranteed transmission with the same bandwidth and the same transmission delay throughout the communication. It was the decreasing correctness of the transmitted signal that resulted in a graceful degradation of the audio signal. The stable channel properties assured a continuous and isochronous transmission of the audio signal.

Conclusion 2
An ideal channel for real-time audio communication provides a guaranteed bandwidth and a constant transmission delay and assures a continuous and isochronous transmis-

sion of the audio signal. Worsening channel conditions affect only the correctness of the transmission signal.

4 Channel characteristics of packet switched networks

Packet switched and lossy networks such as the Internet do not have any of the characteristics a channel for real-time audio communication should have. The Internet does not assure a continuous and isochronous transmission of the audio signal. A worsening of the channel conditions does not affect the correctness of information, but bandwidth and transmission delay or even the transmission of information at all.

Using the Internet for the transmission of a continuous audio signal requires to split the audio signal into single packets an send them one by one as depicted in figure 7. A lossy network

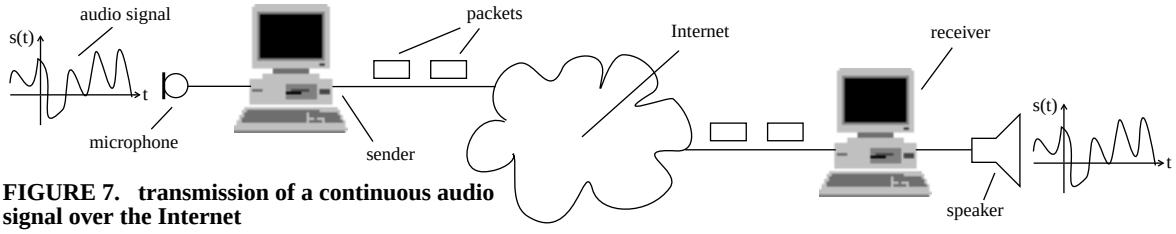


FIGURE 7. transmission of a continuous audio signal over the Internet

such as the Internet does not provide any guarantees for the transmission of the packets, so that unpredictable losses will occur inevitably. Obviously, one could use a network protocol that retransmits the data whenever packets are lost. However, the use of such reliable protocols (like TCP over IP) [23] usually results in increased transmission delay and the lack of control over the detection and handling of losses. In case of frequent losses packets accumulate at the sending side while the protocol keeps on trying to retransmit previously lost packets. For a real-time audio communication the value of a packet of audio information is strongly dependent on its actuality. A protocol that stores and therefore delays the most recent audio packets in order to transmit older ones that already lost their value for the communication is definitely ineligible. It is for this reason that most audio communication software uses unreliable protocols (like UDP over IP) as the preferred mode of transmission. In this way, losses can be detected and dealt with if necessary, and the application can decide whether retransmission is both necessary and tractable [9].

The successful transmission of a packet and the transmission delay depends on numerous factors, like the actual traffic load of the network, the chosen path of the packet through the network, the condition of routers, gateways and physical links. These circumstances make packet losses practically unavoidable and unpredictable and introduce a variation in packet arrival delays (jitter).

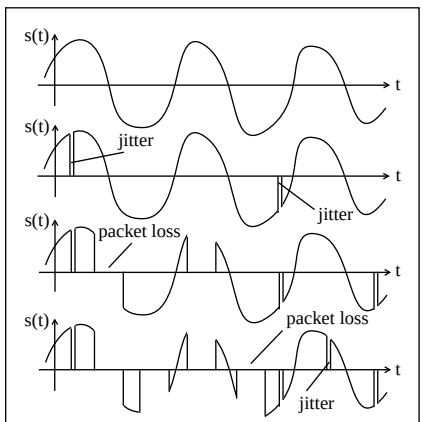


FIGURE 8. abrupt degradation of an audio signal

As a channel for the transmission of real-time audio data the Internet should have the same behaviour that we have observed for the two real world examples. With decreasing channel quality there should be a graceful degradation in the audio quality. For packet switched networks decreasing channel quality means an increase of packet loss and jitter. An unprotected stream of audio packets is very sensitive towards transmission failures and packet loss or jitter cause not a graceful, but a drastic drop off

in audio quality. Bad channel conditions result into an interrupted audio signal as depicted in figure 8. In the former scenarios decreasing channel quality meant a continuous degradation of the audio signals correctness. In the actual case decreasing channel quality means that some parts of the signal are completely lost, whereas the other remaining parts are still absolutely correct.
The reason for this is the way the audio information is transmitted combined with the way it is lost. Each packet corresponds to a small time interval of the audio signal and consecutive packets correspond to consecutive time intervals. The representation of the audio signal in every single packet is totally disjunct to that in any other packet. All packets together describe the complete audio signal. It is immediate that the loss of a single packet equals the loss of the corresponding time interval of the audio signal.

It is desirable to protect the audio information in a way that the loss of packets degrades the quality gracefully.

5 Graceful degradation for packet switched networks

In this paper we present two transmission schemes for real-time audio communication over packet switched networks that model the behaviour in regard to decreasing channel quality which other real-time communication channels have. With decreasing channel quality there is a graceful degradation of the quality of the transmitted audio signal.
For packet switched networks like the Internet decreasing channel quality means increasing packet loss and jitter. For achieving the desired graceful degradation we have to assure that loss of packets results into a decreasing correctness and not into a partly loss of the audio signal. In figure 9 this difference is depicted graphically.

How can we turn the abrupt degradation of audio quality into a graceful one? Packet loss means the loss of parts of the encoding information. With an encoding scheme where each part of the encoded information directly corresponds to a time interval of the audio signal, losses always result in an abrupt degradation of audio quality. The standard representation of audio signals in Pulse Code Modulation [38] (simple sampling and quantizing) for instance spreads the information about the signal evenly over the codification. For a signal that is sampled with 8 kHz each sample represents 0.125 ms of the signal. The loss of 300 samples for instance equals the loss of 37.5 ms of the audio signal in time.
Therefore we need an audio encoding scheme that allows the reconstructions of the audio signal at different levels of quality. Layered audio encoding schemes are designed to deal will the loss of encoding information. Such an encoding scheme concentrates the rough overall shape of the audio signal - the average information - in one portion of the codification and keeps the detail information in the remaining part. A more distinguished classification in very coarse, coarse, fine and very fine can yield even better results. A reconstruction from fragments of the encoding information gracefully degrades the sound quality depending on the amount of detail data lost.

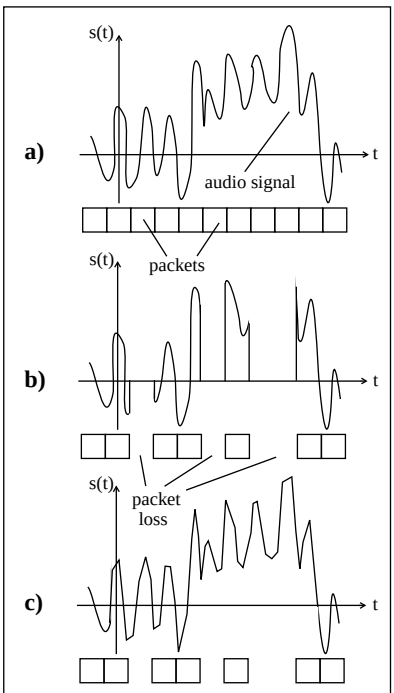


FIGURE 9. the original signal (a) and an abrupt (b) and a graceful (c) degraded version of the same

A major focus of this paper is to discuss and to develop a layered audio encoding scheme with exactly these properties.

How can we assure that packet loss affects the detail information first? For a layered audio encoding scheme some parts of the codification are more important than others. As described before the quality of the audio signal degrades gracefully depending on the amount of detail data lost. Since the loss of packets is unpredictable the way we transmit the encoding has to assure that detail information is lost first.
The two approaches to discuss both introduce redundancy to protect the more important parts of the codification. One such resilient scheme - the Priority Encoding Transmission (PET) [33] - was developed by researchers from the International Computer Science Institute (ICSI) at Berkeley. The other approach was designed within the ESPRIT project ‘Multimedia Integrated Conferencing for European Researchers’ (MICE) [30] at the University College London (UCL) and has already been successfully integrated into the ‘Reliable Audio Tool’ (RAT) [36].

6 Structure of the thesis

This paper claims to be a stand-alone introduction into the problems involved with real-time transmission of multimedia information over lossy packet switched networks focusing on the case of audio communication.
In chapter II we provide the necessary knowledge about audio processing and packet switched networks. The impact of network limitations towards real-time audio communication and two approaches that reduce/eliminate those are examined in chapter III. There we will see that the existing standard audio encoding schemes are not perfectly suited for the protection of audio streams against packet loss. In the remaining chapters we discuss and develop new audio codecs that have the desired properties. These audio coders are based on transforms of the audio signal within the time-frequency domain. In chapter IV the mathematical background of time-frequency analysis is given, while chapter V concentrates on ‘wavelets‘ - a certain family of transformations that are a very flexible tool for the time-frequency analysis of audio signals. The usefulness of these transforms for audio encoding purposes is carefully evaluated in chapter VI, where finally the resulting new and efficient audio codecs are presented.
After all the achieved results are briefly summarized and the direction of current and future research work is outlined in chapter VII.

II

Background

This chapter is meant to provide the necessary background knowledge for the reader about digital representation, processing and storage of audio signals and about the properties of packet switched networks like the Internet.
The first section gives a basic introduction into the way audio signals are processed for subsequent digital treatment. In the second section we describe some conventional audio encoding schemes and the quality that can be achieved using them. The third section introduces the transmission concepts of the Internet, while the next section concentrates on the problems related to the characteristics of such a packet switched network. Through the measurements in the last section we try to characterize the packet loss behaviour of the Internet.

1 Digital audio signal processing

The analog signal one receives from a microphone represents a sound wave through a continuous electrical voltage with varying amplitude. For digital signal processing purposes it is necessary to transfer this time continuous audio signal into a discrete representation. The conversion of an analog signal, such as the audio signal from a microphone, to a form in which it may digitally stored or manipulated requires three distinct processes: filtering, sampling and quantizing. Filtering is concerned with reducing the information in the signal to a capable amount.

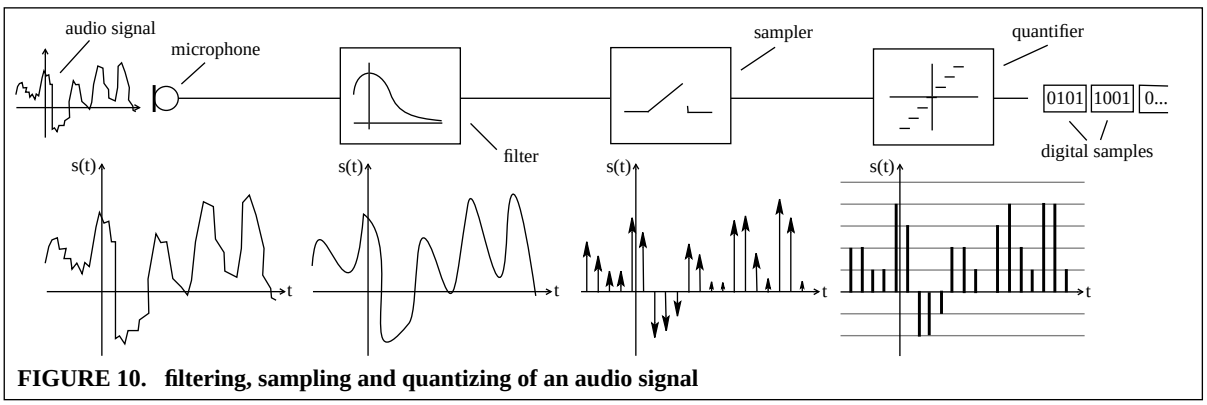


FIGURE 10. filtering, sampling and quantizing of an audio signal

Sampling is concerned with the capture of an analogue quantity of the signal at a certain instant in time. Quantizing is concerned with the representation of this quantity by a digital word of finite length [22]. In figure 10 these three processes are depicted graphically.

1.1 Filtering and Sampling

Sampling can be roughly defined as the capture of a continuously varying amplitude at precisely determined moments in time. Usually signals are sampled in equal distant time steps Δt. In this case a signal is said to be sampled with the sampling frequency $f_s = 1/(\Delta t)$, the reciprocal of the time step Δt. The number of samples taken per second is called the sampling rate. Telephone speech for example is sampled 8000 times per second resulting in a sample rate of 8 kHz.

The Nyquist sampling theorem: For sampling a signal in time steps Δt there is a special frequency f_c - called the Nyquist critical frequency - given by $f_c = 1/(2\Delta t)$ or $f_c = f_s/2$. If a continuous function $s(t)$ sampled in time steps Δt happens to be bandwidth limited to frequencies smaller than the Nyquist critical frequency f_c, then the function $s(t)$ is completely determined by its samples.

The Nyquist theorem states that the necessary minimum sampling frequency is twice the highest frequency of the signal. If a signal contains frequencies higher than the Nyquist critical frequency it is not possible to accurately reconstruct the signal from its samples. A direct corollary of this is that the maximum frequency of a signal that can be represented is half the sampling frequency.
Since audio signals by nature are not band-limited they must be filtered prior to sampling. The 8 kHz sampled telephone signal for example is band-limited to a 3.1 kHz band ranging from 300 Hz to 3400 Hz, although a bandwidth of 4 kHz would be possible.

1.2 Quantizing

At some point the sampled analogue amplitude has to be converted to a digital word of finite length. This quantization process is generally done immediately after the sampler so that subsequent signal manipulation can be done digitally.

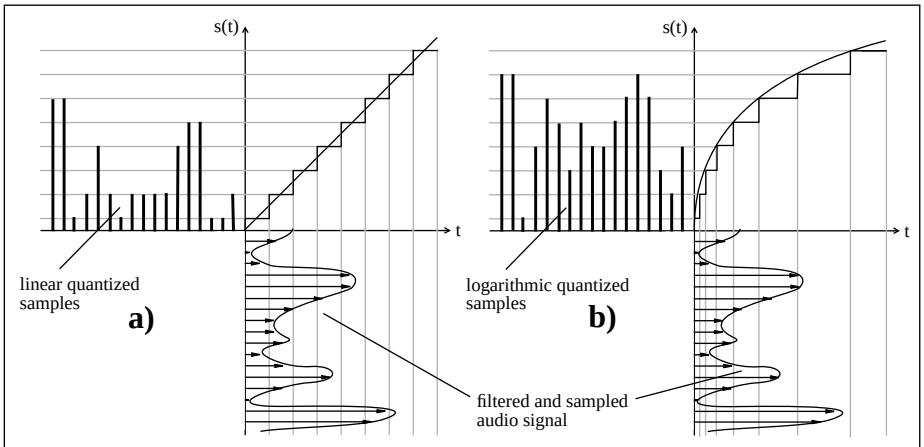


FIGURE 11. (a) linear and (b) logarithmic quantizing of a sampled

Every sample is rounded up or down to the closest fixed quantization level. Using 8 bits as the digital word length for every sample results in 256 different quantization levels. Common word length used for digital audio are 8, 12 and 16 bits. Each sample of the quantized signal differs from the original sample by the difference between the rounded and the original value. The error that is systematically introduced this way is called quantization error. Using more bits per sample increases the number of quantization levels and reduces the quantization error. The step size between the quantization levels does not have to be uniform. Especially for audio signals it makes sense to use a logarithmic scale to reduce the distortion ratio of the quantized signal (see figure 11). A logarithmic scale lowers the quantization error for small

amplitudes and - as a trade-off - raises it for larger amplitudes. Since the human ear is much more sensitive towards the disturbance of soft sounds than towards noise in loud sounds, non-uniform step sizes between the quantization levels improve the audible quality of the signal.
To quantize telephone speech a 13 bit uniform quantifier (i.e. 8192 reconstruction levels) is necessary to provide toll quality. Using a logarithmic scheme it is possible to obtain toll quality speech with a 8 bit logarithmic quantifier.

In the previous methods each sample was quantized independently from its neighbouring samples. Rate distortion theory tells us that this is not the most efficient method of quantizing the input data. It is always more efficient to quantize the data in blocks of n samples. The process is simply an extension of the previous scalar quantization methods described above. With scalar quantization the input sample is treated as a number on the real number-line and is rounded off to predetermined discrete points. With vector quantization on the other hand, the block of n samples is treated as a n-dimensional vector and is quantized to predetermined points in the n-dimensional space.
Vector quantization can always outperform scalar quantization. However, it is more sensitive to transmission errors and usually involves a much greater computational complexity than scalar quantization. The audio encoding schemes developed by the author use vector quantization.

1.3 Digital filters

A Finite Impulse Response (FIR) filter - which is used in chapter V - produces an output w_n that is the weighted sum of the current and past inputs v_i.

$$w_n = c_0 v_{n-m} + \ldots + c_{m-1} v_{n-1} + c_m v_n = \sum_{i=0}^{m} c_i v_{n-m+i}$$

The weights c_i are called filter coefficients. The FIR filter applied to a continuous sampled signal as depicted in figure 13 results in a filtered signal with attributes that depend on the chosen filter coefficients.

The FIR filter is not the one used for filtering prior to sampling. This band-limiting is done earlier by analog filters directly on the analog signal. The frequency response of a FIR filter determines which frequencies are kept in the filtered signal and thus which frequencies are discarded through filtering. This characteristic is typically illustrated by a frequency response curve as in figure 12. The normalized frequency on the x-axis ranges from 0 to 0.5. Multiplied with the sample rate of the filtered signal it ranges from the zero frequency to the Nyquist critical frequency f_c.

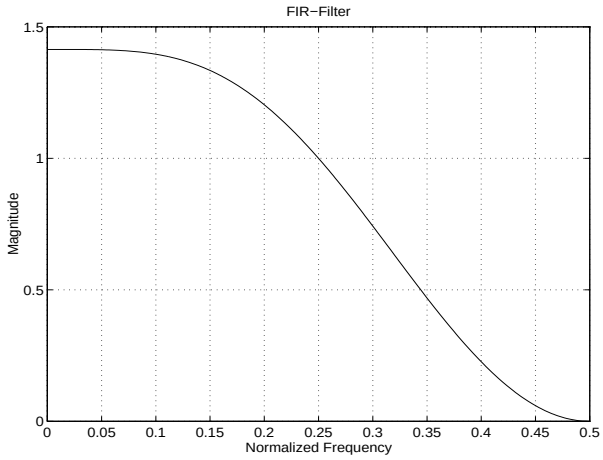


FIGURE 12. the frequency response curve of a Finite Impulse Respond filter

A Quadrate Mirror Filter (QMF) is a specially designed pair of distinctive Finite Impulse Response filters. The frequency responses of the two FIR filters separate the high-frequency and the low-frequency components of the input

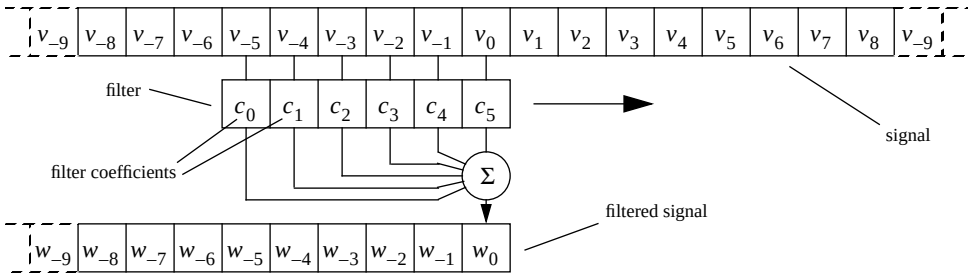


FIGURE 13. a Finite Impulse Respond filter with six filter coefficients applied to a signal

signal. The dividing point is usually halfway between 0 Hz and half the sampling rate of the input signal. The output of both the low-pass filter and the high-pass filter is decimated by two. That is, every other output sample of the filter is kept and the others are discarded as depicted in figure 14. This process is called down-sampling by two. The output of a QMF filter pair allows a perfect reconstruction of the original input signal using a corresponding pair of reconstruction filters. The previously down-sampled output values need to be upsampled by first inserting zeros between the coefficients as shown in figure 15. The process of low-pass and high-pass fil-

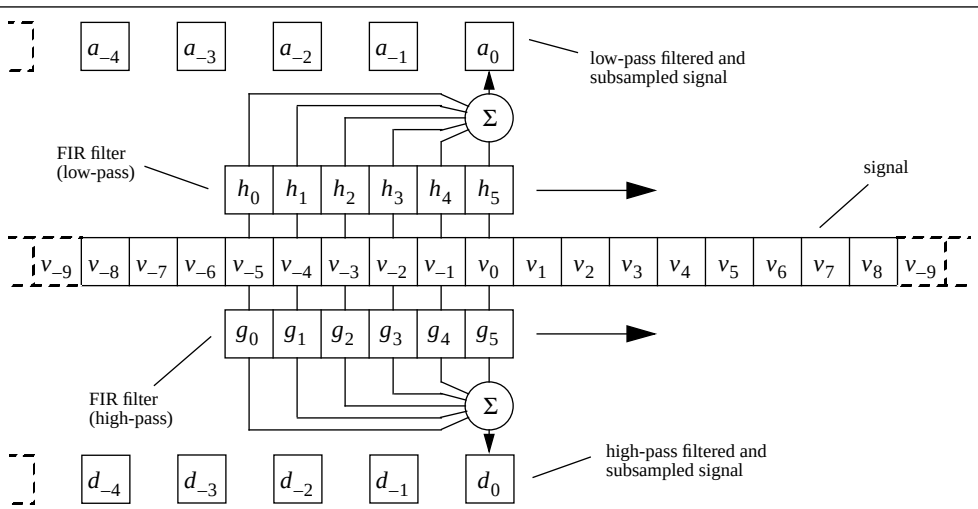


FIGURE 14. a Quadrate Mirror Filter pair applied to a signal

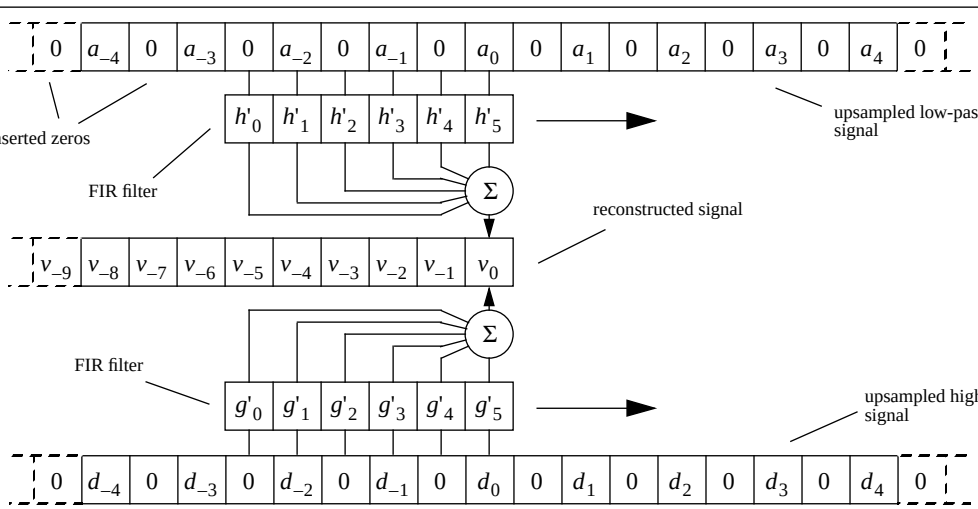


FIGURE 15. perfect reconstruction of the signal with a corresponding pair of filters

tering with successive down-sampling and its inverse is typically illustrated as subband filtering [34] like in figure 16.

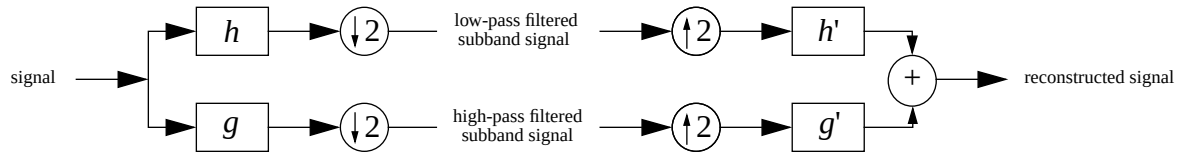


FIGURE 16. subband filtering and its inverse

2 Standard audio codecs

On the 14th of february 1876 Alexander Graham Bell filed an application with the US Patent Office for an ‘electric-speaking telephone’. The name comes from the Greek words for ‘far’ (tele) and ‘voice’ (phone). Nowadays the telephone sets the standard for full duplex person to person audio communication. The actual telephone signal extends from about 300 Hz to 3400 Hz, producing a bandwidth of 3100 Hz, which is based on the distance between the decisive highest and lowest frequency of the human voice. The nominal bit rate for toll or telephone quality speech is 64 kbps which conforms the available bandwidth on ISDN telephone lines. The telephone signal is uniformly sampled with 8000 samples per second using 8 bits to store the sample value.

There are quality orientated strategies for an improvement of the audio signal. Especially when considering high-quality audio transmissions it is necessary to increase the transmitted frequency bandwidth and/or to use more bits for storing the sample values. For digital CD audio quality the signal is sampled 44100 times per second and the sample values are stored with 16 bits depth.
Due to the bandwidth restrictions on the Internet there are numerous different standards for voice and audio encoding available. Most of them provide the means to achieve toll quality speech at bit rates beneath 64 kbps. Some of the most common are introduced here:

Pulse Code Modulation. Pulse Code Modulation (PCM) is the simplest type of audio encoding. It is essentially just a quantization process at sampling rates ranging usually between 8000 and 48000 samples per second. For linear PCM the sample values are quantized with a linear quantization function into either 8, 12 or 16 bits, while μlaw PCM uses only 8 bits and a logarithmic quantization function to amplify an audio signal. This results into an effective dynamic range of 13 bits and a higher resolution for small signal amplitudes. In fact the International Telegraph and Telephone Consultative Committee‘s (CCITT) G.711 standard defines 8 bit μlaw PCM as the standard method of coding telephone speech.

Differential Pulse Code Modulation. Because PCM makes no assumptions about the nature of the waveform to be coded, it works very well for any kind of signal. However, when coding speech or music there is a very high correlation between adjacent samples. This correlation could be used to reduce the resulting bit rate. One simple method of doing this is to transmit only the differences between each sample. This difference signal will have a much lower dynamic range than the original signal, so it can be effectively quantized using a quantifier with fewer reconstruction levels. For this method the previous sample is being used to predict the value of the present sample. Obviously the prediction is improved if a larger block of audio samples is used to make the prediction. This technique is known as differential pulse code modulation (DPCM).

Adaptive Differential Pulse Code Modulation. With DPCM both the predictor and the quantifier remain fixed in time. Greater efficiency could be achieved if the quantifier would adapt to the changing statistics of the prediction residual. Further gains could be made if the predictor itself would adapt to the audio signal. This would ensure that the mean squared prediction error was being continually minimized independently of the speaker and the speech signal.
Adaptive differential pulse code modulation (ADPCM) is very useful for coding speech at medium bit rates. ADPCM is a predictive coding scheme that exploits the correlation between neighbouring samples to compress the audio signal using a feedbackward adaptation scheme for both the quantifier and the predictor. The coder with 16 bit linear PCM at 8000 samples per second as input produces bit rates of either 48 kbps, 32 kbps or 16 kbps depending on the codec ADPCM6, ADPCM4 or ADPCM2 respectively. The computational complexity is very low and we can get almost toll-quality speech with 32 kbps whereas ADPCM2 results in a fairly distorted signal. The CCITT has formalized an ADPCM standard for coding telephone speech with 32 kbps. This is the G.721 standard.

GSM. This is a Regular Pulse Excited Linear Predictive Coder (RPE-LPC) that is used by GSM (Global System Mobile) telephones to reduce the data rate by a factor of almost five compared to ISDN telephone. The speech signal is divided into 20 millisecond intervals, each of which is encoded with 264 Bits, giving a total bit rate of 13.2 kbps. The result is only slightly beneath toll-quality speech, but the computational complexity of this coder is immense.

coding scheme	relative CPU cost	bandwidth
16 bit linear PCM	1	128 kbps
8 bit μlaw PCM	1	64 kbps
ADPCM6	13	48 kbps
ADPCM4	11	32 kbps
ADPCM2	9	16 kbps
GSM	1200	13.2 kbps
LPC	110	4.8 kbps

TABLE 1. relative CPU cost and bandwidth requirements of various coders

LPC. The Linear Predictive Coding (LPC) reduces the data rate by more than a factor of 12 but results in a signal that is way below toll quality. Several LPC standards have been defined that yield into bit rates down to 1 kbps. The most common used LPC encoder with a bit rate of 4.8 kbps achieves the greatest degree of compression among standard codecs but it is computational extremely intense. This synthetic quality speech coding algorithm is generally considered to contain about 60% of the information content of the speech signal. The overall shape of the frequency spectrum is preserved at the expense of short-term amplitude and pitch variation [19]. An LPC coder fits speech into a simple analytic model of the vocal tract, then throws away the speech and keeps only the parameters of the best-fit model. An LPC decoder uses those parameters to generate synthetic speech that is usually more-or-less similar to the original. The result is intelligible but sounds like a machine is talking. LPC compression is extremely sensitive to high frequency noise and clipping caused by setting the audio input level too high. Users with high pitched voices may not be able to use LPC compression at all: it just loses too much high-frequency information.

The relative CPU costs and the bandwidth requirements of the standard audio codecs are listed in table 1 on page 22. For packet audio systems these different codecs result in enormous size

differences for the respective packets. In table 2 we summarized example packet sizes for different time intervals using various coders at a sampling rate of 8 kHz.

coding scheme	20 ms 160 samples	32 ms 256 samples	40 ms 320 samples	64 ms 512 samples	80 ms 640 samples
16 bit linear PCM	320 bytes	512 bytes	640 bytes	1024 bytes	1280 bytes
8 bit µlaw PCM	160 bytes	256 bytes	320 bytes	512 bytes	640 bytes
ADPCM6	120 bytes	192 bytes	240 bytes	384 bytes	480 bytes
ADPCM4	80 bytes	128 bytes	160 bytes	256 bytes	320 bytes
ADPCM2	40 bytes	64 bytes	80 bytes	128 bytes	160 bytes
GSM	33 bytes	--------[a]	66 bytes	--------	99 bytes
LPC	12 bytes	--------	24 bytes	--------	48 bytes

TABLE 2. size of audio packets
a. only multiples of 160 can be encoded

3 Transmission over the Internet

Designed for a scenario where asynchronous transmission of data embodies the only form of communication, the Internet - a packet switched network - lacks support for real-time communication [5]. The Internet Protocol (IP) [20] uses packets called IP datagrams of a certain size, which is dictated by the physical network, as the atomic units of communication between two hosts in the network. These packets - equipped with a destination address - are routed connectionless from the sending to the receiving host. The transmission of IP datagrams offers neither reliability nor any other quality of service parameters (QoS). The packets may arrive damaged, out of order, duplicated, or not at all with a varying unpredictable transmission delay. This makes the Internet an unreliable and lossy network.

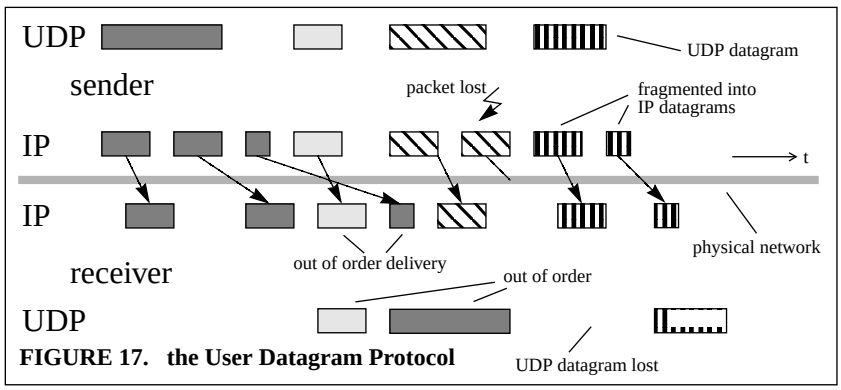


FIGURE 17. the User Datagram Protocol

The User Datagram Protocol (UDP) [20] is layered directly above the Internet Protocol (IP). It acts as the application interface of the Internet that directly reflects its transmission philosophy but abstracts from the physical environment. Transmission of data happens in terms of packets called UDP datagrams with a maximum size of 65535 bytes, which may be fragmented into several IP datagrams. The delivery of UDP datagrams is not guaranteed, consecutive packets may arrive out of order and even duplicate packets may be received. In case of transmission errors a packet is discarded by the protocol (UDP). For the application there is no difference whether a packet is lost during transmission and therefore does not even arrive or whether an erroneous packet does arrive but is silently discarded. This scenario is illustrated in figure 17. An UDP datagram and the corresponding IP datagram fragments are depicted by rectangles filled with the same pattern.

The Transmission Control Protocol (TCP) [20] is a reliable byte-stream protocol layered above the Internet Protocol (IP). It offers applications virtual connections that guarantee the correct in order delivery of data. The protocol handles failures that occur in the IP layer and uses sophisticated retransmission strategies in case of packet loss or damage.

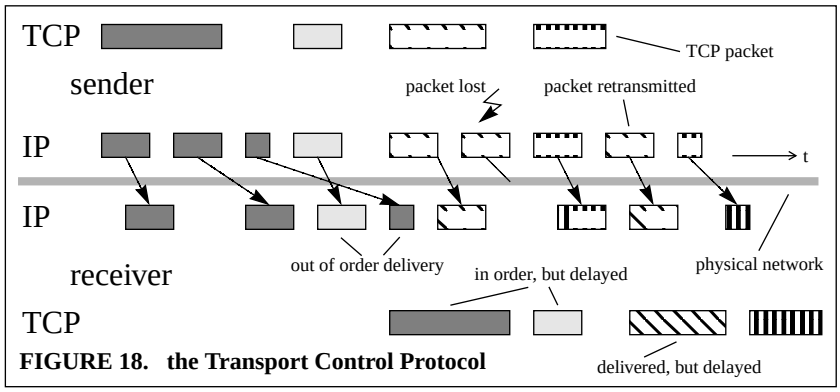


FIGURE 18. the Transport Control Protocol

4 Transmission problems

Employing some standard audio coding algorithm audio applications group the emerging stream of code words into packets for transmission over the network. The underlying transport architecture imposes several problems:

Packet loss. When transmitting over unreliable networks like the Internet losses will almost inevitably occur. The loss of packets is a persistent problem, particularly given the increasing popularity, and therefore increasing lead, of the Internet. Possible ways of combatting congestion include bandwidth reservation and moves towards an integrated service management on the Internet. These would require wide scaled changes to be agreed upon and implemented, so that these solutions will only be available in the medium to long term [19].

Transmission delay. Researches in the field of audio communication have indicated that humans can tolerate end-to-end delays between 150 to 300 ms in a two party conversation [9]. The end-to-end delay is the difference between the time the audio signal is produced by the sender and the time it is played at the receiver. Whereas for local area networks the transmission delay is almost negligible, it can be as high as 180 ms and more from the United states to countries in Europe. Next to the transmission delay there is another delay that is systematically introduced through the packetizing of audio data. Since a sender has to collect enough audio data to fill a packet before sending it, larger packets add more delay than shorter ones do. For full duplex audio communication packets usually contain 20 to 80 ms of the audio signal. For unidirectional communication like radio broadcasts delay is not a constraint at all.

Jitter. Audio packets are usually sent out at regular time intervals. However, at the receiving end packets do not arrive with fixed delays. This variation in packet arrival delays is called jitter. Each packet generated by a source is routed to the destination via a sequence of intermediate nodes. Variable processing and queueing delays at each hop on the way to the destination sum up to a varying end-to-end delay.

Bandwidth. The Internet does not provide a guaranteed bandwidth for the transmission of data. The available bandwidth depends on numerous factors. Being routed from hop to hop packets may be rejected at intermediate nodes because of buffer overflow or they may be discarded because of transmission errors. The actual bandwidth can never be estimated exactly - at the receiving end it can be calculated with the amount of received data within the past time interval.

5 Packet loss statistics

In this section, we characterize the packet loss process of audio streams sent over the Internet using measurements done by other researchers as well as investigations by the author.
In southern France the researchers Bolot, Crepin and Vega García have done a number of measurements between INRIA Sophia Antipolis and the University College London (UCL) in the UK, which were presented in [4], [5], [6] and [7]. In all experiments 320-byte packets are used that were sent periodically every 40 ms seconds. The plots in figure 19 by courtesy of Jean-Chrysostome Bolot show typical results for 30000 consecutively sent packets. The left picture illustrates the number of subsequently lost packets measured at 3:00 pm. The average loss rate of 0.21 is quite high because the INRIA-UCL connection is heavily loaded during daytime. However, it appears that most loss periods involve only one or two packets. This observation is confirmed by looking at the corresponding frequency distribution on the right. It shows the number of occurrences of n consecutive losses for different n. The slope of the distribution decreases linearly near the origin. Since the figure is drawn on a logarithmic scale, this indicates that the probability decreases geometrically fast away from the origin [7].

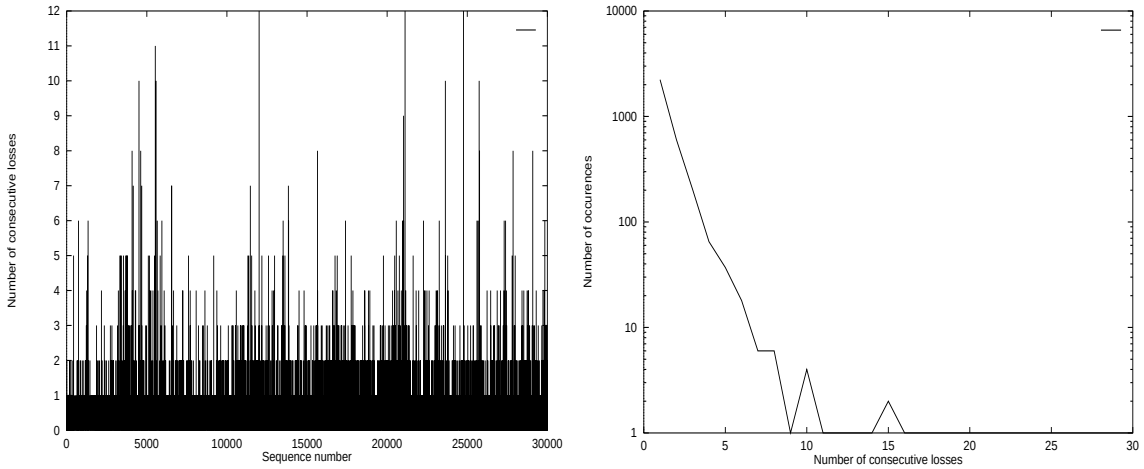


FIGURE 19. the packet loss measurements for 30000 packets of 320 bytes sent every 40 ms between the INRIA (France) and the UCL (United Kingdom) by courtesy of Jean-Chrysostome Bolot

Several measurements done by the author and Hartmut Chodura are to presented now. We used a small ping tool written by Hartmut that allows to send UDP datagrams of eligible size at some eligible clock rate from one host to the other. On the receiving side the arrival of the packets is observed and the occurrence of packet loss is recorded.
An intercontinental test series from the Fraunhofer-Institut für Graphische Datenverarbeitung at Darmstadt/Germany (FhG) to the International Computer Science Institute at Berkeley/USA (ICSI) confirmed the results of Bolot and García. At 3 pm PWT 14894 UDP packets of 400 bytes were sent with a 40 ms clock rate over this link. On the whole 583 packets were lost resulting into an average loss rate of 0.04. The equivalent plots of this measurement (figure 20) show that the frequency distribution of the number of consecutively lost packets is similar to that described above, even though the relative amount of lost packets was more than five times smaller.

To investigate the influence the sending clock rate has towards packet loss the same measurements (figure 21 on page 26) have been taken with a packet size and a clock rate both reduced by the factor 4. With UDP packets of 100 bytes sent every 10 ms the resulting bandwidth is identical to the one from the last test run. Of course the number of transmitted packets within

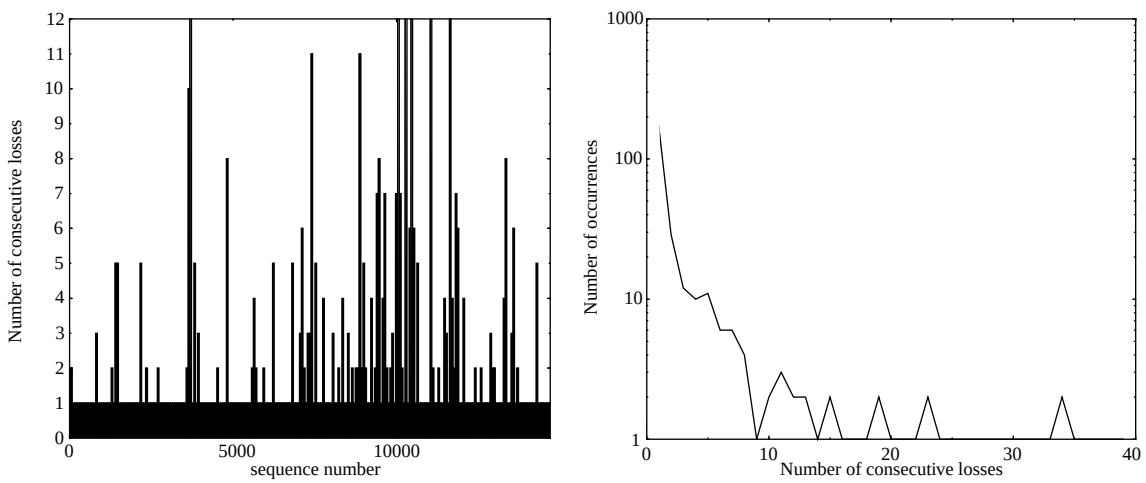


FIGURE 20. the packet loss measurements for 14894 packets of 400 bytes sent every 40 ms between the FhG (Germany) and the ICSI (United States)

the same time period is four times higher. From a total of 59599 sent packets 3567 did not arrive at the receiver which equals an average loss rate of 0.06. No doubt - this number alone does not allow any statements whether more small packets at a higher clock rate have a smaller

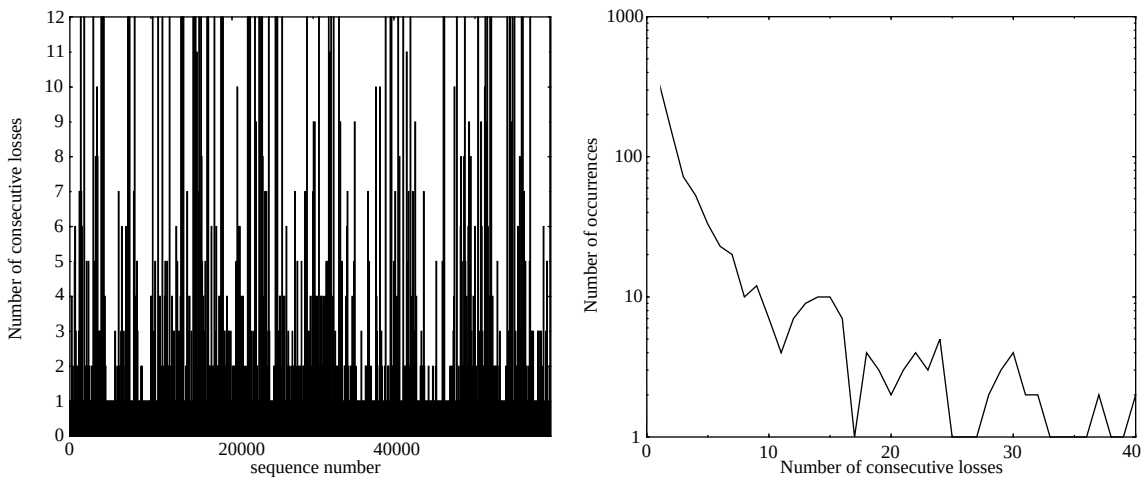


FIGURE 21. the packet loss measurements for 59599 packets of 100 bytes sent every 10 ms between the FhG (Germany) and the ICSI (United States)

throughput of data than less bigger packets at a lower clock rate. We did quite a couple of tests not only over the intercontinental link between the FhG and the ICSI but also within the United States from the Center for Research in Computer Graphics (CRCG) in Providence at the East coast to the ICSI on the West coast. Although it could not be proofed it seemed as if many small packets performed worse than fewer big packets in respect to amount of data per time. Other researchers share this opinion and talk about the ‘per-packet rather than size-of-packet network penalty for small packets‘ [19]. More exactly this should mean that for packets below the fragmentation size (remember that one UDP datagram may be fragmented into several IP datagrams) the number of packets has a much bigger impact towards the packet loss probability than the packet size. Thinking about the increased IO burden and the raised protocol overhead for many small packets, these statements have a rational foundation.

III

Scenarios

In the following we briefly introduce the way existing applications are already using the Internet to establish audio communication. Once again we point out why audio communication over heterogeneous and possibly lossy networks like the Internet still is a problem-child. Then we present two different approaches that overcome these problems.
The first section deals with the effects of common transmission problems of the Internet on the encoded audio stream and the standard solutions to cope with them. The next two sections introduce two very different transmission schemes that master the network limitations by adding redundancy to the audio encoding. Whereas the first of this transmission schemes can work with conventional audio encodings, the other requires the development of a completely new audio codec.

1 Simple transmission

A simple audio application processes the audio signal with a common audio codec, groups the code words into packets of a fixed size and sends them immediately over the network. On the receiving side arriving audio packets are decoded and fed into the audio device right away. How packet loss and jitter in packet arrival times affect the received audio signal is depicted graphically in figure 22. Bandwidth limitations derogate the transmission in form of additional packet loss, so that it results into the same outcome like ordinary packet loss.

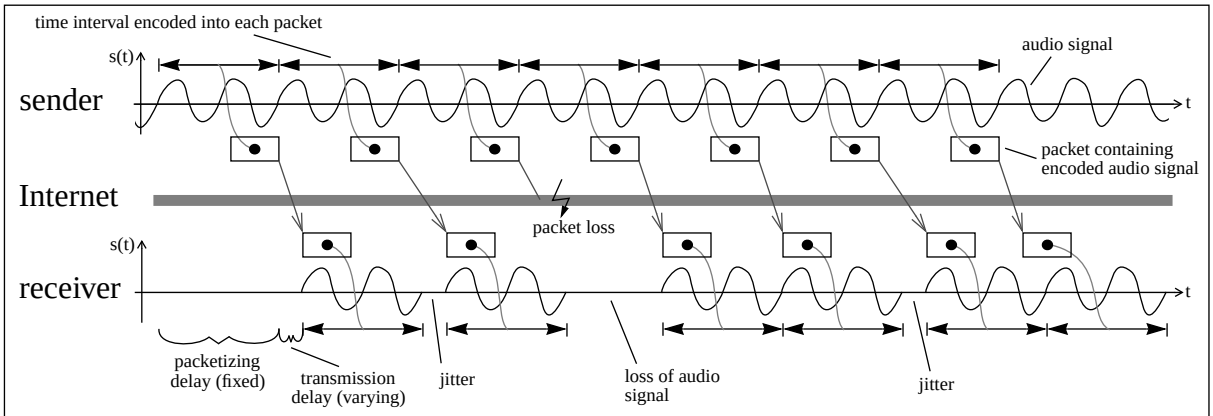


FIGURE 22. simple audio application affected by packet loss, jitter and delay

The audio quality suffers heavily from interruptions in the audio signal. We will introduce methods that fight the interruption through jitter and that turn the interruption through packet loss into a degradation in audio quality.

Fighting the transmission jitter. At the receiving side the packets do not arrive in uniform time intervals. Consequently the receiver must introduce an artificial *smooth out delay* to repair

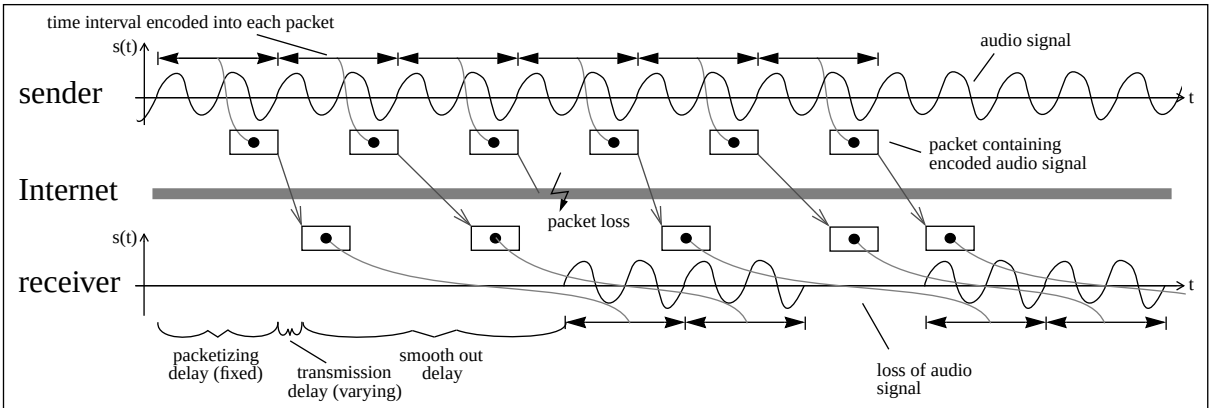


FIGURE 23. fighting the transmission jitter using a smooth out delay

the network effects. This enables the sample play-out to retain without interruptions. If the maximum end-to-end delay d_{max} between the source and a destination is known, then it is enough to delay the play-out time of every packet by an amount such that all packets are played back d_{max} after they have been sent by the source. However, the value of d_{max} is not known in advance for networks such as the Internet.

For interactive audio the necessary smooth out delay d_{max} can become too large. The goal then becomes that of delaying the play-out time by an amount such that most packets have been received before their scheduled play-back time. The smooth out delay may be set too high - leading to unnecessary delay - or to low - leading to losses due to late arrivals - if the delay and jitter processes in the network are not well understood [4]. A rough estimate of the reconstruction delay required to smooth out packet arrival times is two packets worth in ms, although the true value can be substantially in excess of this rule of thumb [19]. A *packetizing delay* equal to the size of one packet is incurred at the transmitter, since the samples for a packet have to be collected before the packet can be sent. Therefore a minimum of three packets worth of delay is incurred on an end-to-end basis, before the *transmission delay* of the network has been taken into account. In figure 23 this mechanism is depicted graphically. The introduced smooth out delay will be enough to receive most of the packets in time, but some will always arrive too late to be played back and can be considered lost.

For broadcast applications jitter in packet arrival times does not impose a major burden. A smooth out delay of several seconds can be introduced to assure that even late packets arrive in time to be played back.

Fighting the packet loss. Evidence from regular users of audio applications over the Internet indicates that mediocre audio quality is essentially due to excessive packet losses. This makes it important to implement an efficient loss recovery mechanism. Repair methods for packet loss can be either receiver-only techniques or combined channel and source techniques. Receiver-only techniques are those that try to reconstruct the missing segment of the audio signal solely at the receiver. Combined source and channel techniques are those that try to make the system robust to loss by either arranging for the transmitter to code the audio signal in such a way as to be robust to packet loss, and/or by transmitting redundant information about the signal. These techniques generally show significant improvement over receiver only techniques [19]. The two transmission schemes that will be introduced in the last two sections of this chapter are combined source and channel techniques.

Receiver-only techniques construct a suitable dummy packet for each lost packet, so that the loss is as imperceptible as possible. Conventional methods use either silence, white noise or the last correctly received packet as a substitution for lost packets like depicted in figure 24. Silence substitution is favored because it is simple to implement, but noise substitution and packet repetition have shown a subjective improvement over this method [46]. However, these mechanisms fail when packet sizes are large and/or the loss rate is high. A more detailed explanation of receiver-only techniques can be found in [18].

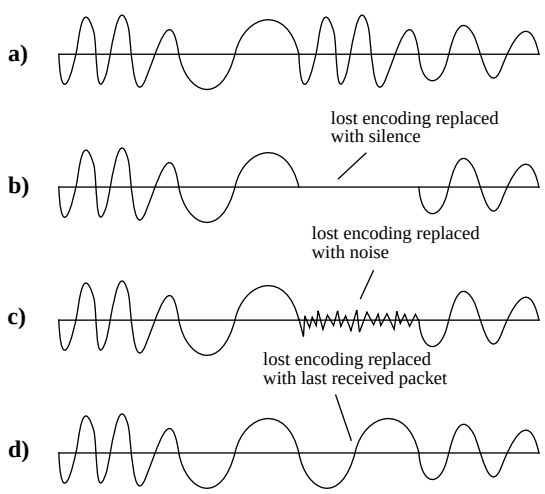


FIGURE 24. (a) original signal, (b) silence substitution, (c) noise substitution, (d) packet repetition

A loss recovery scheme using combined source and channel techniques is required if the number of lost audio packets is higher than tolerated by the listener at the destination. Loss recovery is typically achieved in one of two ways. With closed loop mechanisms such as Automatic Repeat Request (ARQ) mechanisms, packets not received at the destination are retransmitted. With open loop mechanisms such as Forward Error Correction (FEC) mechanisms, redundant information is transmitted along with the original information so that lost original data can be recovered from the redundant information.

ARQ mechanisms are generally not acceptable for real-time audio applications because they increase the end-to-end latency. FEC is an attractive alternative to ARQ for providing reliability without increasing latency [2]. However, the potential of FEC mechanisms to recover from losses depends crucially on the characteristics of the packet loss process in the network. FEC mechanisms are more effective when lost packets are dispersed throughout the stream of packets sent from a source to a destination [7]. Thus, it is important to evaluate the correlation between successive packet losses, or equivalently the distribution of the number of consecutively lost packets. The measurements of packet loss that we have done earlier indicate that FEC methods are particularly well suited for audio applications over the Internet.

2 Piggyback protected transmission

In this chapter we introduce a Forward Error Correction scheme (FEC) that was developed in the ESPRIT projects ‘Multimedia Integrated Conferencing for European Researchers’ (MICE) [30] an European funded project at the University College London and ‘Remote Language Teaching for SuperJANET’ (ReLaTe) [37]. The mechanism was implemented in the Robust Audio Tool (RAT) [36] written by Vicky Hardman and Isidor Kouvelas from the department of computer science at the University College London. The project MICE ended in September 1995 but the new project ‘Multimedia European Research Conferencing Integration’ (MERCI) [29] started in December 1995 with the aim of continuing the research themes of MICE. An almost parallel development that incorporates a retouched version of the same Forward Error Correction scheme is Free Phone [16]. Free Phone is an audio conferencing tool for the Internet designed by members of the High-Speed Networking group at INRIA. Andrés Vega-García wrote the core code of Free Phone which was first released in April 1996 as a part of his Ph.D.

thesis supervised by Jean Bolot. The work on Free Phone has been carried out within the MERCI project.
When asked how we should refer to this FEC scheme, Jean Bolot said „I refer to the FEC scheme as ‘the MICE FEC scheme‘ because we developed it within the MICE project (of which INRIA and UCL are part). I guess you could you refer to it as a piggybacking FEC scheme.“. We decided to call it ‘piggyback protected transmission‘.

Many FEC mechanisms proposed in the literature involve exclusive-OR operations. The idea is to send every nth packet a redundant packet obtained by exclusive-ORing the other n packets [39]. This mechanism can recover from a single loss within a series of n packets, but it increases the sending rate of the source by a factor of $1/n$, and it adds latency since n packets have to be received before the lost packet can be reconstructed.

The approach developed by the MICE researchers is a rather simple scheme. The idea is to append the most important information of every audio packet piggybacked to the following packet. In case a packet was lost a low resolution version can be reconstructed as soon as the next packet arrives. In other words with every packet n a redundant version of the previous packet $n-1$ is sent. When audio is transmitted using an PCM or an ADPCM encoding, the redundant version is typically obtained with a low bit rate codec such as LPC or GSM. This mechanism can recover from isolated packet losses. If packet n is lost, the destination waits for packet $n+1$, decodes the redundant information and sends the reconstructed samples to the audio driver. This mechanism is only feasible because of the *reconstruction delay* introduced at the receiving side as depicted in figure 25. Then the audio output consists of a mixture of PCM, LPC, GSM or ADPCM coded speech. The subjective quality of this reconstructed speech has been carefully evaluated within the MICE project at the UCL. The results show that the audio quality as measured by intelligibility hardly decreases as the loss rate reaches 30% even when a

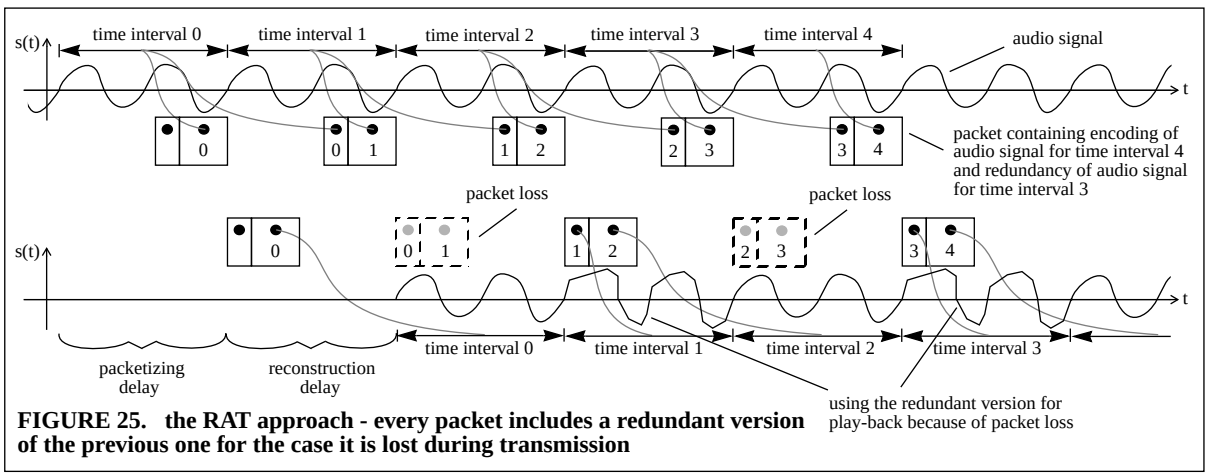


FIGURE 25. the RAT approach - every packet includes a redundant version of the previous one for the case it is lost during transmission

relatively low quality LPC coder is used to obtain the redundant information. LPC is a very CPU-intensive coding algorithm. However, it adds very little overhead to every audio packet. The used LPC codec increased the bandwidth by only 4.8 kbps. But this approach obviously fails if packet losses are bursty and consecutive packets are lost.

Even though most loss periods involve one packet it is important to recover from multiple losses. This is intuitively clear since longer loss periods have a larger impact on the audio quality than shorter periods do. Bolot has found in [7] that in high loss and high load situations, the most important task of an audio tool is to deliver decent quality audio to the destination. Thus, the approach is to increase the amount of redundancy carried in each packet. Faced with consecutive losses of two or more packets it is immediate to extend the previous approach by add-

ing to every packet n not only a redundant version of the previous packet $n-1$ but also redundant versions of other previous packets as depicted in figure 26. As redundant information

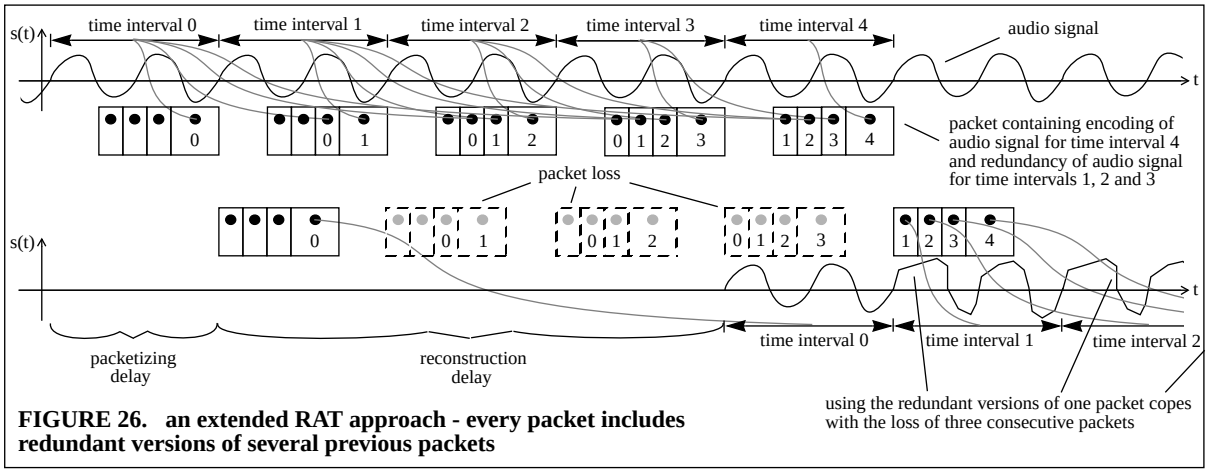


FIGURE 26. an extended RAT approach - every packet includes redundant versions of several previous packets

of every packet n we use LPC, GSM or ADPCM coded versions of the packets $n-1$, $n-2$ and $n-3$. Even though this scheme can reconstruct three consecutive packet losses one has to observe the huge *reconstruction delay* that needs to be introduced at the receiving side. In case the redundant information of packet n includes packet $n-i$, the reconstruction delay is i times the length of a packet. In other words the receiving side must delay the play-out of packet $n-i$ until packet n arrived. Otherwise the redundant information about packet $n-i$ carried within packet n would arrive too late to reconstruct the lost packet $n-i$.

There are different combinations to attach redundant versions of previous packets to a current packet, but clearly adding more redundant information increases the CPU usage and the bandwidth requirements. We follow the notation used by Bolot and Vega Garcia in [7] and use throughout the rest of the paper (coding algorithm, redundant algorithm(i)) to indicate that the audio packet n includes as redundant information packet $n-i$ encoded with the appropriate coding algorithm. Possible combinations of main and redundant information associated with CPU costs, bandwidth requirements and delay are listed in table 3. As expected adding redun-

combination	relative CPU cost	delay	bandwidth
(PCM)	1	1	64 kbps
(PCM, GSM(1))	1201	2	77 kbps
(PCM, LPC(1))	111	2	69 kbps
(ADPCM4, LPC(1))	121	2	37 kbps
(ADPCM4, LPC(1), LPC(2))	121	3	42 kbps
(ADPCM4, LPC(1), LPC(3))	121	4	42 kbps
(ADPCM4, LPC(1), LPC(2), LPC(3))	121	4	47 kbps

TABLE 3. relative CPU cost and bandwidth requirements of various coders

dant information increases the CPU costs, the bandwidth requirements and the delay. We note that the last few combinations in the table are clearly overkills if the network load is low and packet losses are rare occurrences. Mechanisms that adjust the amount of redundancy added at the source based on the loss process in the network as measured at the destination are required. Such mechanisms are described in [7] and have been successful implemented in [16].

We want to take a closer look on the redundant information that is added to every sent packet in the approach of Bolot and his research team. For instance the encoding combination

(ADPCM4, LPC(1), LPC(2), LPC(3)) - which can cope with the loss of three consecutive packets - transmits a lot of *redundant redundancy* in case only one packet is lost. Suppose packet n is lost, then we can either use the redundant version included in packet $n+1$, or the one from packet $n+2$ or else we choose the one in packet $n+3$. Three times exactly the same redundant information about the lost packet n is received. This is what was meant with *redundant* redundancy. One copy would have been sufficient - receiving the same redundant version of packet n three times does not improve the audio quality of its play-back. Only one copy is used to reconstruct the lost packet at a low audio quality, while the other two copies are simply discarded. Our idea is straight forward. The added redundancy should be made less redundant so that receiving more than one redundant version of packet n increases the quality of the reconstruction. Suppose we receive the packets $n+1$, $n+2$ and $n+3$, each of them carrying a redundant version of packet n. It is desirable to combine the three redundancies in order to get a reconstruction of better quality. The same holds when receiving only packet $n+1$ and packet $n+3$. Then we use the redundant information of two packets to reconstruct the lost packet n.
The standard audio codecs employed by Bolot and Vega-García do not qualify for this approach. These encoding schemes work either with the complete encoding information or not at all. What is needed is a codec that allows the reconstruction of the audio signal independent from the received amount of encoding information. The quality of the reconstructed signal should increase when more encoding information is available. Such an audio encoding scheme is presented in chapter VI.

Considering the two scenarios broadcasting and full duplex communication the transmission scheme introduced here is designed for the latter. In order to keep the imposed end-to-end delay as minimal as possible failures in case of bursty losses are accepted. This makes sense for time critical communication but for broadcast application the end-to-end delay is not a constraint at all. In such a scenario we can spent much more time on collection, processing and protection of the audio signal. Then it is possible to spread the encoding information over a larger number of packets so that even frequent consecutive packet losses do not affect the continuous reconstruction of the audio signal.

3 Priority encoded transmission

The work done by the author looks into employing a different Forward Error Correction scheme to protect the audio signal against the impact of packet loss. We want to compare this approach with the one of the MICE researchers and examine the advantages and disadvantages of both schemes in different scenarios.

A novel approach named Priority Encoding Transmission (PET) recently proposed in [1] and described briefly in [26] is an efficient Forward Error Correction scheme (FEC) that provides a general and flexible method to cope with packet loss. It has been designed for real-time data streams that consist of several parts with different importance. The user has the ability to assign different loss probabilities to each parts of the data stream. Then PET protects these parts with the appropriate redundancy and thus guarantees, that the more important parts arrive before the less important ones. The PET mechanism distributes all parts of the data stream together with the added redundancy over a certain number packets. The receiving side is able to recover the transmitted information in priority order, based on the number of packets received. This way packet loss at first affects the less important parts of the data stream yielding into a graceful degradation of its quality.
It is immediate that the data streams accomplished by this Forward Error Correction scheme

needs to have the following characteristics: The data stream must be layered in a way that some layers are more important than others. The data stream must retain of value for the receiver even if not all layers do arrive. Consequently the layers with the lowest importance are the first to be lost with worsening network conditions.

Another important feature of PET is to allow broadcasting in heterogeneous networks and to receivers with widely different capabilities. This is very suitable for multicast applications where information is sent through a lossy media to multiple receivers with vastly different processing powers and storage capabilities which are connected with widely varying bandwidth capacities. Even users with minimal resources or in congested parts of the network will be able to recover basic information, whereas users with high bandwidth connections and large computing resources will obtain high quality data.

The first step when using Priority Encoding Transmission is partitioning the data that need to be transmitted. These portions are the basic units called messages, which are encoded one at a time. A message then is split into segments which are dispersed using erasure codes [27] into a certain number of packets as depicted in figure 27.

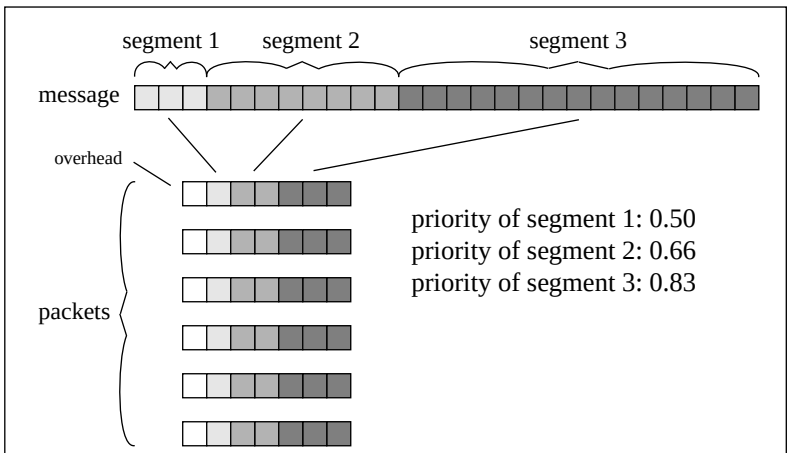


FIGURE 27. PET message striping process

This message striping process respects the user assigned priority for each segment. The given priority value equals the fraction of packets from a message that is needed to recover the respective segment. Regarding the previous example in figure 27 segment 1 can be recovered from any of three packets from the total of six packets. Although the segment 1 encompasses only around 11.54 % of the total message it covers a sixth of the encoding. Since any three packets are already sufficient to recover segment 1, the overall overhead sent for segment 1 is

segment	fraction of message	priority	fraction of encoding[a]	redundancy added	packets needed
1	11.54 %	0.50	16.66 %	100 %	3 of 6
2	30.77 %	0.66	33.33 %	50 %	4 of 6
3	57.69 %	0.83	50.00 %	20 %	5 of 6

TABLE 4. PET information distribution
a. not including the overhead of PET

100%. The corresponding values for the other segments are listed in table 4 in priority order. The total encoding length adds up to 138.46 % of the original message, not including the overhead of PET.

Due to bursty losses good quality video transmission with MPEG over lossy networks like the Internet was impractical. This forced the data communication industry to broadcast video using motion JPEG, which requires several times the transmission rate of MPEG. A version of PET was applied to the transmission of MPEG video over the Internet. In typical examples there was

a dramatic improvement of picture quality in the presence of losses when PET was used, even though the overall data sent was increased by only 20% over the original length [26].
Implemented in the framework of VIC, a video conferencing tool, each group of pictures (GOP) of the MPEG encoding was mapped on one message. The message was segmented into GOP header, the I-frames, the P-frames and the B-frames. The priority for each segment was decreasing in this order, so that in case of packet loss the B- and P-frames were lost first. The dependencies among different types of frames directly provided a somewhat natural priority scheme for PET. For a detailed description of this approach see [27].

Employing PET for an improvement in the transmission of audio data over lossy networks is a major issue of this paper. Looking through the common audio codecs introduced earlier there is not one that anyhow would provide a natural layering as it is needed for segmenting the audio information. These codecs do not provide a graceful degradation of audio quality when less than the complete original codification is available. A significant part of the work done by the author was to develop a layered audio encoding scheme that is suited to be protected by PET.

Again we consider the case of broadcast and full duplex communication. Some example calculations will show that the PET approach is only limited qualified to protect time critical real-time audio communication.
Let us assume a packetizing delay of 64 ms would be still acceptable. This roughly equals the delay that is introduced by the piggyback protected transmission scheme with two redundant packets at a packet size of 20 ms. Let us furthermore assume[1] an layered audio encoding scheme with three layers and an average bit rate of 45 kbps. The bottom most layer with a bit rate of 9 kbps should be protected with 50% priority, the next layer with a bit rate of 18 kbps should be protected with 80% priority and the last layer of the same bit rate remains unprotected. The bit rate of the protected audio stream is approximately 60 kbps without adding the PET overhead. Then 480 bytes of audio information and redundancy accumulate within 64 ms for one PET message, which we may spread over six packets. Then we may lose one packet while still receiving the second layer and up to three packets for keeping the bottom most layer. This might look promising but a closer investigation and a comparison to the RAT approach (ADPCM, LPC(1), LPC(2)) with the same delay and a lower bit rate shows some disadvantages. The piggybacking scheme sends three packets of 168 bytes while the PET approach sends six packets of 80 bytes. The piggybacking scheme can lose two out of the three packets without an interrupt in the audio signal, the PET approach can lose three out of six. Because of the ‘per-packet rather than size-of-packet network penalty for small packets‘ - see “Packet loss statistics” on page 25 - they are more likely to be lost. Furthermore the PET algorithm is rather complex and has not been designed to deal with packets of such a small size, according to Prof. Michael Luby, who is one of the inventors of PET.

The big advantage of PET over other Forward Error Correction schemes becomes visible in the broadcast scenario. There we are able to collect almost arbitrary much data for a PET message so that the striping mechanism can produce a big number of packets of a reasonable size. Suppose we allow a delay of five seconds and use the same coder and the same distribution of priorities as before. Then the data for one PET message accumulates to roughly 38000 bytes which may be striped into 76 packets of 500 bytes. As long as not more than half of them are lost a reconstruction of the audio signal at a low quality is possible. For this calculation we may lose as many as 15 out of the total of 76 packets and still be able to reconstruct the audio signal from two layers of the encoding.

1. This layered audio encoding scheme is presented in chapter VI. The bit rate used in this example calculation has a realistic foundation. The exactly achieved values are given on page 90.

IV

Time-frequency analysis

This chapter is meant to provide the reader with a theoretical knowledge beyond of what is actually needed to understand the concepts for robust audio transmission. As mentioned earlier a major part of the work done by the author was to develop an audio encoding scheme with the necessary properties to allow graceful degradation of quality in the presence of loss of the codification. Here we present the mathematical background for different possible representations of audio signals in the time-frequency plane. The discrete case - having an audio signal represented by sample vectors of n values - is of specific interest for us. The appendix contains a short tutorial on linear algebra for those who are not familiar with the concepts of a vector space.
It may be noted that the standard audio codecs introduced earlier represent an audio signal directly in the time domain and/or compress it within the time domain using prediction methods.

1 Continuous signals

In signal processing one deals with what is both result and measurement of a physical process - a signal. A signal as it is usually measured, describes a process through a continuously in time varying value h. Therefore a signal is simply a function $h(t)$ of time. Then this function $h(t)$ describes the physical process in the *time domain*. The same process can be specified in the *frequency domain* by giving its amplitude H (generally a complex number indicating the phase also) as a function of frequency f, that is $H(f)$. For many purposes it is useful to think of $h(t)$ and $H(f)$ as being two representations of the same function. One goes back and forward between these two representations by means of the Fourier transform equations:

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{2\pi ift}dt \qquad h(t) = \int_{-\infty}^{\infty} H(f)e^{-2\pi ift}df$$

Adapting the concept of a vector space from linear algebra the signal $h(t)$ can be seen as a vector of the infinite dimensional vector space of all square integrable functions $L_2(R)$. There are numerous different bases for the vector space $L_2(R)$ and thus every signal $h(t)$ can be represented by a coordinate vector $(h(t))_B$ relative to some basis B. The Fourier transform goes back and forward between the absolute time localized standard basis and the absolute frequency localized Fourier basis. Of course the coordinate vector $(h(t))_B$ has an infinite number of coefficients which takes away its practical use for most applications. But it should be mentioned that a coordinate vector $(h(t))_B$ - an infinite tupel of real numbers - can be treated like a

vector of the infinite vector space R^{∞}. It is immediate that every signal $h(t)$ can be exactly approximated by a vector of R^{∞}

2 Discrete signals

In digital signal processing it is not possible to work with continuous signals. Therefore a time continuous signal $h(t)$ is first band-limited and then sampled in regular time steps Δt. In theory sampling is done by a convolution of the signal $h(t)$ with a so called dirac impulse $\delta(t)$. A dirac distribution is defined as:

$$\delta(t) = \begin{cases} \infty & \text{for } t = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{with the characteristic} \qquad \int_{-\infty}^{\infty} \delta(t)dt = 1$$

The sample value of the signal $h(t)$ at the moment t_k is calculated with:

$$h(t_k) = \int_{-\infty}^{\infty} h(t)\delta(t - t_k)dt$$

This results into a sampled signal $h(t_k)$ where $t_k = k\Delta t$ for $k = -\infty, \ldots, \infty$. This is an infinite and steady stream of discrete sample values $h_k = h(t_k)$ that represent the original signal. When the time steps Δt are small enough, so that the sampling frequency $f_s = 1/\Delta t$ is twice the highest frequency of the signal $h(t)$, then the sample values h_k completely determine the signal.
In most cases subsequent processing of the signal uses discrete transformations. Since discrete transformations work on a finite number of samples, the steady stream of discrete sample values h_k with $k = -\infty, \ldots, \infty$ is chopped up into intervals of n samples with a time duration of $T = n\Delta t$. This segmentation of the sampled signal $h(t_k)$ results into sequences (h_k) from h_0 to h_{n-1} of n samples that represent the signal $h(t)$ within the corresponding time interval. The example in figure 28 depicts the whole process graphically.

For subsequent transformations it is useful to treat the sequence h_0 to h_{n-1} of n samples - a finite tupel of real numbers - as a vector $\bar{h}$ of the finite n-dimensional vector space R^n. The corresponding coordinate vector $(\bar{h})_S = \bar{h}$ addresses the segment of the signal in respect to the absolutely time localized standard basis S. The expression ‘absolute time localized‘ should be intuitively clear. Each coefficient of the coordinate vector $(\bar{h})_S$ addresses a certain sample h_k through the corresponding basis vector $\bar{e}_k$ from the basis S. Each sample h_k describes the signal $h(t)$ at a distinct - absolute localized - moment in time. The basis vectors $\bar{e}_k$ of the basis S correspond to dirac impulses $\delta(t - t_k)$ where t_k is the moment the sample h_k was taken.

The connection between a signal approximated by n sample values and a vector of the vector space R^n is immediate. Generally speaking any signal $h(t)$ of $L_2(R)$ that is represented by n sample values can be treated as a n-dimensional vector. Therefore the n-dimensional vector space R^n contains all signal representations with n samples.

Example: A continuous signal is sampled in regular time steps Δt resulting into a sampled signal $h(t_k)$ - an endless stream of discrete sample values $h_k = h(t_k)$
A sequence (h_k) of 16 sample values h_0 to h_{15} represents the continuous signal $h(t)$ within the corresponding time interval of length $T = 16\Delta t$. The 16 sample values h_0 to h_{15} of the time continuous signal $h(t)$ can be treated as the coefficients of a vector $\bar{h}$ of the finite vector space R^{16}. The identical coordinate vector $(\bar{h})_S = \bar{h}$ addresses the segment of the signal in respect to the absolutely time localized standard basis S. The basis vectors $\bar{e}_0$ to $\bar{e}_{15}$ correspond to the dirac functions $\delta(t - t_0)$ to $\delta(t - t_{15})$ where t_k is the moment sample h_k was taken.

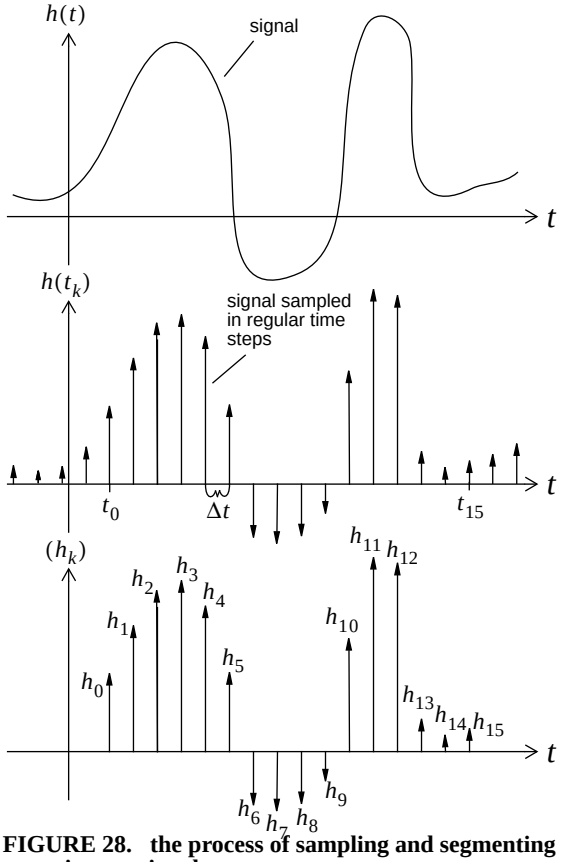


FIGURE 28. the process of sampling and segmenting a continuous signal

Each of the n sample values describes the signal at a specific moment of infinite short duration in time. A single sample directly corresponds to a specific dirac impulse. The basis vectors $\bar{e}_k$ of the basis S that are addressed by the n sample values act as a set of building blocks that approximate the signal. The time localization of these building blocks is still infinite short. For a with n points uniformly sampled signal it makes sense to widen the time localization of the building blocks up to the duration Δt of one sampling step. The duration Δt becomes the unit of length for the representation of the sampled signal. Then the n basis vectors $\bar{e}_k$ cover the represented sampling interval T with the thinnest patches allowed by the duration of the sampling steps. Furthermore the width of the visible and relevant portion of the time plane becomes n times the unit of length and therefore is n. In figure 29 this shall become more clear.

Example: The function $h(t)$ from the last example was represented by 16 sample values. The illustration of the basis vectors $\bar{e}_k$ of the vector space R^{16} is widened to building blocks of the time duration Δt. Then we use blocks rather than impulses to illustrate the vector $\bar{h}$ or the sequence (h_k) graphically.

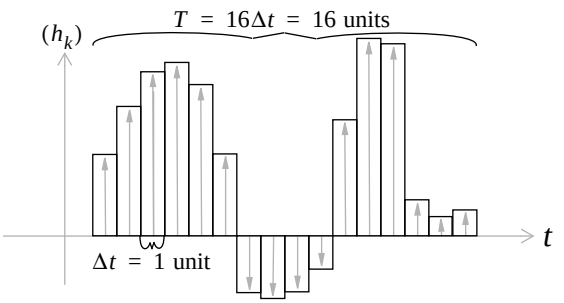


FIGURE 29. The time duration of one sampling step becomes the unit length of the representation.

3 The time-frequency plane

A signal can be represented either in the time domain, or in the frequency domain, or else somewhere in between in the time-frequency domain. Whether a representation is localized in time, in frequency or in both time and frequency, depends on which building blocks the representation uses to describe the signal. These building blocks called *time-frequency atoms* $\phi(t)$ are functions which act as the basis functions for the respective representation. The idealized time-frequency plane illustrates the properties time and frequency of the time-frequency atoms. A basis function is represented by a rectangle in this plane with its sides parallel to the time and frequency axis. Lets call such a rectangle an *information cell*. The range in time and the frequency covered by the time-frequency atoms corresponds to the sides of the cell. These time and the frequency measurements contain uncertainty and Heisenberg‘s inequality prevents us from making the product of the uncertainties smaller than a fixed constant [21]. The total signal power defined as

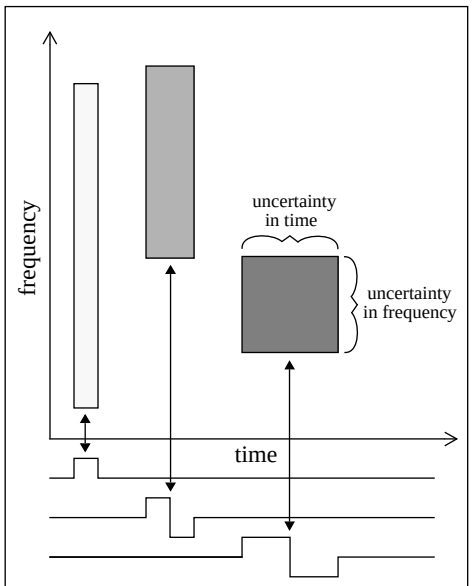


FIGURE 30. information cells in the time-frequency plane

$$\text{total power} = \int_{-\infty}^{\infty} \phi(t)^2 dt$$

of each building block is encoded by the darkness of the rectangles color. The plots of three arbitrary orthogonal basis functions are drawn schematically at the bottom of figure 30. The location of these time-frequency atoms in the time-frequency plane is illustrated in the graph above. The two at the left have small time uncertainty but big frequency uncertainty. The wider waveform at the right has smaller frequency uncertainty, so its information cell is not so tall as the ones for the narrower waveforms. It also contains more energy, so its cell is darker than the preceding two. A family of time-frequency atoms that completely covers the time-frequency plane is a basis for representing any signal of $L_2(R)$. An orthogonal basis is depicted as a cover of disjoint rectangles in this idealization.

Since we only deal with signals of finitely many points, we can construct a finite version of the time-frequency plane. If the signal is uniformly sampled at n points and we take the unit of length to be one sampling step, then the width of the visible and relevant portion of the time-frequency plane is n. A sequence (c_m) of n values spans the sampled signal $h(t_k)$ with a particular family of n time-frequency atoms $\phi_m(t)$ with:

$$h(t_k) = \sum_{m=0}^{n-1} c_m \phi_m(t_k) \qquad \text{for} \quad 0 \le k < n$$

In case the family of n time-frequency atoms $\phi_m(t)$ corresponds to the dirac impulses that were used to sample the signal, the sequence (c_m) that spans the sampled signal equals the sequence (h_k) of n taken samples.
It may be noted that for discrete signals sampled at n points the time-frequency atoms $\phi_m(t)$ are also no continuous functions but approximating sequences $(\phi_{m,k})$ or vectors $\underline{\phi}_m$ with n values.

The representation of a signal that is obtained by sampling in uniform time steps is absolutely localized in time. The basis functions or the time-frequency atoms of this representation have the same time-frequency characteristics as the corresponding dirac impulses. A dirac impulse is absolutely time localized and has absolutely no localization in frequency. This basis is called the standard basis.

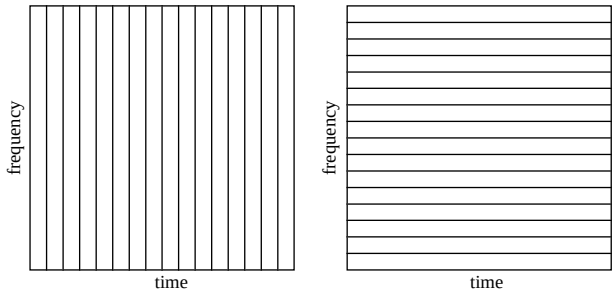


FIGURE 31. The standard basis (left) and the Fourier basis (right) tile a finite version of the time-frequency plane.

The Fourier basis on the other hand has optimal frequency localization, but no time localization. The dirac basis and the Fourier basis cover the time-frequency plane by the thinnest patches allowed by the sampling steps as depicted in figure 31. Both bases are orthogonal and the information cells of their time-frequency atoms are a disjunct cover of the finite time-frequency plane. The total signal power of a time-frequency atom $\phi_m(t)$ is calculated - since we are in the discrete domain - with

$$\text{total power} = \sum_{k=0}^{n-1} \phi_{m,k}^2$$

where $\phi_{m,k}$ is the respective value of $\phi_m(t)$ at the moment t_k. If the total power is 1 for all building blocks $\phi_m(t)$ the corresponding basis is normalized.

The wavelet theory in the next chapter introduces new families of time-frequency atoms that are also orthogonal tilings of the time-frequency plane with a different localization in time and frequency.

V

Wavelets

This chapter presents the wavelet theory following the chronological order of its development during the last thirty years. The emphasis that is put on the discrete case results from the fact that later the introduced wavelet transforms are applied to discrete signal representations.

1 The wavelet transform

Some very good introductions into the wavelet transform are given in [17], [41] and [45]. In this literature the continuous wavelet transform is usually introduced first, but we want to concentrate on the definitions for the discrete wavelet transform. We repeat the definition of the Fourier transform for a quick comparison of the two transforms.

1.1 The discrete wavelet transform

The discrete wavelet transform (DWT) transforms a function from the time domain, where the basis vectors correspond to dirac impulses $\delta(x)$ to the time-frequency domain, where the basis vectors are dilations and translations of a mother scaling function $\phi(x)$ and a mother wavelet $\psi(x)$ yielding into a hierarchically decomposition that describes a function in terms of coarse approximation and detail information ranging from broad to narrow [10].

The wavelet basis. The DWT uses a set of scaling functions $\phi_{j,k}(x)$ and a set of wavelets $\psi_{j,k}(x)$ as an orthogonal basis for representing other functions. Each scaling function $\phi_{j,k}(x)$ is a dilated and translated version of a mother scaling function $\phi(x)$ and respective each wavelet $\psi_{j,k}(x)$ is a is dilated and translated version of a mother wavelet $\psi(x)$. The scaling function and the wavelet are functions that satisfy certain requirements. These requirements assure that the set of dilations and translations of the mother scaling function $\phi(x)$ and the mother wavelet $\psi(x)$ define an orthogonal basis. The name ‘wavelet’ comes from the requirement that it should integrate to zero - waving above and below the x-axis.
The choice of the mother scaling function $\phi(x)$ and the mother wavelet $\psi(x)$ determines a wavelet basis. Therefore the attributes of a wavelet basis depend on the attributes of the corresponding mother functions. These attributes are localization in time, localization in frequency, degree of smoothness, compact support, symmetry and the number of vanishing moments [25]. Probably the most important characteristic is that scaling functions and wavelets are simultaneously localized in time and in frequency.

1.2 The discrete Fourier transform

A function in the time domain, where the basis vectors correspond to the dirac functions $\delta(x)$, is transformed by the discrete Fourier transform (DFT) into a function in the frequency domain, where the basis vectors are sines and cosines. The function can then be analyzed for its frequency content. The Fourier coefficients of the transformed function represent the contribution of each sine and cosine function at each frequency.

The Fourier basis. Sines and cosines of different frequencies are used by the DFT as an orthogonal basis for representing other functions. The frequencies of basis functions are equal distant distributed over the representable spectrum. Thus each frequency appears as one basis function which also can be phase shifted between 0 and 2π.

1.3 The discrete wavelet transform versus the discrete Fourier transform

Both, the DWT and the DFT, transform a function or a signal from the representation in the time domain into a different representation. For the DWT this new domain contains basis functions that are scaling functions and wavelets. For the DFT this new domain contain basis functions that are sines and cosines.
The similarity of both transforms is the frequency localization of the basis functions in the domain transformed to. The dissimilarity is that the functions of the wavelet basis are also localized in time, whereas the sine and cosine functions of the Fourier basis are not.
Sharp local transitions of the signal in the time domain result in numerous high frequency components in its DFT transformation. This is because the sharp time localized spike is modeled with smooth unlocalized sines and cosines. The ‘unlocalization‘ in time of each contributing sines and cosines function produces changes everywhere in the time domain. Thus the Fourier transform of the signal does not convey any information about the localization in time of the sharp transition. The different results obtained when applying the DWT and the DFT to a signal or a function with a sharp irregularity are illustrated in figure 32. Represented by 128 coefficients the top most plot depicts the signal in the time domain. The plots below show the coefficients that represent the same signal in respect to a wavelet basis and in respect to the Fourier basis. Both transformations have been normalized so that all basis vectors have a signal power of 1. It is very nice to see that the complete signal power that the original signal had in its sharp spike is almost evenly distributed over all coefficients of the Fourier transform.

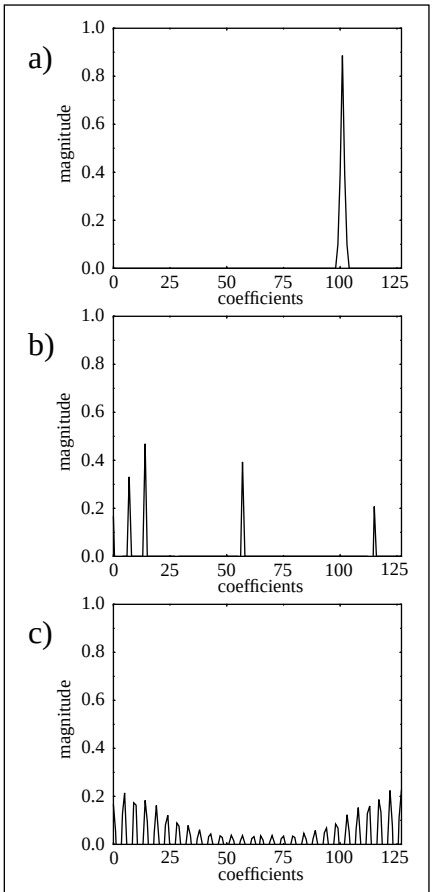


FIGURE 32. a signal with a sharp irregularity (a) transformed with the DWT (b) and the DFT (c)

1.4 Example

To get a sense how the discrete wavelet transform decomposes a function or a signal it is very helpful to follow a simple example step by step.

A function sampled with the eight values 2, 6, 9, 7, 1, 3, 4 and 8 is represented by the vector $\bar{u} = (2, 6, 9, 7, 1, 3, 4, 8)$ of the vector space R^8. The standard basis S of R^8 is the collection of unit vectors $\bar{e}_1$ to $\bar{e}_8$, which correspond to the dirac functions $\delta(t - t_1)$ to $\delta(t - t_8)$ respectively, where t_i is the moment sample u_i was taken. The coordinate vector of $\bar{u}$ in respect to the standard basis S is obviously identical $(\bar{u})_S = (2, 6, 9, 7, 1, 3, 4, 8)$.

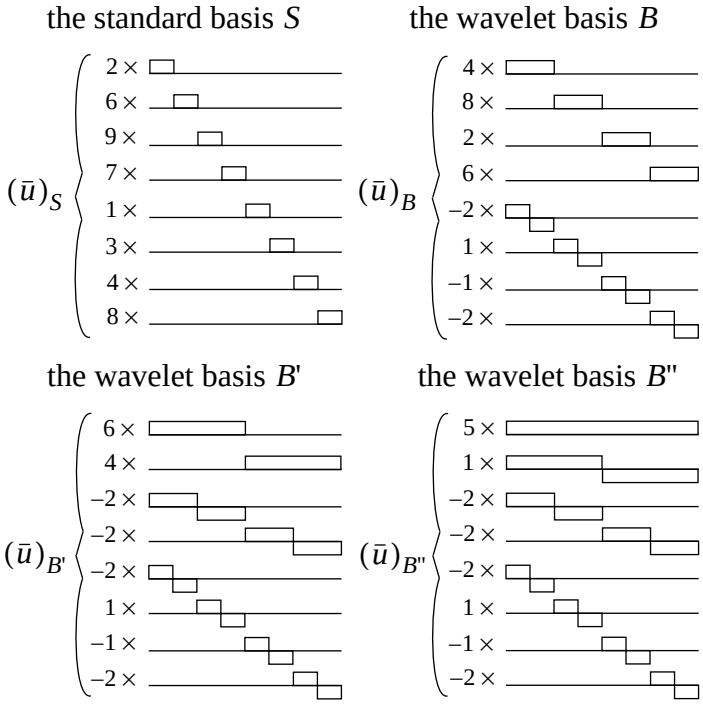


FIGURE 33. the basis vectors of the standard basis S and the three different wavelet bases B, B' and B''

The wavelet decomposition happens in several iterations. Every iteration means an orthogonal basis transformation.

Summing and halving the entries of $(\bar{u})_S = (2, 6, 9, 7, 1, 3, 4, 8)$ pairwise together results in the average coefficients a_1 to a_4 namely 4, 8, 2 and 6. The detail information that was lost in this averaging process will be stored in the detail coefficients.

Differenzing and halving the entries of the vector $(\bar{u})_S$ pairwise together results in the detail coefficients d_1 to d_4 namely -2, 1, -1 and -2. Together the average coefficients a_i and the detail coefficients d_i build the coordinate vector $(\bar{u})_B = (4, 8, 2, 6, -2, 1, -1, -2)$ in respect to the new basis B. The old basis S and the new basis B are depicted graphically in figure 33. A reconstruction of all the coefficients u_i of the original vector $(\bar{u})_S$ is done easily. The first entry u_1 of the vector $(\bar{u})_S$ is calculated with $u_1 = a_1 + d_1$, the second with $u_2 = a_1 - d_1$, the third with $u_3 = a_2 + d_2$.

An interesting feature is a reconstruction that omits the detail coefficients. This results in a coarse approximation of the original vector at a lower resolution.

The full wavelet decomposition is achieved by repeating the process of summing and differenzing recursively on the average coefficients is depicted in figure 34. The next iteration splits the coefficients a_1 to a_4 into the average coefficients aa_1 and aa_2 and the detail coefficients ad_1 and ad_2. The detail coefficients d_1 to d_4 are kept yielding into another coordinate vector $(\bar{u})_{B'} = (6, 4, -2, -2, -2, 1, -1, -2)$ regarding to another new basis B'. A reconstruction that uses only aa_1 and aa_2 produces an even coarser image of the original vector.

The final step decomposes the average coefficients aa_1 and aa_2 into one average coefficient

aaa_1 and one detail coefficient aad_1. This results in the final coordinate vector

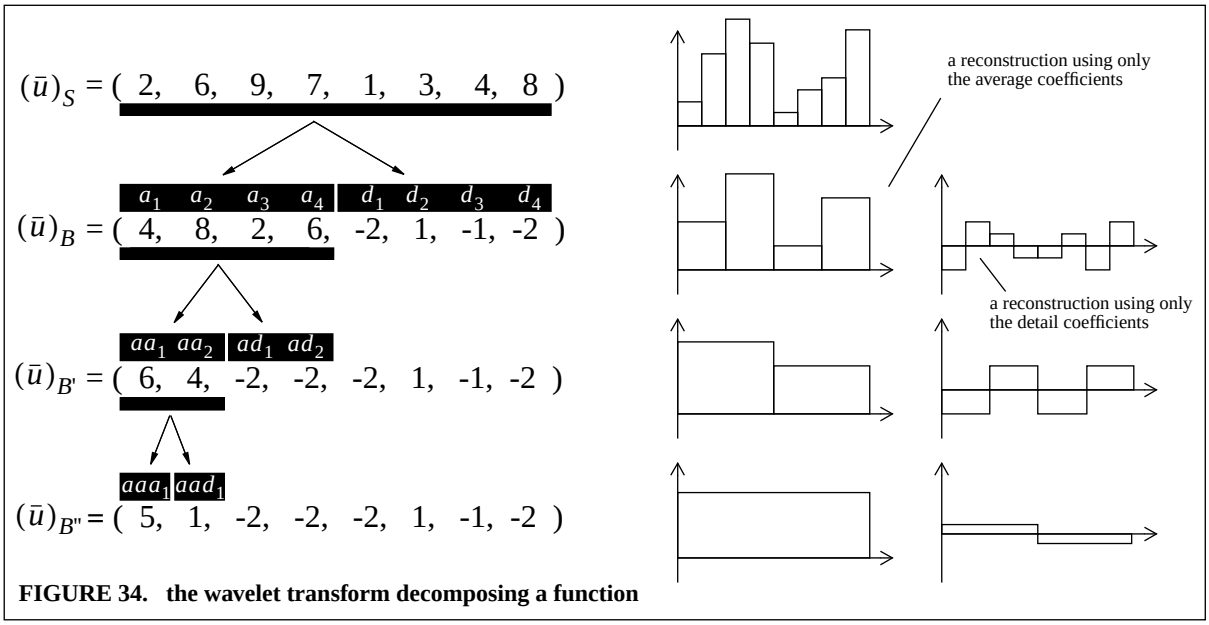


FIGURE 34. the wavelet transform decomposing a function

$(\bar{u})_{B''} = (5, 1, -2, -2, -2, 1, -1, -2)$ regarding to the basis B''. The different bases of each decomposition step are shown in figure 33.

The wavelet transform of the original vector $(\bar{u})_S$ is defined as the average coefficients of the lowest resolution and the detail coefficients in order of increasing resolution. Therefore the coordinate vector $(\bar{u})_{B''} = (5, 1, -2, -2, -2, 1, -1, -2)$ is the wavelet transform of the vector $(\bar{u})_S$. We better say that $(\bar{u})_{B''}$ is the full wavelet transform of $(\bar{u})_S$ down to the bottom most possible level of decomposition. The wavelet decomposition can stop at any resolution level. The wavelet transform $(\bar{u})_{B'} = (6, 4, -2, -2, -2, 1, -1, -2)$ omits the final step of the full wavelet decomposition and the wavelet transform $(\bar{u})_B = (4, 8, 2, 6, -2, 1, -1, -2)$ applies just one decomposition step to the original vector $(\bar{u})_S$.

One way to 'compress' the wavelet representation of the function by omitting the smallest coefficients beneath a certain threshold. This means simply to set them to zero. This 'compression' is lossy since the function reconstructed from the 'compressed' wavelet representation differs from the original function. The higher the rate of compression, meaning more coefficients are set to zero, the bigger is the difference between the original and the 'compressed' function. This trade-off is depicted graphically in figure 35.

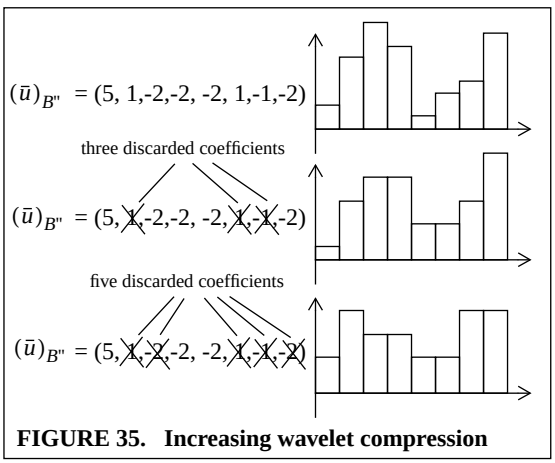


FIGURE 35. Increasing wavelet compression

1.5 The mother scaling function $\phi(x)$ and the average spaces V_j

A discrete orthogonal multiresolution analysis (MRA) of R^n is an increasing sequence of closed subspaces V_j of R^n which approximate R^n from coarse to fine. For the discrete MRA n has to be a power of two.
The following properties describe a discrete multiresolution analysis:

1. $V_0 \subset V_1 \subset V_2 \subset \ldots\ldots \subset V_{\log_2 n}$
2. $v(x) \in V_j \Leftrightarrow v(2x) \in V_{j+1}$
3. $v(x) \in V_j \Leftrightarrow v(2x-1) \in V_{j+1}$
4. A mother scaling function $\phi(x)$ with a non vanishing integral exists that is an orthonormal basis for V_0.

Since $\phi(x) \in V_0 \subset V_1$ a sequence $\{h_k\}$ exists such that the mother scaling function $\phi(x)$ satisfies a refinement equation:

$$\phi(x) = 2\sum_k h_k \phi(2x-k) \qquad (1)$$

The following equations can be proofed:

$$\sum_k h_k = 1 \qquad \int \phi(x)dx = 1$$

It is immediate that the functions $\phi(2x)$ and $\phi(2x-1)$ form an orthogonal basis for V_1 and more general that the collection of functions $\{\phi(2^j x-k) \mid 0 < k < 2^j - 1\}$ is an orthogonal basis of V_j. Normalized to an orthonormal basis for V_j the basis functions are $\{\phi_{j,k}(x) \mid 0 < k < 2^j - 1\}$ where the scaling functions $\phi_{j,k}(x)$ are dilated, translated and normalized versions of the mother scaling function $\phi(x)$:

$$\phi_{j,k}(x) = \sqrt{2^j}\phi(2^j x-k)$$

together with (1) follows

$$\phi(x) = \phi_{0,0}(x) = \sqrt{2}\sum_m h_m \phi_{1,m}(x)$$

and more general

$$\phi_{j,k}(x) = \sqrt{2}\sum_m h_m \phi_{j+1,m+2k}(x)$$

1.6 The mother wavelet $\psi(x)$ and the detail spaces W_i

We define W_i to be the orthogonal complement of V_i in V_{i+1}. Hence W_i is also a subspace of V_{i+1} and satisfies:

$$V_{i+1} = V_i \oplus W_i$$

where the symbol $\oplus$ stands for the direct sum. In other words, W_i has the missing details to go from V_i to V_{i+1}.
The properties that held for average spaces have to be rewritten for the detail spaces:

1. $W_i \subset V_{i+1}$
2. $W_0 \not\subset W_1 \not\subset W_2 \not\subset \ldots\ldots \not\subset W_{(\log_2 n)-1}$
3. $W_{(\log_2 n)-1} \not\subset \ldots\ldots \not\subset W_2 \not\subset W_1 \not\subset W_0$
4. $w(x) \in W_i \Leftrightarrow w(2x) \in V_{i+1}$

5. $w(x) \in W_i \Leftrightarrow w(2x-1) \in V_{i+1}$
6. A mother wavelet $\psi(x)$, with a vanishing integral, exists that is an orthonormal basis for W_0.

Since $\psi(x) \in W_0 \subset V_1$ a sequence $\{g_k\}$ exists such that the mother wavelet $\psi(x)$ satisfies a refinement equation:

$$\psi(x) = 2\sum_k g_k \phi(2x-k)$$

The following equations can be proofed:

$$\sum_k g_k = 0 \qquad\qquad \int \psi(x)dx = 0$$

For orthogonal scaling functions and wavelets the n coefficients g_k are connected to the coefficients h_k through the equation $g_k = (-1)^k h_{n-k}$. It is immediate that the functions $\psi(2x)$ and $\psi(2x-1)$ form an orthogonal basis for W_1 and more general that the collection of functions $\{\psi(2^j x-k) \mid 0 < k < 2^j - 1\}$ is an orthogonal basis of W_j. Normalized to an orthonormal basis for W_j the basis functions are $\{\psi_{j,k}(x) \mid 0 < k < 2^j - 1\}$ where the wavelets $\psi_{j,k}(x)$ are dilated, translated and normalized versions of the mother wavelet $\psi(x)$:

$$\psi_{j,k}(x) = \sqrt{2^j}\psi(2^j x-k)$$

1.7 The fast wavelet transform

Any function[1] $v_j(x) \in V_j$ can be written as the linear combination of the scaling functions $\phi_{j,k}(x)$ and any function $w_j(x) \in W_j$ can be written as the linear combination of the wavelets $\psi_{j,k}(x)$. Since V_{j+1} is equal to $V_{j+1} = V_j \oplus W_j$ any function $v_{j+1}(x) \in V_{j+1}$ can be written uniquely as the sum of a function $v_j(x) \in V_j$ and a function $w_j(x) \in W_j$:

$$v_{j+1}(x) = v_j(x) + w_j(x)$$

$$\sum_{k=0}^{2^{j+1}-1} \alpha_{j+1,k}\phi_{j+1,k}(x) = \sum_{k=0}^{2^j-1} \alpha_{j,k}\phi_{j,k}(x) + \sum_{k=0}^{2^j-1} \beta_{j,k}\psi_{j,k}(x)$$

In other words, we have two representations of the function $v_{j+1}(x)$, one as an element in V_{j+1} and associated with the sequence $\{\alpha_{j+1,k}\}$, and another as a sum of elements in V_j and W_j associated with the sequences $\{\alpha_{j,k}\}$ and $\{\beta_{j,k}\}$. The following relations show how to pass between these representations:

$$\alpha_{j,k} = \langle v_{j+1} | \phi_{j,k}(x) \rangle = \langle v_{j+1} | \sqrt{2}\sum_m h_m \phi_{j+1,m+2k}(x) \rangle = \sqrt{2}\sum_m h_m \alpha_{j+1,m+2k} \qquad (2)$$

1. for the discrete case these are vectors $\overline{v_j} \in V_j$ of the vector space R^n

and similar.

$$\beta_{j,k} = \langle v_{j+1} | \psi_{j,k}(x) \rangle = \langle v_{j+1} | \sqrt{2} \sum_m g_m \phi_{j+1,m+2k}(x) \rangle = \sqrt{2} \sum_m g_m \alpha_{j+1,m+2k} \qquad (3)$$

The opposite direction from the $\alpha_{j,k}$ and the $\beta_{j,k}$ to the $\alpha_{j+1,k}$ is equally easy:

$$\alpha_{j+1,k} = \sqrt{2} \sum_m h_m \alpha_{j,m} + \sqrt{2} \sum_m g_m \beta_{j,m} \qquad (4)$$

When applied recursively, these formulae define the fast wavelet transform. The relation (2) and (3) define the forward transform, while (4) defines the inverse transform. The scheme of decomposition and reconstruction are illustrated graphically in figure 36 and in figure 37.

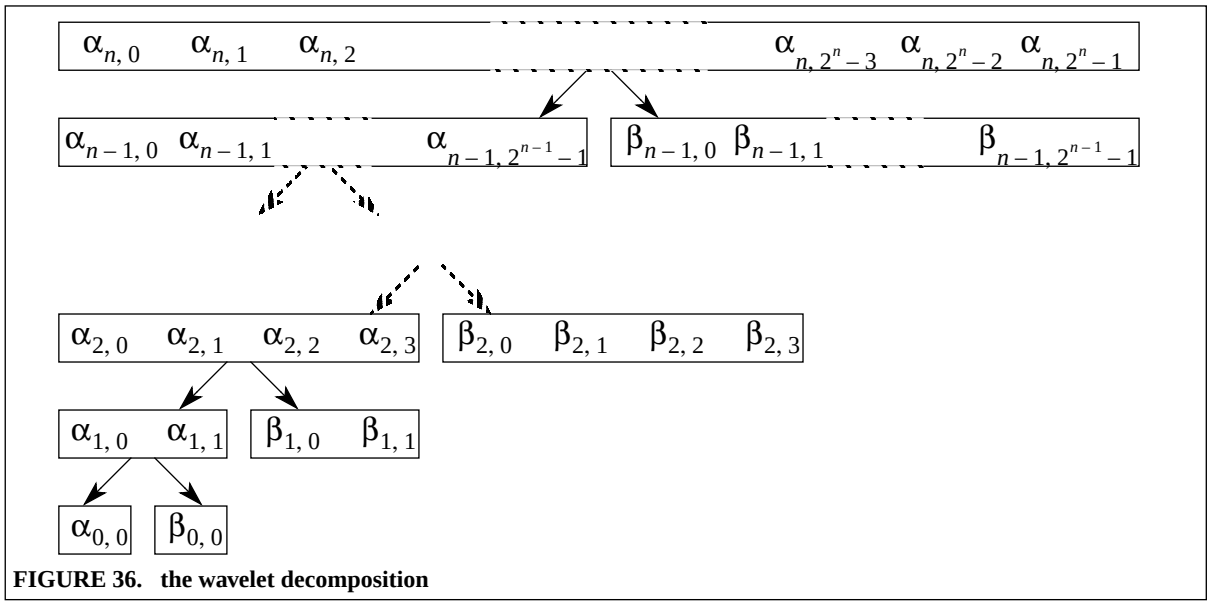


FIGURE 36. the wavelet decomposition

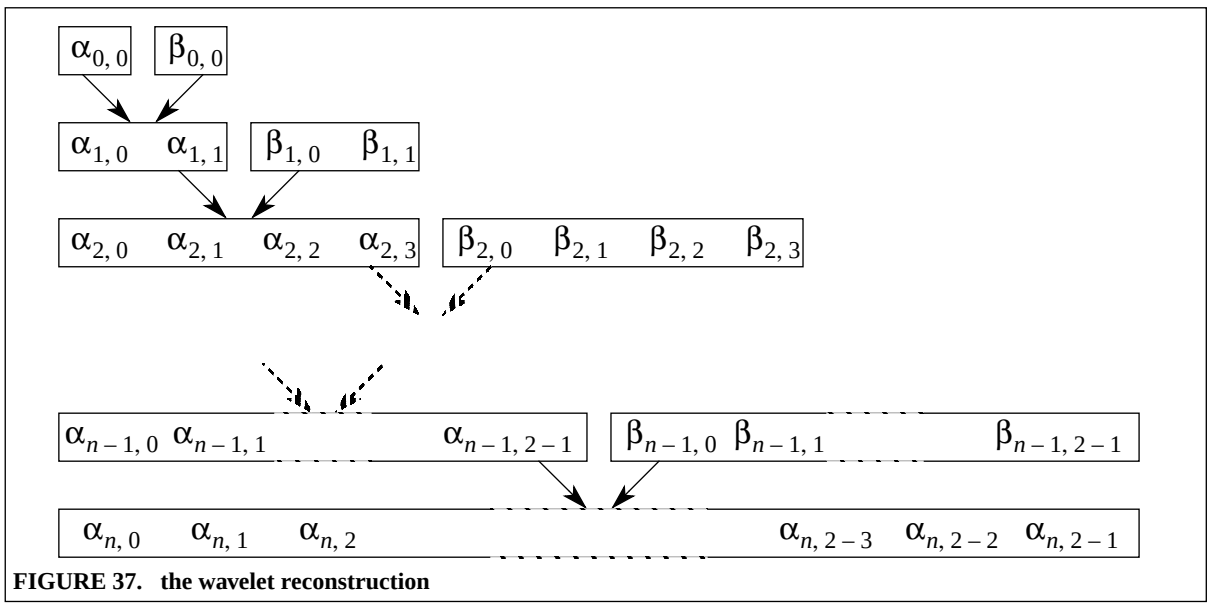


FIGURE 37. the wavelet reconstruction

1.8 Computing the discrete wavelet transform

Let the original function be represented by n sample values, that is a vector $\bar{u}$ of R^n. In the view of linear algebra the DWT is an basis transformation from the standard basis S to a wavelet basis W. Any basis transformation in R^n can be realized with a transformation matrix D. The wavelet decomposition $\bar{u}$ could be calculated with $[\bar{u}]_W = D[\bar{u}]_S$, where $[\bar{u}]_S$ is the coordinate matrix of $\bar{u}$ relative to S and $[\bar{u}]_W$ is the coordinate matrix of $\bar{u}$ relative to W. The inverse DWT that restores the original representation $[\bar{u}]_S$ from the wavelet decomposition $[\bar{u}]_W$ is done with the inverse of the transformation matrix. Because the DWT is usually a normalized orthogonal transform, the transformation matrix D is orthonormal and its inverse D^{-1} is its transpose D^T. The wavelet composition $[\bar{u}]_S$ could be calculated with $[\bar{u}]_S = D^T[\bar{u}]_W$. This would be a computational expensive way to compute the DWT. The transform of a vector of the length n requires n^2 multiplications and n^2 summations resulting in a complexity of $O(n^2)$.

Now recall the introductory example given in the first section of this chapter. It said that the wavelet transform is done in several iterations and that each iteration is an orthogonal basis transformation. The basis transformation matrix for each of those iterations is a sparse matrix. We carry on with this example and show how each iteration is done using a sparse transformation matrix.

It is immediate that the first transformation matrix transforms the vector $\bar{u}$ that represents the sampled functions into the coordinate vector $(\bar{u})_B$. The transformation matrix transforms from the standard basis S to the basis B. The second iteration uses a matrix that transforms from the basis B to another basis B'.

$$[\bar{u}]_B = D[\bar{u}]_S = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & -0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 9 \\ 7 \\ 1 \\ 3 \\ 4 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 8 \\ 2 \\ 6 \\ -2 \\ 1 \\ -1 \\ -2 \end{bmatrix}$$

$$[\bar{u}]_{B'} = D'[\bar{u}]_B = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 8 \\ 2 \\ 6 \\ -2 \\ 1 \\ -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ -2 \\ -2 \\ -2 \\ 1 \\ -1 \\ -2 \end{bmatrix}$$

$$[\bar{u}]_{B''} = D''[\bar{u}]_{B'} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 4 \\ -2 \\ -2 \\ -2 \\ 1 \\ -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ -2 \\ -2 \\ -2 \\ 1 \\ -1 \\ -2 \end{bmatrix}$$

The last iteration results into the final wavelet decomposition of the vector $\bar{u}$. It is a basis transformation from the basis B' to the basis B''. Obviously we could have done these three basis transformation in one step.
If the matrix D transforms from the basis S to B, the matrix D' from the basis B to the basis

$$[\bar{u}]_{B''} = D''D'D[\bar{u}]_S = D'''[\bar{u}]_S = \begin{bmatrix} 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & -0.125 & -0.125 & -0.125 & -0.125 \\ 0.25 & 0.25 & -0.25 & -0.25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & 0.25 & -0.25 & -0.25 \\ 0.5 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & -0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 6 \\ 9 \\ 7 \\ 1 \\ 3 \\ 4 \\ 8 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ -2 \\ -2 \\ -2 \\ 1 \\ -1 \\ -2 \end{bmatrix}$$

B' and the transformation from B' to the basis B'' is done by the matrix D'', then the matrix $D''' = D''D'D$ transforms the vector $\bar{u}$ directly to its representation in regard to the basis B''.

Ingrid Daubechies has discovered that the wavelet transform can be implemented with a specially designed pair of Finite Impulse Response (FIR) filters called Quadrature Mirror Filter (QMF), see “Digital filters” on page 19. A FIR filter performs the dot product between the filter coefficients and the discrete samples in the tapped delay line of the filter. The act of passing a set of discrete samples, representing a signal, through a FIR filter is a discrete convolution of the signal with the filters coefficients [10]. Then the wavelet decomposition can be done with the complexity of O(n) as depicted in figure 38. The averaging as we called it earlier corresponds to the low-pass filtering and respectively the differenzing corresponds to the high-pass filtering. In this example the length of each of the two FIR filters is two.

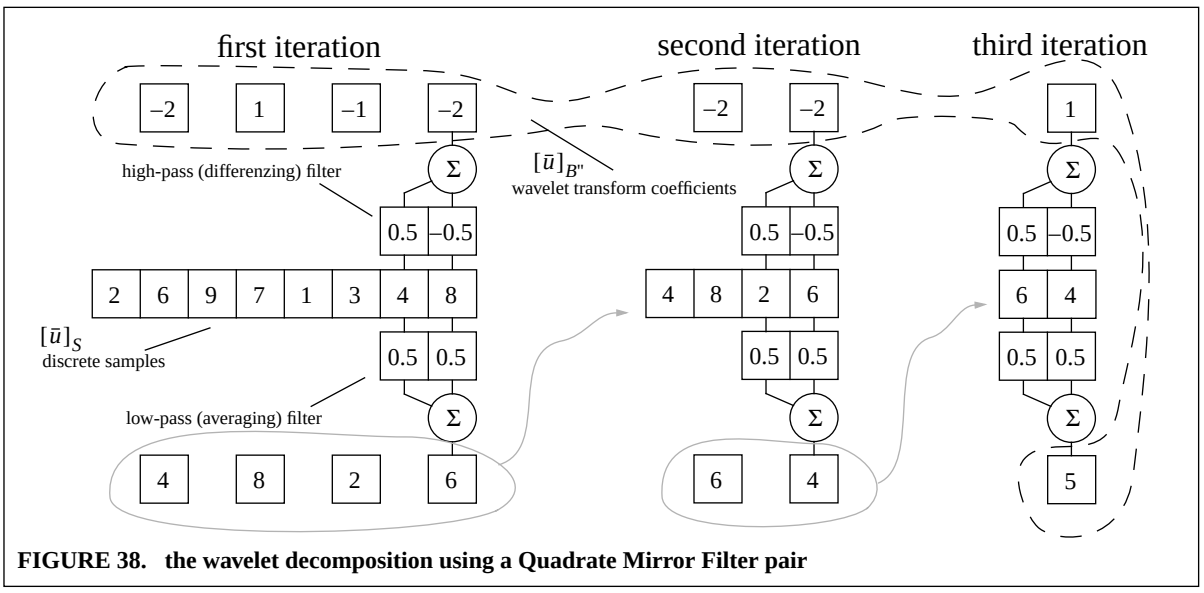


FIGURE 38. the wavelet decomposition using a Quadrate Mirror Filter pair

2 The Haar wavelet

The simplest wavelet basis that was already used in 1930 by Paul Levy is called the Haar wavelet. He found it superior to the Fourier basis for studying small complicated details in the Brownian motion. In the introductory example we already used the Haar wavelet transform, although - for the sake of clarity - the basis vectors were not normalized.

The Haar wavelet transform has a very good time resolution, however, its resolution in the other domain - the frequency domain - is very poor.

2.1 The Haar scaling function

The Haar mother scaling function is defined by

$$\phi(x) = \begin{cases} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

and satisfies the refinement equation

$$\phi(x) = 2\sum_k h_k \phi(2x-k) = 2\left(\frac{1}{2}\phi(2x) + \frac{1}{2}\phi(2x-1)\right)$$

for $h_0 = \frac{1}{2}$ and $h_1 = \frac{1}{2}$. Resulting in the normalized filter coefficients:

$$c_0 = \sqrt{2}h_0 = \frac{1}{\sqrt{2}} \qquad c_1 = \sqrt{2}h_1 = \frac{1}{\sqrt{2}}$$

The resulting FIR filter that is used to compute one half of the wavelet transform has a frequency response as depicted in figure 39. Obviously its localization in frequency is not good. Together with the FIR filter for the Haar wavelet it makes a Quadrature Mirror Filter pair.

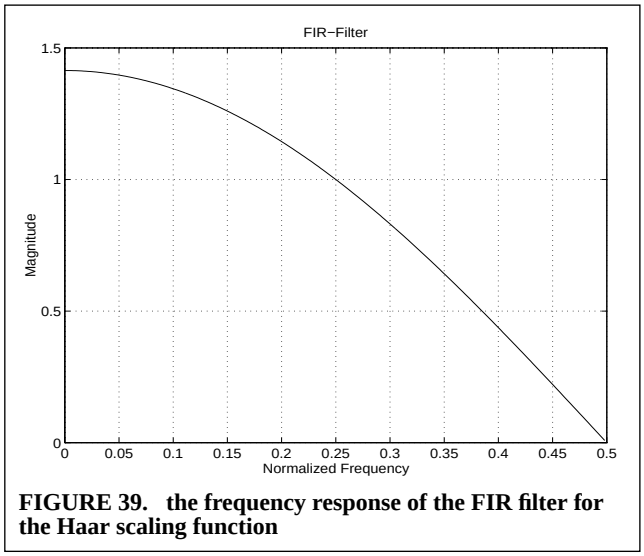


FIGURE 39. the frequency response of the FIR filter for the Haar scaling function

Some dilated and translated versions of the Haar mother scaling function are shown in figure 40. Note that for the sake of clarity the basis functions have not been normalized. In case the Haar wavelet transform is actually applied to a n-dimensional vector that represents a function sampled at n points, the Haar scaling functions are n-dimensional vectors. They are the basis vectors that span the spaces V_0 to $V_{\log_2 n}$. Given an 8-dimensional vector the spaces range between V_0 and V_3. The corresponding Haar scaling functions that span each of these spaces

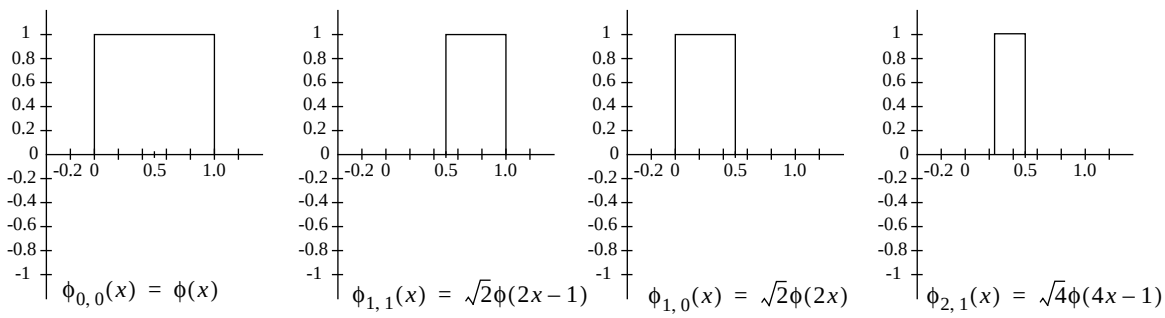


FIGURE 40. different Haar scaling functions

are shown in figure 41 on page 51.

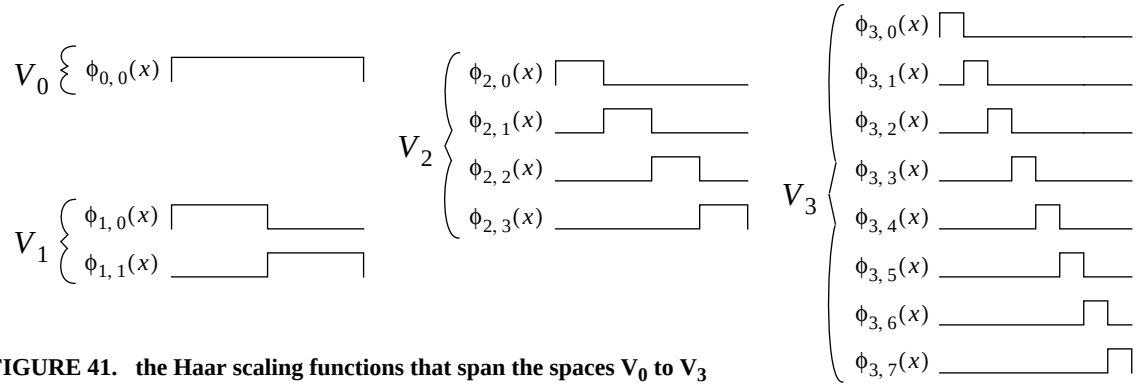


FIGURE 41. the Haar scaling functions that span the spaces V_0 to V_3

2.2 The Haar wavelet

The Haar mother wavelet is defined by

$$\psi(x) = \begin{cases} 1 \text{ for } 0 \le x < \frac{1}{2} \\ -1 \text{ for } \frac{1}{2} \le x < 1 \\ 0 \text{ otherwise} \end{cases}$$

and satisfies the refinement equation

$$\psi(x) = 2\sum_k g_k \phi(2x-k) = 2\left(\frac{1}{2}\phi(2x) - \frac{1}{2}\phi(2x-1)\right)$$

for $g_0 = \frac{1}{2}$ and $g_1 = -\frac{1}{2}$. Resulting in the normalized filter coefficients:

$$c_0 = \sqrt{2}g_0 = \frac{1}{\sqrt{2}} \qquad c_1 = \sqrt{2}g_1 = -\frac{1}{\sqrt{2}}$$

The resulting FIR filter for the Haar wavelet has a frequency response as depicted in figure 42. What held for the FIR filter for the Haar scaling function is also true here. The localization in frequency is not good.
Some dilated and translated versions of the Haar mother wavelet are shown in figure 43. Note that for the sake of clarity the basis functions have not been normalized. Again, in case the Haar wavelet transform is actually applied to a n-dimensional vector that represents a function sampled at n points, the Haar wavelets are n-dimensional vectors. They are

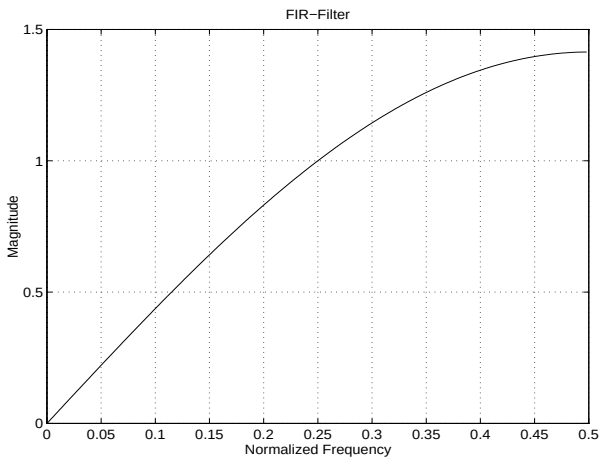


FIGURE 42. the frequency response of the FIR filter for the Haar wavelet

the basis vectors that span the spaces W_0 to $W_{(\log_2 n)-1}$. Given an 8-dimensional vector the

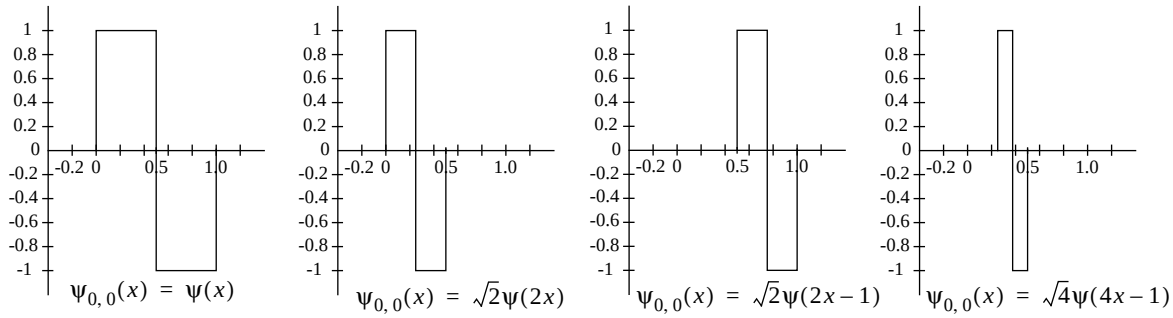


FIGURE 43. different Haar wavelets

spaces range between W_0 and W_2. The corresponding Haar wavelets that span each of these spaces are shown in figure 44.

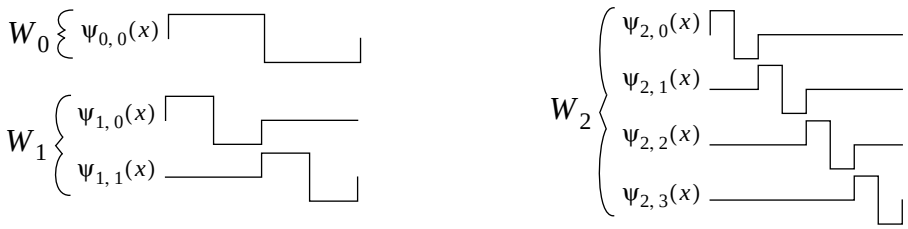


FIGURE 44. the Haar wavelets that span the spaces W_0 to W_2

3 The Daubechies wavelet

Until Daubechies' ground breaking work the Haar wavelet was thought to be the only wavelet to have compact support (a finite number of filter coefficients) and to form an orthonormal multiresolution analysis. The Daubechies wavelets build a whole family of wavelets. They are distinguished by an index that equals the number of filter coefficients for decomposition and reconstruction. At this point we will introduce the Daubechies4 scaling function and wavelet since they are realized with a Quadrature Mirror Filter of the shortest length.

3.1 The Daubechies scaling function

There is no closed form for the Daubechies mother scaling function. However a recursive display algorithm can be used to draw the graph of it [43]. The Daubechies mother scaling function satisfies the refinement equation

$$\phi(x) = 2\sum_k h_k \phi(2x-k)$$

$$= 2\left(\frac{1+\sqrt{3}}{8}\phi(2x) + \frac{3+\sqrt{3}}{8}\phi(2x-1) + \frac{3-\sqrt{3}}{8}\phi(2x-1) + \frac{1-\sqrt{3}}{8}\phi(2x-1)\right)$$

for $h_0 = \frac{1+\sqrt{3}}{8}$. $h_1 = \frac{3+\sqrt{3}}{8}$, $h_2 = \frac{3-\sqrt{3}}{8}$ and $h_3 = \frac{1-\sqrt{3}}{8}$. Resulting in the normalized filter coefficients:

$$c_0 = \sqrt{2}h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \qquad c_1 = \sqrt{2}h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \qquad c_2 = \sqrt{2}h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$$

$$c_3 = \sqrt{2}h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

The resulting FIR filter that is used to compute one half of the wavelet transform has a frequency response as depicted in figure 45. Obviously its localization in frequency is already better than that of the Haar scaling function. Together with the FIR filter for the Daubechies wavelet it makes a Quadrature Mirror Filter pair.

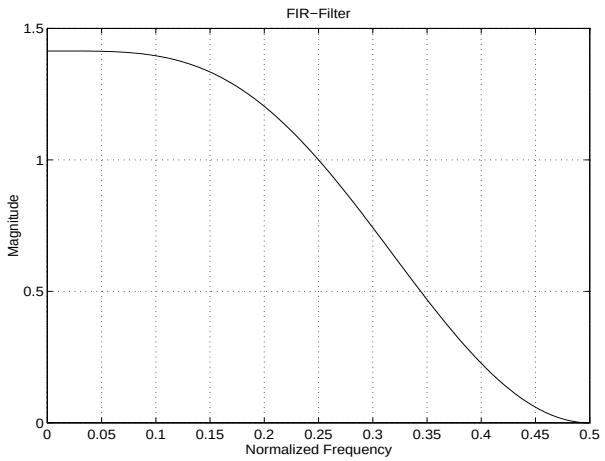


FIGURE 45. the frequency response of the FIR filter for the Daubechies scaling function

Some dilated and translated versions of the Daubechies mother scaling function are shown in figure 46. As we said, the Daubechies scaling function is defined by a recursive algorithm. The depicted versions of the scaling function are calculated on an interval of 2048 points. The mother scaling function $\phi_{0,0}(x)$ covers the complete interval in the same way the mother Haar scaling function does it. The same that holds for the Haar wavelet transform is valid for the Daubechies wavelet transform. Applied to a n-dimensional vector that represents a function sampled at n points, the Daubechies scaling functions are n-dimensional vectors. They are the basis vectors that span the spaces V_0 to $V_{\log_2 n}$. Given an 8-dimensional vector the spaces range between V_0 and V_3. The corresponding Daubechies scaling functions that span each of these spaces are shown in figure 47 on page 54.

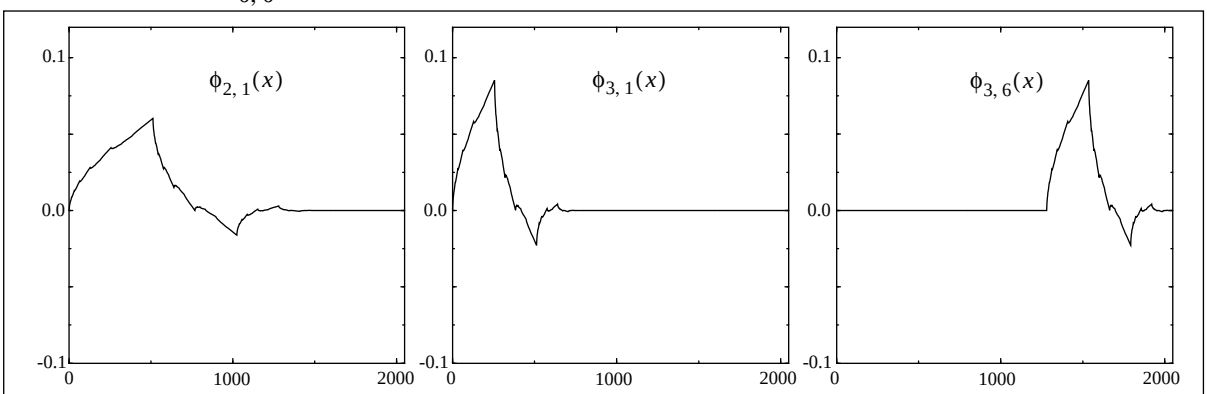


FIGURE 46. different Daubechies scaling functions

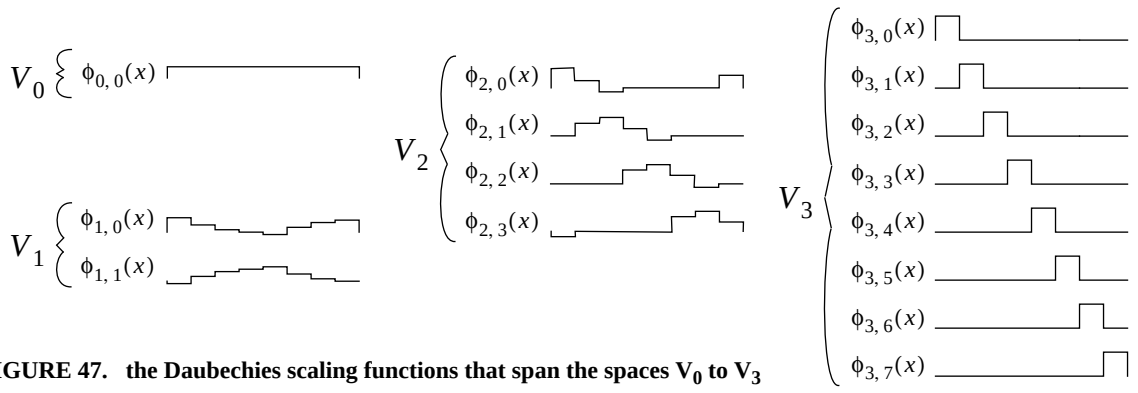


FIGURE 47. the Daubechies scaling functions that span the spaces V_0 to V_3

3.2 The Daubechies wavelet

There is consequently also no closed form for the Daubechies mother wavelet. In the same manner a recursive algorithm is used to calculate it. The Daubechies mother wavelet satisfies the refinement equation

$$\psi(x) = 2\sum_k h_k \phi(2x-k)$$

$$= 2\left(\frac{1+\sqrt{3}}{8}\psi(2x) + \frac{3+\sqrt{3}}{8}\psi(2x-1) + \frac{3-\sqrt{3}}{8}\psi(2x-1) + \frac{1-\sqrt{3}}{8}\psi(2x-1)\right)$$

for $g_0 = \frac{1-\sqrt{3}}{8}$. $g_1 = -\frac{3-\sqrt{3}}{8}$, $g_2 = \frac{3+\sqrt{3}}{8}$ and $g_3 = -\frac{1+\sqrt{3}}{8}$. Resulting in the normalized filter coefficients:

$$c_0 = \sqrt{2}g_0 = \frac{1-\sqrt{3}}{4\sqrt{2}} \qquad c_1 = \sqrt{2}g_1 = -\frac{3-\sqrt{3}}{4\sqrt{2}} \qquad c_2 = \sqrt{2}g_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}$$

$$c_3 = \sqrt{2}h_3 = -\frac{1+\sqrt{3}}{4\sqrt{2}}$$

The resulting FIR filter for the Daubechies wavelet has a frequency response as depicted in figure 48. What held for the FIR filter for the Daubechies scaling function is also true here. The localization in frequency is already better than the frequency localization of the Haar wavelet.

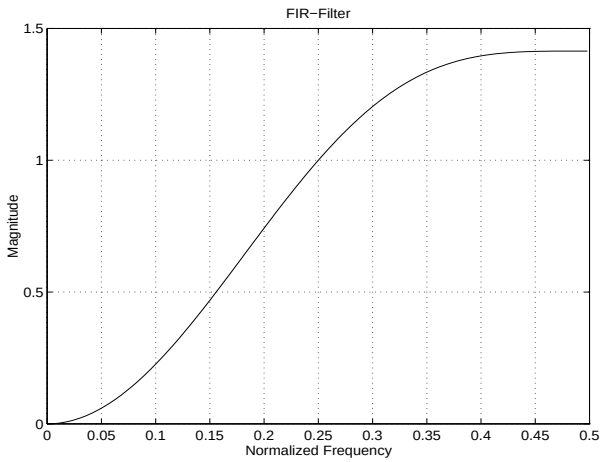


FIGURE 48. the frequency response of the FIR filter for the Daubechies wavelet

Some dilated and translated versions of the Daubechies mother wavelet are shown in figure 49. The depicted versions of the wavelet are again calculated on an interval of 2048 points. It is immediate that mother wavelet $\psi_{0,0}(x)$ covers the complete interval as well. On applying the

Daubechies wavelet transform to a n-dimensional vector that represents a function sampled at

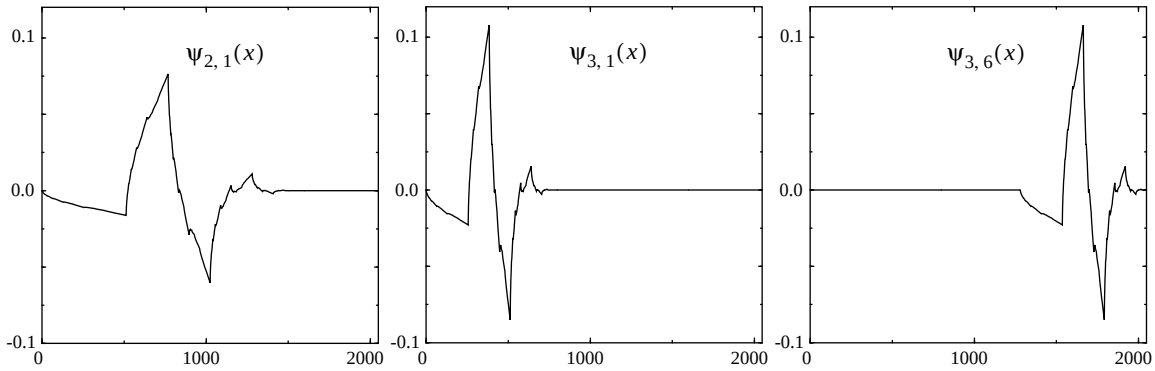


FIGURE 49. different Daubechies wavelets

n points, the Daubechies wavelets are n-dimensional vectors. They are the basis vectors that span the spaces W_0 to $W_{(\log_2 n)-1}$. Given an 8-dimensional vector the spaces range between W_0 and W_2. The corresponding Daubechies wavelets that span each of these spaces are shown

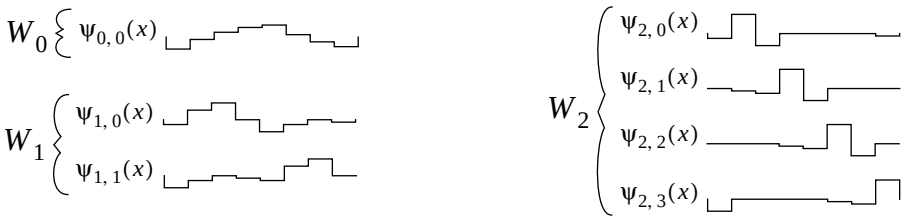


FIGURE 50. the Daubechies wavelets that span the spaces W_0 to W_2

in figure 50.

3.3 Other Daubechies wavelets and scaling functions

As we said the Daubechies wavelets build a whole family of wavelets. In this section we want to take a brief look how other members of this family look like and what the differences between the members are. The Daubechies4 wavelet is the ‘shortest’ wavelet in the family meaning that it has the least number of FIR filter coefficients - exactly four. Actually the Haar wavelet is sometimes referred to be the smallest member of this family. In this case it is called Daubechies2 wavelet - obviously it has only two filter coefficients. The Daubechies6, Daubechies8, Daubechies10, ... and so on consequently have FIR filters with 6, 8, 10, ... and so on coefficients respectively. The frequency localization of the Daubechies4 wavelet was

Haar	0.70710678, 0.70710678
Daubechies4	0.48296291, 0.83651630, 0.22414386, -0.12940952
Daubechies6	0.33267055, 0.80689150, 0.45987750, -0.13501102, -0.08544127, 0.03522629
Daubechies8	0.23037781, 0.71484657, 0.63088076, -0.02798376, -0.18703481, 0.03084138, 0.03288301, -0.01059740
Daubechies10	0.16010239, 0.60382926, 0.72430852, 0.13842814, -0.24229488, -0.03224486, 0.07757149, -0.00624149, -0.01258075, 0.00333572
Daubechies12	0.11154074, 0.49462389, 0.75113390, 0.31525035, -0.22626469, -0.12976686, 0.09750160, 0.02752286, -0.03158203, 0.00055384, 0.00477725, -0.00107730

TABLE 5. FIR filter coefficients for the Daubechies family

already better than that of the Haar wavelet. We will see that the localization in frequency increases with the length of the filter. Therefore the ‘longer’ wavelets of the Daubechies family have a better localization in this domain. But it is immediate that with increasing filter length the localization in the time domain becomes poorer. The Daubechies8 wavelet and scaling

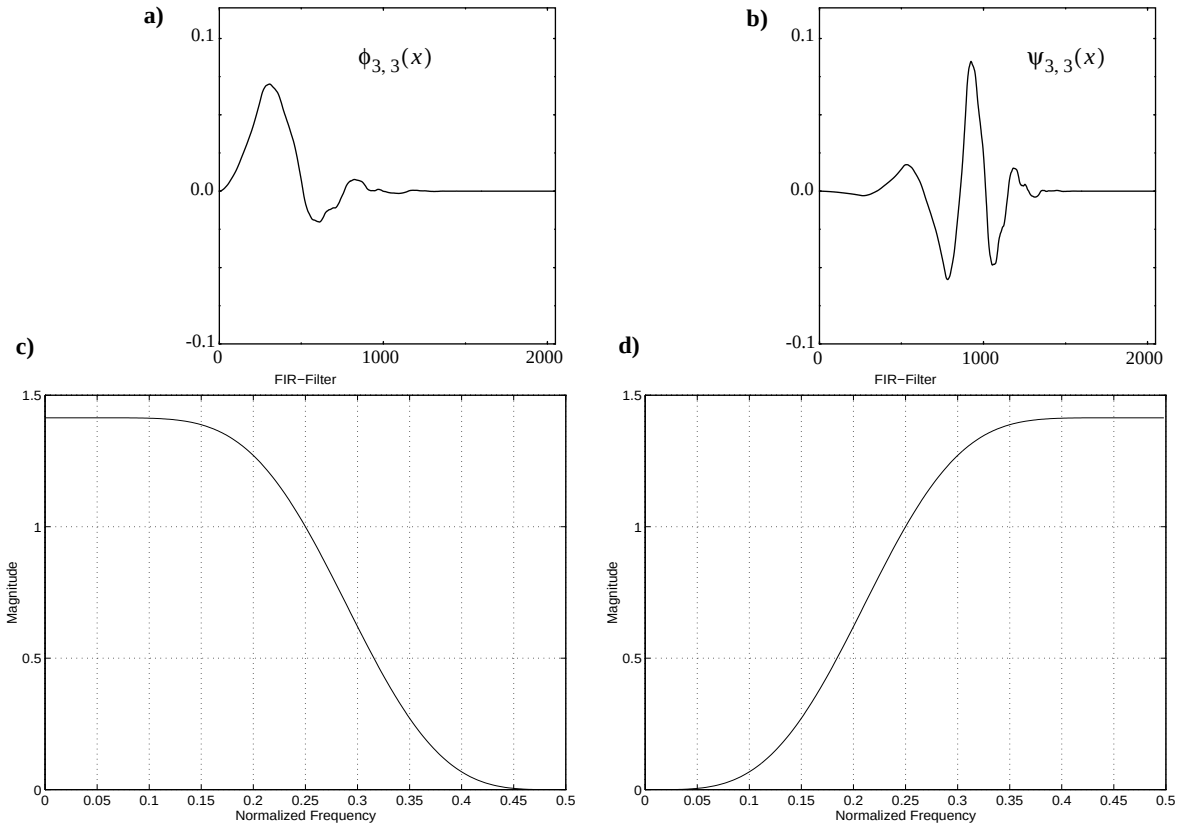


FIGURE 51. (a) some Daubechies8 scaling function, (b) some Daubechies8 wavelet, (c) the frequency response of the low-pass FIR filter, (d) the frequency response of the high-pass FIR filter

function are given in figure 51. There are also the frequency responses of the corresponding low-pass and high-pass FIR filter illustrated. As said earlier the localization in the frequency domain improves with the length of the filter. The two filters do a more accurate job in splitting the signal into two subbands.

The frequency response and the immediate increase of frequency localization is summarized in figure 52 on page 57.

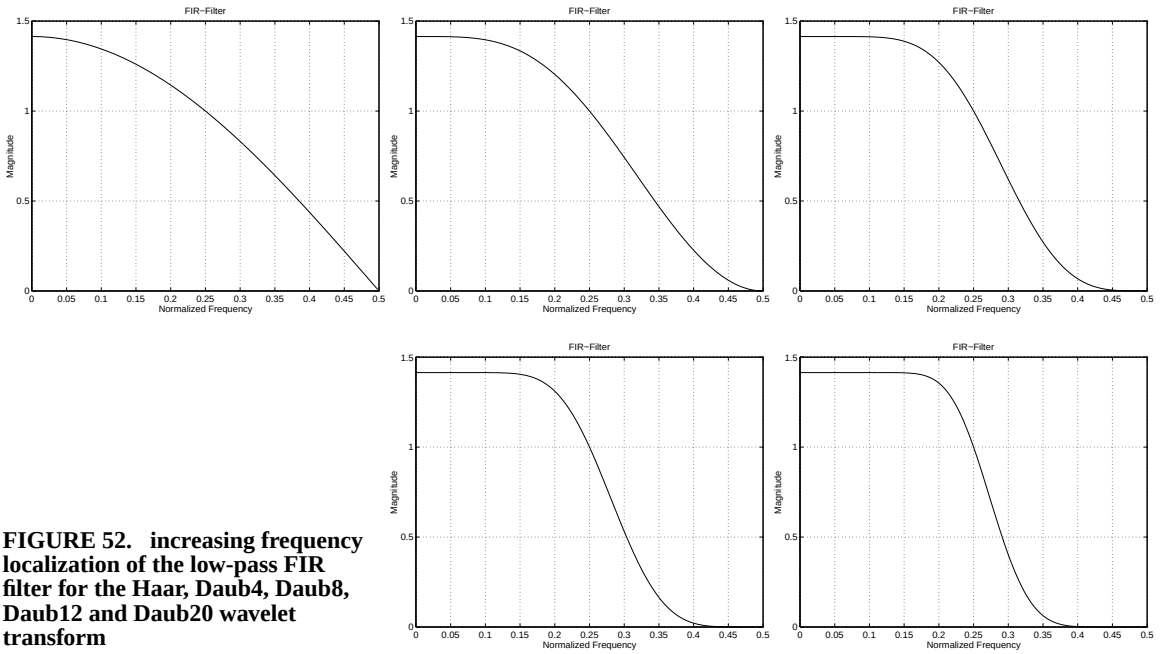


FIGURE 52. increasing frequency localization of the low-pass FIR filter for the Haar, Daub4, Daub8, Daub12 and Daub20 wavelet transform

We have now examined the frequency response of the FIR filters used for the wavelet transform in detail. One has to keep in mind that the wavelet transform applies those FIR filters recursively onto the signal. The transform coefficients that are result from the first convolution of the signal samples with the filters coefficients address basis vectors with a frequency localization equal to the frequency response of the FIR filters. Iteratively the FIR filters are applied towards the previous low-pass coefficients. The basis vectors addressed by the resulting coefficients of the second convolution have a more precise localization in frequency but therefore lose their time localization. Illustrated by the corresponding information cells in the time-frequency plane (see figure 53) with every decomposition step the uncertainty in frequency is halved while the uncertainty in time is doubled.

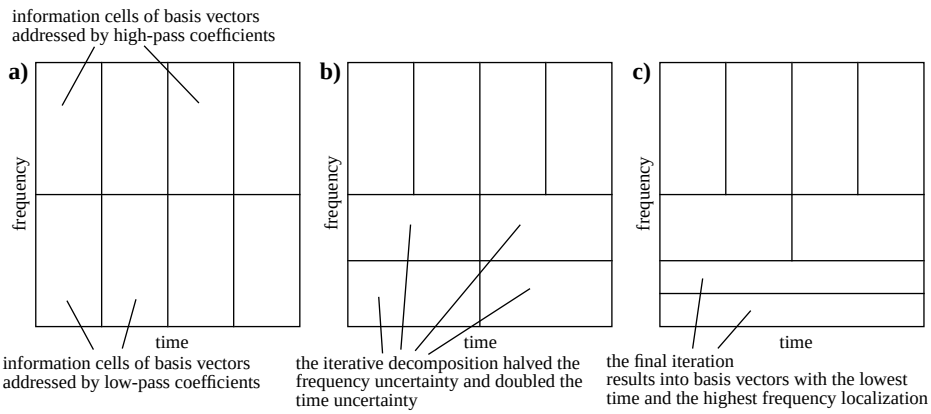


FIGURE 53. the idealized tiling of the time-frequency plane by the basis vectors of the wavelet transform after one (a), two (b) and three iterative decomposition steps

4 The wave packet transform

It turns out that the discrete wavelet transform is actually a subset of a far more versatile transform, the wave packet transform (WPT). Developed by Ronald Coifman of Yale University, the wave packet transform generalizes the time-frequency analysis of the wavelet transform. It yields into a family of orthonormal transform basis where the wavelet transform is only one member [11].

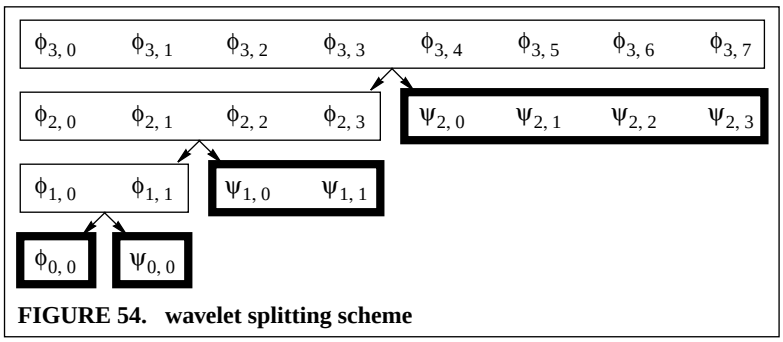


FIGURE 54. wavelet splitting scheme

We see that the classical multiresolution analysis is obtained by splitting a space V_j into two orthogonal subspaces V_{j-1} and W_{j-1}. and then doing the same for the space V_{j-1} recursively [25]. At each step of the recursion the subspaces V_i are spanned by scaling functions $\phi_{i,k}$ and the subspaces W_i are spanned by wavelets $\psi_{i,k}$. And at each step of the recursion the direct sum $V_i \oplus W_i \oplus W_{i+1} \oplus \ldots \oplus W_{j-1}$ forms an orthonormal basis for the space V_j. The corresponding orthonormal basis vectors are the scaling functions of the subspace V_i and the wavelets of the subspaces W_i to W_{j-1}. In figure 54 on page 58 we give a schematic representation of a space and its subspaces after using splitting over three recursions. The top rectangle represents the space V_3 and each other rectangle corresponds to a certain subspace of V_3. The splitting is done by a Quadrature Mirror Filter pair. The slanted lines between the rectangles indicate the splitting. The left refers to the low-pass filter and the right refers to the high-pass filter. The wavelet decomposition can now be viewed as a partial graph of a binary tree. In the idealized time-frequency plane the wavelet transform basis can be illustrated as in figure 55 on page 58.

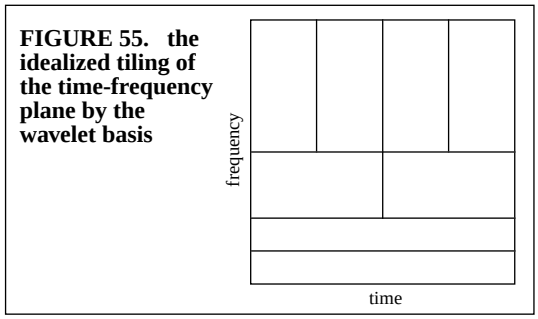


FIGURE 55. the idealized tiling of the time-frequency plane by the wavelet basis

The wave packets are the basis functions that we obtain if we also split the W_i spaces. So starting from a space V_j the wave packet decomposition can be represented as a full binary tree as shown in figure 56. Each rectangle is a direct sum of the two subspaces denoted by its children. The bold rectangles then correspond to the wavelet multiresolution analysis. This wavelet transform basis is actually a subset of as family of bases formed by the wave packet transform.

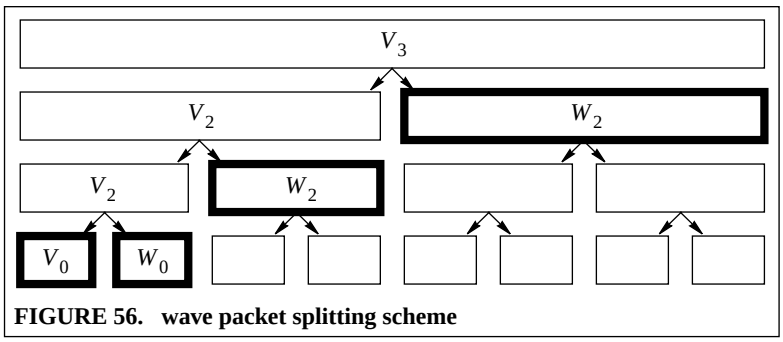


FIGURE 56. wave packet splitting scheme

Any disjunct cover of rectangles forms an orthonormal basis for the space V_j. In figure 57 two example bases are shown. The first one is a so called subband basis since it splits the decomposed function in four equal spaces frequency bands. All basis vectors within a rectangle have the same frequency response. Since each splitting iteration divides the input signal through low-pass and high-pass filtering into two frequency bands, n recursively applied iterations to the signal result into a subband representation with 2^n frequency bands.

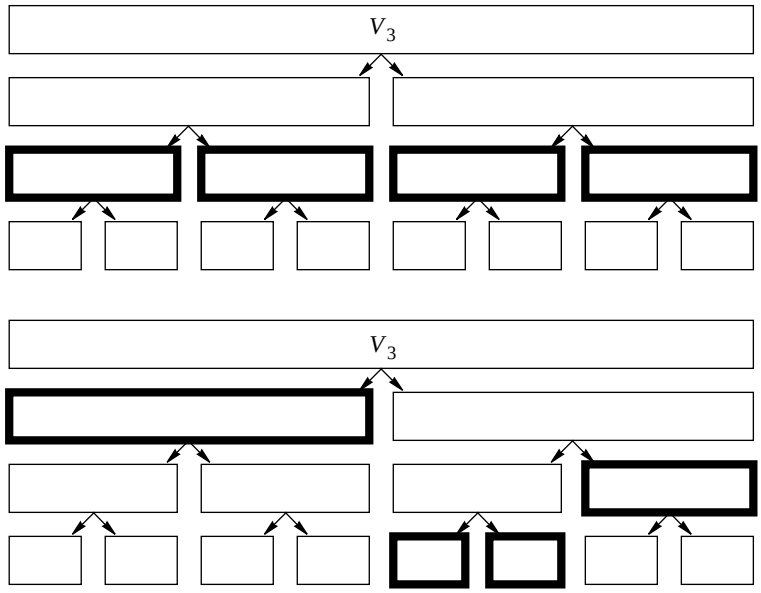


FIGURE 57. two orthonormal bases from the wave packet library

The second wave packet basis from figure 57 is also a decomposition in four frequency bands. But here the bands are not equal spaced. The large rectangle corresponds to the basis vectors with the highest time and the lowest frequency resolution. The following two rectangles from the bottom of the library have the lowest time resolution and the highest frequency resolution. And consequently the middle sized rectangle on the right contains basis vectors with a medium time resolution and a medium frequency resolution. In figure 58 the decomposition of the original function in time and frequency components regarding to these two wave packet bases is illustrated using the idealized time-frequency plane.

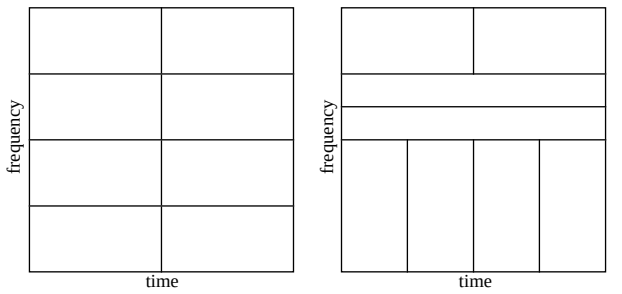


FIGURE 58. the idealized tiling of the time-frequency plane by the two wave packet bases from above

We can extend the introductory wavelet decomposition example as depicted in figure 59 on page 60. Then we have a library of transform bases to choose from. A function that is represented by n sample values expands into a library of $n \cdot \log_2 n$ transform coefficients. The number of possible transform bases is very large - there is a choice from more than 2^n different bases [14]. Here we have 8 sample values which expands into $8 \cdot \log_2 8 = 8 \cdot 3 = 24$ coefficients. Each of these coefficients addresses a basis vector which we call wave packet. These basis vectors include the wavelets and scaling functions that were the basis vectors for the wavelet transform. We can generate one of the 24 basis vectors by filling the tree with 0‘s except for a single 1 and calculate the recomposition.

There are two motivations for the choice of a specific wave packet basis. Either we want to transform a function into a basis with specific characteristics in the time-frequency plane or we want to use any transform basis that allows high compression. The first approach is used for analyzing or modifying certain attributes of a function in the time-frequency domain. Then a fixed wave packet basis is chosen that yields into the desired decomposition. The other approach adapts the chosen basis to the respective function in order to minimize the number of coefficients needed for representing the function.

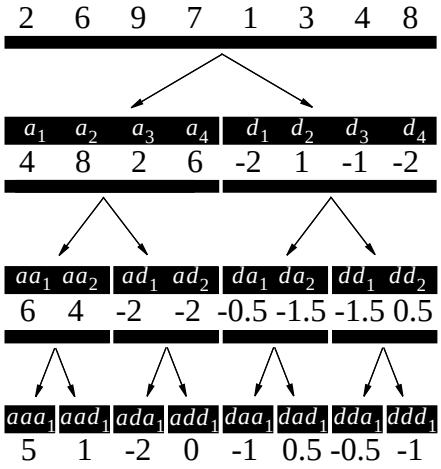


FIGURE 59. the decomposition of a function into the wave packet library of bases

Recall how we compressed the wavelet representation of a function in figure 35 on page 44. We do the same here but choose a basis from the wave packet library that either allows a higher compression or results in a more accurate reconstruction. In figure 60 we use two

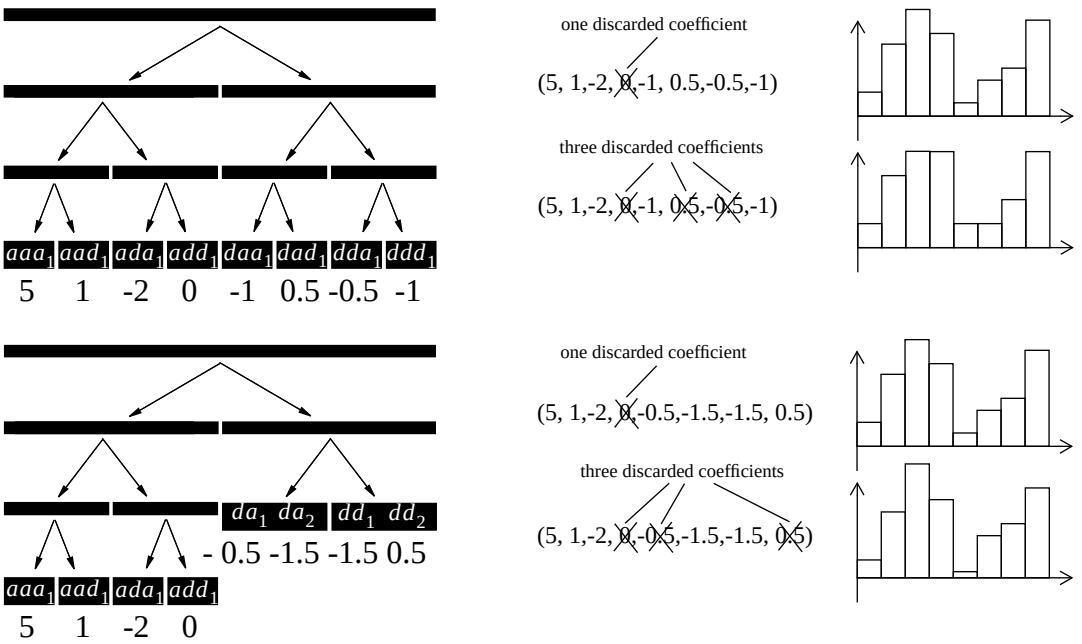


FIGURE 60. two different wave packet bases represent and compress a function

different wave packet bases to represent the example function as depicted by the decomposition tree on the right. Then we compress this representation by discarding at first one and then three transform coefficients. The reconstruction of the function from its compressed representation in the respective wave packet basis is shown on the right.

For compression purposes it is desirable to choose a wave packet basis that concentrates the most information about the decomposed function in only a few transform coefficients. We can choose the set of coefficients to represent the function with respect to a certain criterion. This procedure is called best basis selection and one can design fast search algorithm that make use of the tree structure. The best basis search algorithm was proposed by Wickerhauser and Coifman in [13] and [12].

We now define a real-valued cost functional M on the vectors that represent our functions and search for its minimum over all bases in the wave packet library. Such a functional should, for practical reasons, describe the concentration of information or the number of coefficients required to accurately describe the function. By this we mean that M should be large when all coefficients are roughly of the same size and small when all but a few coefficients are negligible. A map M from vectors $\bar{u} \in R^n$ to R is called an additive information cost function if $M(\bar{0}) = 0$ and $M(\bar{u}) = \sum_k M(u_k)$. If we have a vector $\bar{u} \in R^n$ and an orthonormal basis B for the vector space R^n, written as a matrix of row vectors, then $(\bar{u})_B = B\bar{u}$ is the vector of coefficients of $\bar{u}$ in the orthonormal basis B. The information cost of $\bar{u}$ in the basis B is calculated with $M(B\bar{u})$. The best basis B from a library of bases relative to M for a vector $\bar{u}$ is that for which $M(B\bar{u})$ is minimal.

We explain the best basis algorithm of Wickerhauser and Coifman using a small example. Suppose that the vector $\bar{u} \in R^8$ has been expanded into the library of wave packet bases. Then an admissible basis of this library is any disjoint horizontal cover of rectangles. Since the library of bases - depicted in figure 61 - is a tree, we can find the best basis relative to M by traversing through it. We choose the bottom most row of rectangles as the start basis. At each step of the algorithm the chosen subset of rectangles will be a disjoint cover and thus an admissible basis. We first calculate and store the information costs of all eight rectangles T_0^3 to T_7^3 from the bottom. In the row above the search is started. We calculate the information costs of each parent rectangle T_0^2 to T_3^2 and compare them to the sum of information costs that was stored with their twin dyadic daughters. For the rectangle T_0^2 these are the two rectangles T_0^3 and T_7^3 or in general the two daughters of the parent rectangle T_i^m are T_{2i}^{m+1} and T_{2i+1}^{m+1}. When the costs for the parent are smaller they are stored and the parent rectangle enters the actual selected basis, replacing any rectangles below. Otherwise the sum of information costs of the two daughters is stored and the basis selection remains unaltered. Proceeding iteratively row by row this process ends when we reach the rectangle at the top which corresponds to the original signal. In figure 62 on page 61 this

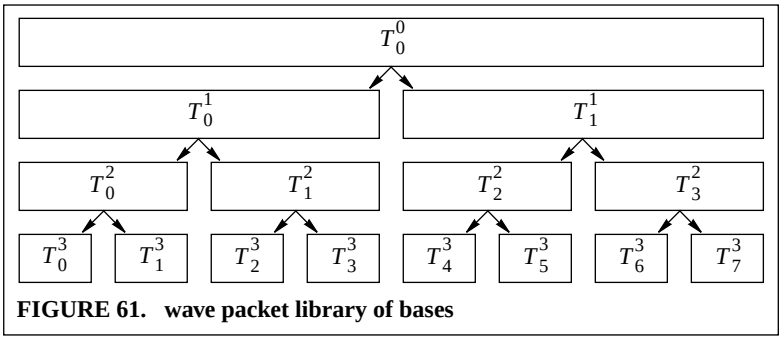


FIGURE 61. wave packet library of bases

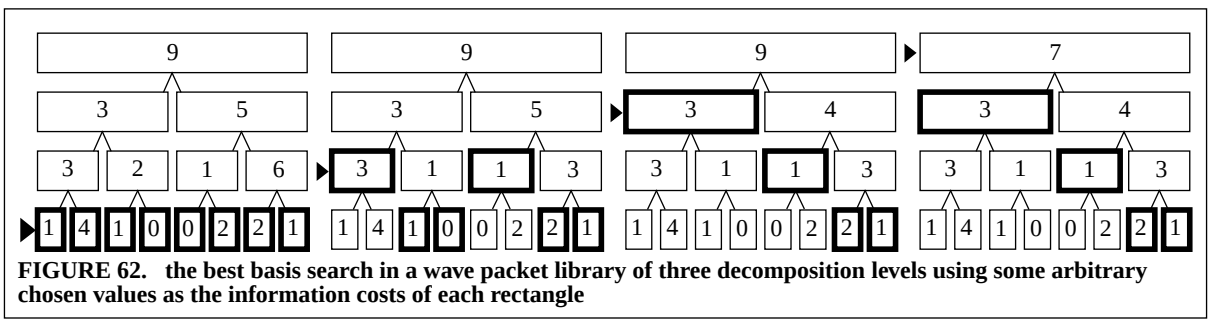


FIGURE 62. the best basis search in a wave packet library of three decomposition levels using some arbitrary chosen values as the information costs of each rectangle

best basis search is illustrated with a small example. Within a wave packet library of three decomposition levels the best basis is selected in respect to some (here arbitrary) cost function M. The values that are written in the rectangles of the left most tree stand for the resulting information costs using this cost function. At four different steps of the algorithm the actual selected basis is shown marked with bold rectangles. The algorithm terminates in the right most tree presenting the basis with the smallest information costs.

Two more things need to be discussed here in further detail. One the one hand we introduce different cost functions M and on the other hand we spend some thoughts on how to store the wave packet basis that was chosen by the algorithm.
Some useful measures of information according to Coifman and Wickerhauser [15] are:

number above a threshold. Set an arbitrary threshold ε and count the coefficients of the vector $\bar{u}$ whose absolute value exceeds ε. This is an additive measure of information. It gives the number of coefficients needed to transmit the signal to accuracy ε. The experience of the author has shown that for compression purposes the value of ε is of great importance. Especially if this value is set to high the best basis search can turn into a worst basis search. For signals with unpredictable or varying energy the threshold ε should adapt to the signal.

bit counts. Choose an arbitrary $\varepsilon > 0$ and count the binary digits in $\lfloor |u_k| / \varepsilon \rfloor$. Summing over k gives an additive measure of information. It corresponds to the number of bits needed to transmit the signal to accuracy ε.

Shannon entropy. The Shannon-Weaver entropy of a vector $\bar{u}$ is $H(\bar{u}) = -\sum_k p_k \log p_k$ where $p_k = |u_k|^2 / ||\bar{u}||^2$ and the summand is interpreted as 0 for any $u_k = 0$. This entropy is a well-known measure of the information of a distribution, but is not additive. We may also use the $L^2 \log L^2$ norm denoted by $h(\bar{u}) = -\sum_k |u_k|^2 \log |u_k|^2$ with the same convention for $u_k = 0$ rather than the entropy. See [13] for more information.

The wave packet transform is said to perform very good for compression purposes because the transform basis adapts itself to the function through the best basis search. Since the basis varies from transform to transform the information which wave packets are addressed by the transform coefficients has to be transmitted as well. Wickerhauser proposes in [47] the following: After expansion into wave packet coefficients and compression by discarding the small ones, it is necessary to transmit the surviving values as well as their position in the dyadic cover. Suppose that the function consists of n samples each b bits long and thus requiring nb bits of storage. When transformed into the optimal basis, say that only n' of the coefficients exceed the cutoff. Since no cheating is allowed, the coefficients can only contain b bits each. Specifying their positions within $\log_2 n$ times n coefficients takes $\log_2 n + \log_2 \log_2 n$ bits. This adds up to $n'(b + \log_2 n + \log_2 \log_2 n)$ bits. The compression ratio [47] can be computed with:

$$\rho = \frac{n'}{n}\left(1 + \frac{\log_2 n + \log_2 \log_2 n}{b}\right)$$

It is immediate that the compression ratio decreases immediately with increasing n.

When the number of discarded coefficients ranges between 0% and 70% the indexing approach of Wickerhauser contains to much redundancy. As one sees in table 6 there are two other approaches that can reduce the needed bit rate. Both take an indexing bit field to determine the used wave packet basis. The picture in figure 63 on page 63 is probably the easiest way to explain how a wave packet basis for a function consisting of n samples is efficiently represented with n bits. Every rectangle has a corresponding bit except for those from the bottom most row. The bit that corresponds to rectangle T_i^m has the index number $2^m + i$. A bit is set when the corresponding rectangle needs to be computed from its daughters. In figure 63 there are two example wave packet bases on the left with an illustration of their respective bit fields on the right. Whether the bits - enumerated from 1 to 7 - are set or not indicates the thickness of the rectangle. In case a rectangle is bold the corresponding bit is set, otherwise it is zero. There is no bit with the index number 0.

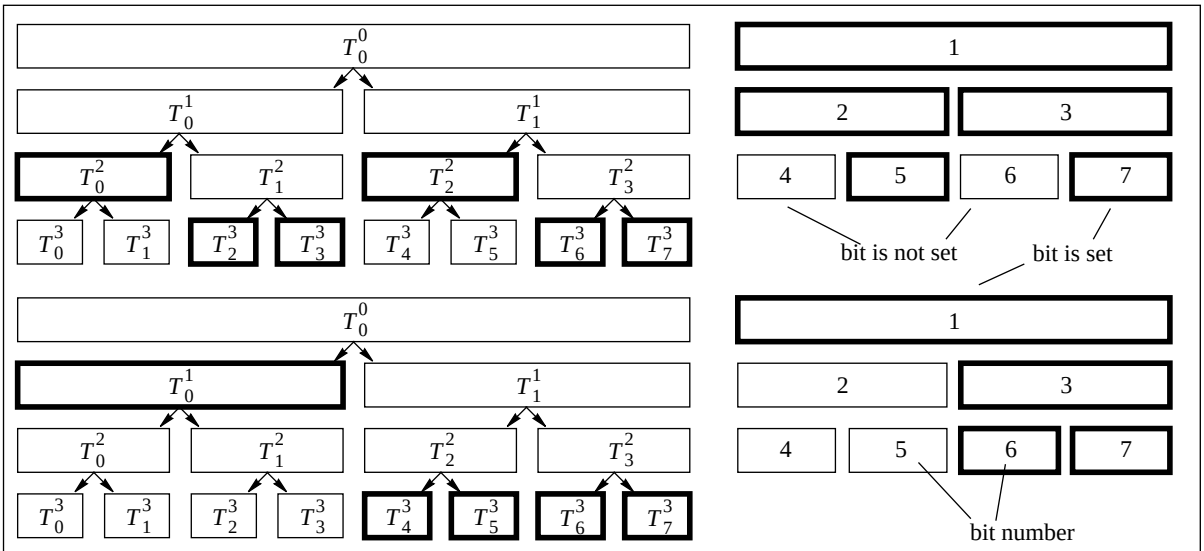


FIGURE 63. indexing a wave packet basis from the library of bases through a bit field

n	10%	25%	50%
256	(a) 26*(8+3) = 286 (b) 256+26*8 = 464 (c) 256+256 = 512	(a) 64*(8+3) = 704 (b) 256+64*8 = 768 (c) 256+256 = 512	(a) 128*(8+3) = 1408 (b) 256+128*8 = 1280 (c) 256+256 = 512
65536	(a) 6554*(16+4) = 131080 (b) 65536+6554*16 = 170400 (c) 65536+65536= 131072	(a) 16384*(16+4) = 327680 (b) 65536+16384*16 = 327680 (c) 65536+65536= 131072	(a) 32768*(16+4) = 655360 (b) 65536+32768*16 = 589824 (c) 65536+65536= 131072

TABLE 6. The number of bits needed to transmit the positions of 10%, 25% or 50% of the coefficients for a wave packet expansion with 256 (65536) samples using three different methods. Method (a) is the one proposed by Wickerhauser, method (b) indexes the wave packet basis but then addresses each sample separately, method (c) indexes the wave packet basis and uses a bit field to determine the transmitted coefficients

Even though the wave packet basis is determined, the coefficients position within the array of n wave packet coefficients needs to be specified. The two different ways to do so result into method (b) and (c) from table 6 on page 63. In method (b) every coefficient is accompanied by its location within the coefficient array. For an array of length n this can be done with $\log_2 n$ bits. Then the compression ratio - which still decreases with increasing n - can be computed with:

$$\rho = \frac{n'}{n}\left(1 + \frac{\log_2 n}{b}\right) + \frac{1}{b}$$

However, for most cases method (c) is more efficient. An indexing bit field of n bits contains the position information about all coefficients. In case the coefficient at position i in the array is transmitted, bit number i in the bit field is set to 1. It is obvious that for this approach the order in which the coefficients are arranged is dictated by the order of the 1‘s in the bit field. Here the resulting compression ratio is calculated by:

$$\rho = \frac{n'}{n} + \frac{1}{b} + \frac{1}{b}$$

5 The wave level transform

The wave packet basis in the top of figure 57 on page 59 consists of rectangles from one decomposition level only. It is what we before called a subband basis. In the following we will call a wave packet decomposition that chooses all transform coefficients from the same fixed decomposition level a wave level transform. Then there is no best basis search and a fixed subband basis becomes the transform basis which is determined by the chosen filter and the decomposition depth of the wave level transform. In figure 64 the three possible subband bases of the wave level transform for the space V_3 are shown.

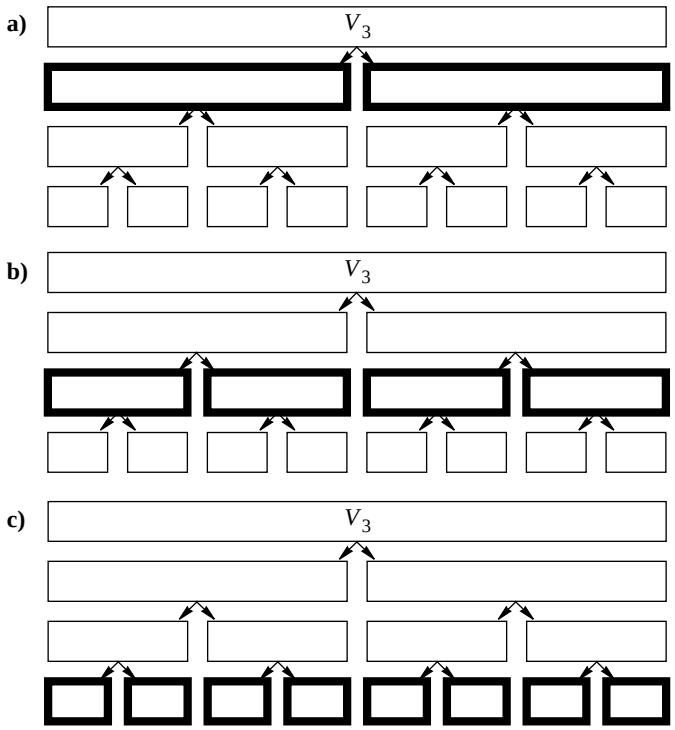


FIGURE 64. the three possible subband basis of the wave level transform: decomposition level one (a), decomposition level two (b), decomposition level three (c)

VI

Audio encoding schemes

In this chapter some new and efficient audio encoding schemes are about to be presented. The main features of these encoders are flexible bit rates, graceful degradation, layering, sample rate independence and small computational complexity. They are the result of the authors research work at the International Computer Science Institute at Berkeley. The initial concepts for the design and structure of these codecs were contributed by Hartmut Chodura from the Fraunhofer Institut für Graphische Datenverarbeitung at Darmstadt.

Up to now there did not exist an audio coder with an output rate that can be controlled over a wide range of bit rates. with a relatively fine granularity. The usual way to work around this problem is to use a panoply of audio codecs. As mentioned earlier there are PCM, various ADPCM, GSM and LPC coders. This makes it possible to choose the coding scheme with a bit rate that is closest to that desired. However, the granularity of the rate adjustment is coarse. Furthermore some of the encoders (LPC) put constraints on the possible sampling rate which makes them impractical for wideband speech or high-quality audio transmissions [8].
Before we introduced two different Forward Error Correction schemes. For the first approach - sending low resolution versions of previous packets sort of piggybacked on every current packet - it was necessary to employ different audio encoders. One for producing the main information about the audio signal and one for producing the redundancy. The audio codec introduced here simplifies this procedure. This encoding scheme can produce both at once - the main and the redundant information. The redundancy that allows the reconstruction of a low resolution versions is simply a specific fraction from the encoding of the main information.
For the second approach - using Priority Encoded Transmission - it was necessary to use an encoding scheme consisting of several layers with different importance. The key word was graceful degradation. Since there was no such codec for real-time audio streams available there was the need to develop a layered audio encoding scheme.

Even though our audio codecs can work with any sampling rate, most of the investigation was done with audio signals sampled at 8000 Hz. Some other existing encoding schemes only work at this fixed sampling rate. Measuring the efficiency of our coders is easier, when there are some standards that we can compare them with. For computational reasons the sample values are transformed into floating point values between -1.0 and 1.0.

1 Transform coding

The coding techniques that are employed here are known as transform coding (TC). In transform coding systems each block of audio samples is transformed independently from all other blocks into a set of transform coefficients. This method is also referred to as vector quantizing.

The operation applied to the block of audio samples is a simple basis transformation. Then a subset of the transform coefficients is quantized and transmitted. An inverse transform is taken at the receiver to obtain the corresponding block of reconstructed audio samples. The traditional aim of transform coding is to compress the amount of data needed to represent an audio signal [49]. We extend this aim and use transform coding for a compressed and layered representation of an audio signal.

We arrange n successive audio samples into the vector $\bar{u}$ - an element of R^n. This vector $\bar{u}$ is the representation of the block of audio samples regarding to the standard basis S. Therefore we better use the notation of a coordinate vector $(\bar{u})_S$. Transform coding is a basis transformation from the standard basis S into some other basis B. That is the transformation of the vector $(\bar{u})_S$ to a vector $(\bar{u})_B$. The idea is to have a better set of basis vectors for representing the block of audio samples than the standard basis S. Better means that more information about the audio signal can be represented by addressing less basis vectors. This yields into a compressed representation of the audio signal. Better also means that using just a few basis vectors results into an approximation of the audio signal with a granularity depending on the amount of used basis vectors. This yields into a layered representation of the audio signal.

We will investigate the basis of the discrete Fourier transform (DFT), the basis of the discrete cosine transform (DCT), the basis of the discrete wavelet transform (DWT), the basis of the discrete wave level transform (WLT) and the library of bases of the wave packet transform (WPT). Whereas the DFT and the DCT are fixed transforms the DWT and the WLT are determined by the choice of the mother scaling function and the mother wavelet or simpler - by the corresponding Quadrature Mirror Filter pair. The WPT is even more flexible since on the one hand every QMF pair yields into a different library of bases and on the other hand there are more than 2^n possible bases in each library for the n-dimensional case. The basis vectors of various transforms for the case $n = 8$ are shown in figure 65.

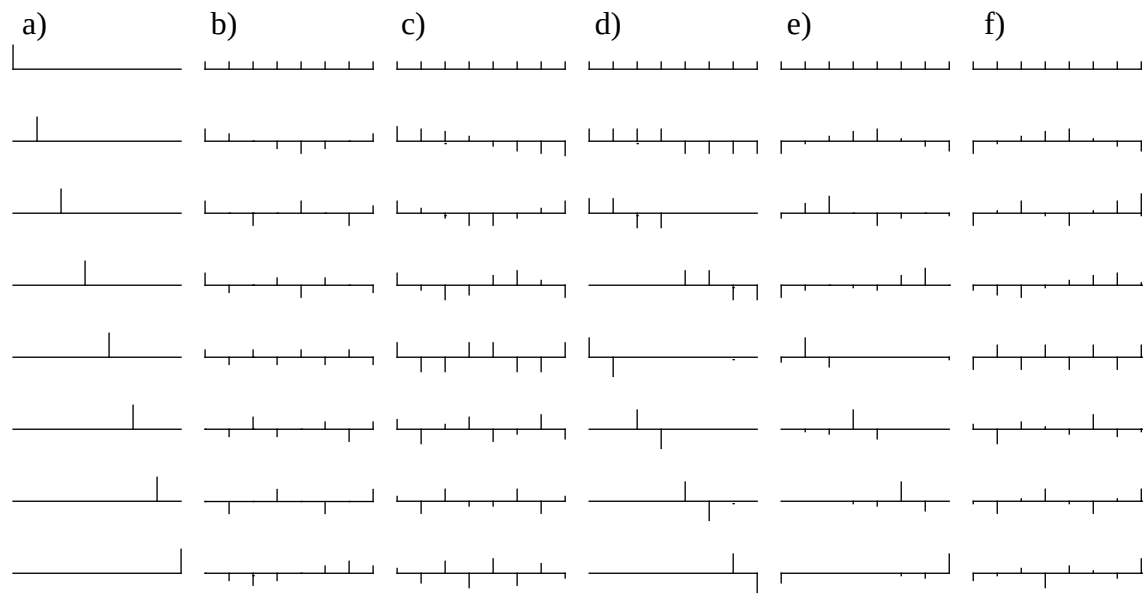


FIGURE 65. basis vectors of various transforms: (a) standard basis, (b) DFT basis, (c) DCT basis, (d) DWT basis (Haar filter), (e) DWT basis (Daubechies4 filter), (f) a subband WPT basis (Daubechies4 filter)

2 Transforming audio signals

In the following we will apply several transforms towards an audio signal and analyse the results in consideration of compression and layering. A voice signal will serve as the test data for our investigations. In figure 66 on page 67 we can see a plot representing the voice signal of the sentence ‘Let’s have some coffee’ spoken by the author. The length of the recording is roughly two seconds of speech, which equals 16384 audio samples at a sampling rate of 8kHz. The first zoomed segment contains 2048 samples which is approximately a quarter of a second and corresponds to the word ‘some’ and the beginning of the word ‘coffee’. The second zoomed segment focuses on the ‘c’ sound of the word ‘coffee’. This part is made of 256 sample

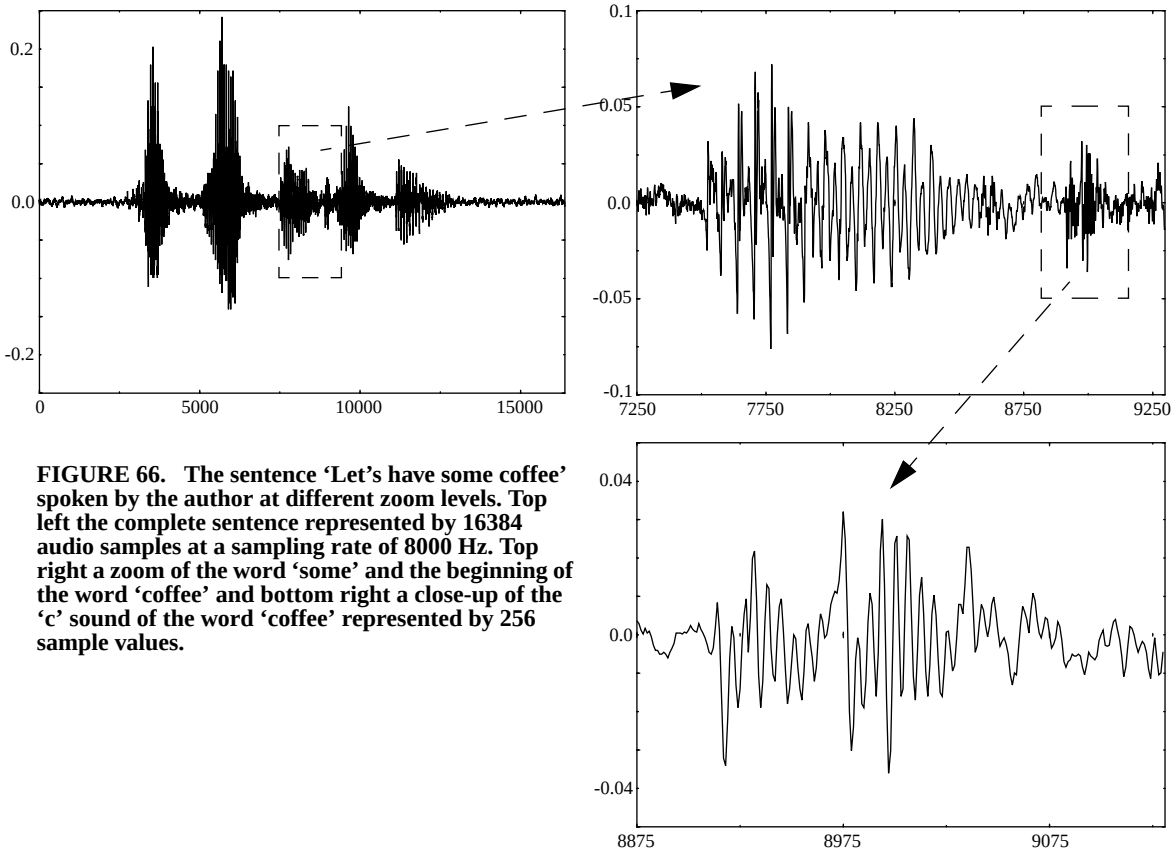


FIGURE 66. The sentence ‘Let’s have some coffee’ spoken by the author at different zoom levels. Top left the complete sentence represented by 16384 audio samples at a sampling rate of 8000 Hz. Top right a zoom of the word ‘some’ and the beginning of the word ‘coffee’ and bottom right a close-up of the ‘c’ sound of the word ‘coffee’ represented by 256 sample values.

values meaning 32 ms in time.

In the following we will apply different transforms towards these two segments and discuss the meaning of the produced representations. Since all of the introduced transforms can be seen as simple basis transformations, the inverse operation does always exits. When leaving the transform coefficients unchanged, the original signal can be exactly reconstructed through the corresponding retransform. But of course it is not our intention to leave the transform coefficients unchanged. The aims are:

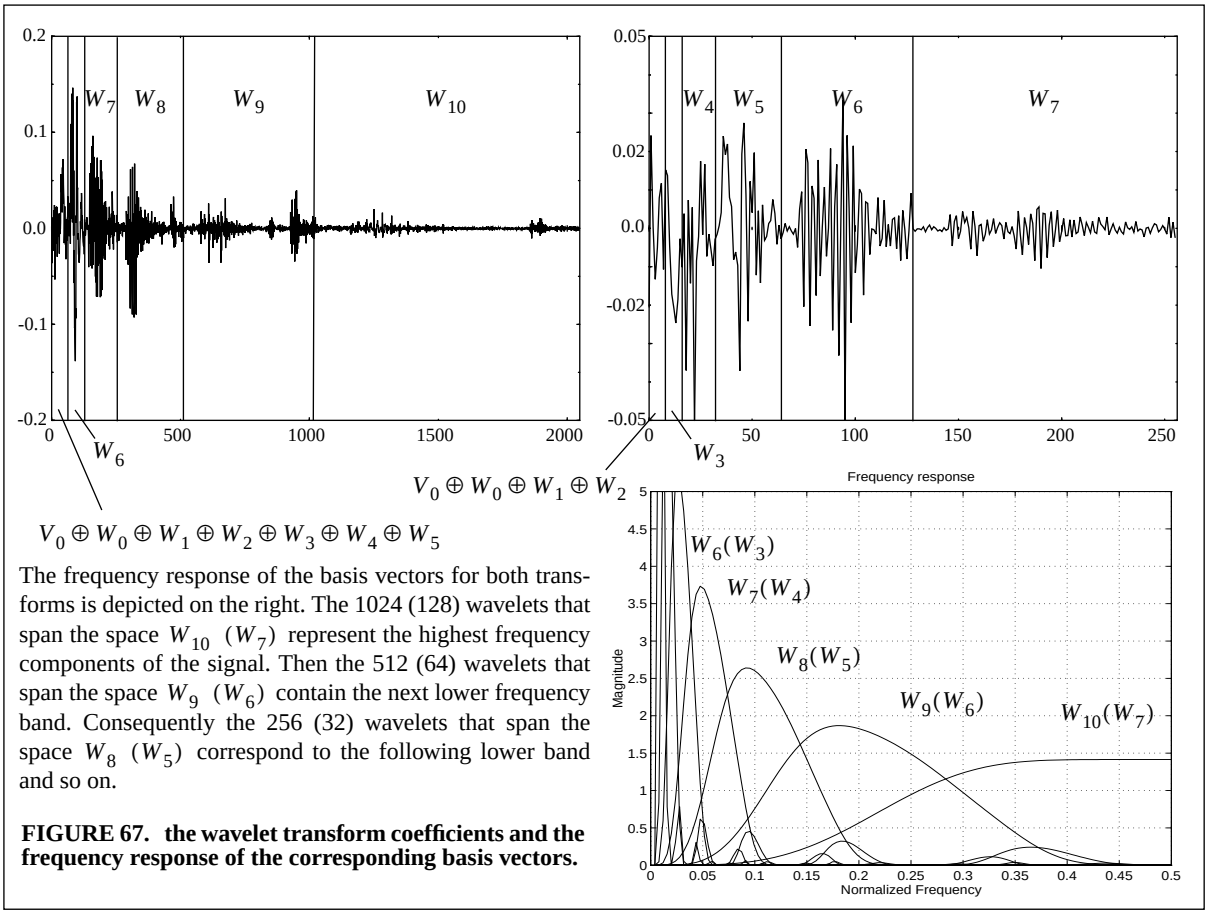
Compression: The representation in the transformed domain can be basically compressed in two ways. Transform coefficients with a very small value have only a very small impact on the recomposition of the signal. Therefore they can be omitted without causing a major distortion of the signal. The amount of data is compressed because only the information about the kept transform coefficients has to be transmitted. On the receiving side the discarded coefficients are

assumed to be zero. Transmitting just the important coefficients raises another question. How does the receiving side know, which coefficients have been transmitted? Either the omitting of coefficients is position independent, then each transmitted coefficients has to be accompanied by its position within the array. This additional information obviously reduces the obtained compression rate, since it has to be transmitted as well. Position dependent discarding of transform coefficients would be much more efficient, but this method does not take the value of the coefficient into consideration.
Besides omitting coefficients the number of bits for representing each of them plays a major role for compression. The fewer bits are used to encode the different coefficient values the greater is the achieved compression ratio. But quantizing the values of the coefficients introduces a quantization error. Using less bits and therefore less quantization levels systematically increases the quantization error.

Layering: To present a layered audio encoding scheme was the motivation for our work. Layering means splitting the encoding information into several layers. The bottom most layer has to be already a stand-alone codification of the audio signal, containing the rough overall shape of the audio signal. Retransforming this layer should result into a signal that is close to the original one. This means the retransformed audio signal should be at least intelligible. The remaining layers - the exact number is arbitrary - should add layer by layer more of the signals detail information to the codification. Consequently a reconstruction including all encoding layers should allow the perfect reconstruction of the audio signal. Then omitting layer by layer of the codification results into recompositions of the audio signal at different resolution levels.

For the sample vector containing 2048 samples - an element of R^{2048} - the decomposition through the DWT into scaling functions and wavelets start at the space V_{11}. The other sample vector with 256 samples - an element of R^{256} - starts at the space V_8. The resulting coefficients after a Daubechies8 wavelet decomposition over the total of 11 or 8 levels respectively are shown in figure 67.
For a sample rate of 8000 Hz the normalized frequency on the x-axis of the frequency response graph ranges from 0 Hz to 4000 Hz. One way to compress the representations in respect to the wavelet basis would be to omit the coefficients that correspond to the high frequencies. Looking at the plot one may think this is a good idea, since the coefficients for W_{10} (W_7) are comparatively small. This is because the frequency spectrum of the human voice ranges mainly between 500 Hz and 1500 Hz. But this procedure - which would be a position dependent discarding of coefficients - equals a band-limiting of the audio signal with a very bad low-pass filter. This fact becomes immediately clear in figure 68. For this way of compression a full wavelet decomposition is an unnecessary computational overhead. Only the first iteration that splits V_{11} (V_8) into V_{10} and W_{10} (V_7 and W_7) would be already sufficient. But - the compressed signal is of poor quality. Because of the missing high frequencies the resulting signal sounds hollow. Furthermore the remaining low-frequency signal sounds distorted since the used low-pass filter is not a perfect one.

The following graphic shown in figure 69 on page 69 elucidates why the wavelet decomposition yields into a higher compressed representation of the audio signal. The absolute values of both - the samples of the original representation and the coefficients of the wavelet transform - are sorted and plotted in decreasing order. Especially for the audio segment with 2048 sample values the desired result is visible. A big number of coefficients is almost zero and there are a few with a very large value. This behaviour almost dictates the design of the compression scheme that we are about to present. One has to recall that both - each coefficient of the wavelet



The frequency response of the basis vectors for both transforms is depicted on the right. The 1024 (128) wavelets that span the space W_{10} (W_7) represent the highest frequency components of the signal. Then the 512 (64) wavelets that span the space W_9 (W_6) contain the next lower frequency band. Consequently the 256 (32) wavelets that span the space W_8 (W_5) correspond to the following lower band and so on.

FIGURE 67. the wavelet transform coefficients and the frequency response of the corresponding basis vectors.

transform and each sample of the original signal - address a basis vector in respect to some basis. For the wavelet coefficients this is the wavelet basis and for the sample values it is the standard basis. All the basis vectors are of the same length: one. Then the audio signal is composed by a linear combination of all basis vectors multiplied with the corresponding coefficients in regard to the respective basis. In case such a coefficient is very small the appropriate

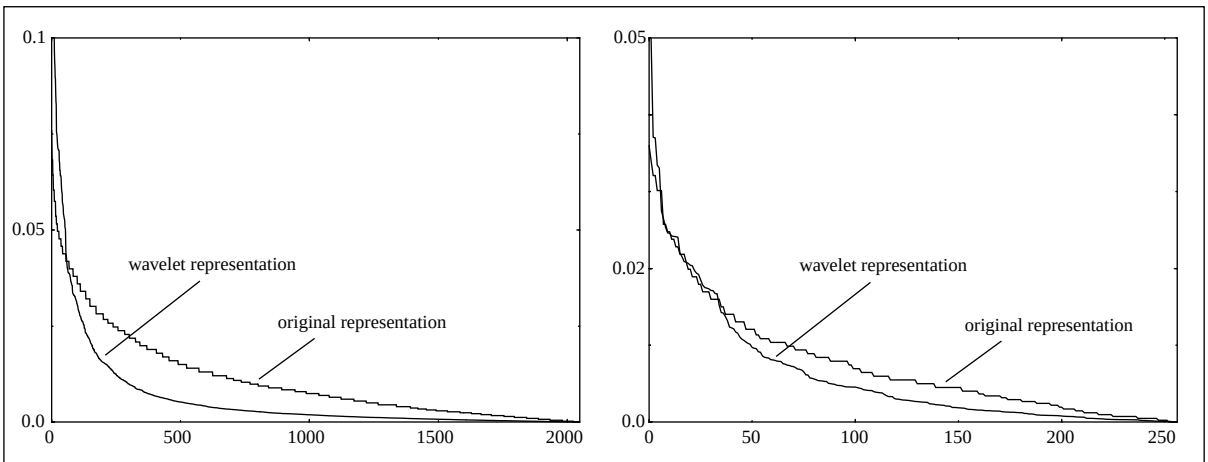
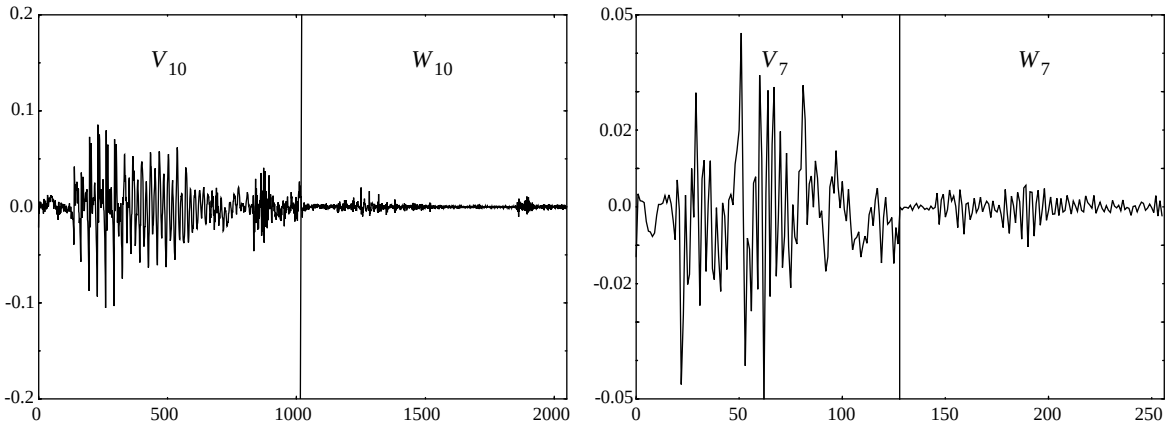


FIGURE 69. The absolute values of both - the samples of the original representation and the coefficients of the Daubechies8 wavelet transform - are sorted and plotted in decreasing order. The left graph contain the speech segment of 2048 samples, the right plot the shorter segment of 256 sample values.

basis vector does not add very much towards this linear combination. Discarding this coefficient - meaning setting it to zero - has only a small impact on the shape of the audio signal. The compression is achieved by discarding a certain number of coefficients according to their abso-



Simply omitting the coefficients that correspond to the highest frequencies, which are in this case the coefficients for W_{10} (W_7), can be seen as a low-pass filtering of the audio signal. The frequency response of this low-pass filter equals the frequency response of the scaling functions that span V_{10} (V_7).

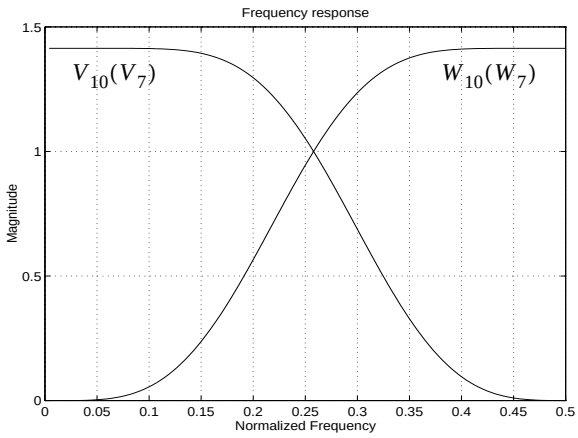


FIGURE 68. the wavelet transform coefficients and the frequency response of the corresponding basis vectors for the DWT with only one decomposition step

lute value. There are basically two ways to do that: Either omitting all coefficients below a certain threshold or else discarding a certain percentage of the lowest valued coefficients. The first method yields into a variable amount of encoding information, the latter assures an codification at a steady bit rate.

In figure 70 on page 71 and in figure 71 on page 72 the second idea is applied towards the two sample audio segments. Rather than talking about discarding a certain percentage of transform coefficients we talk about keeping a certain percentage. Both figures illustrate the same scenario: The left upper most graph shows the original audio signal with its Daubechies8 wavelet transform on the right. Below three different signal reconstructions that use the highest 40%, 20% or 5% of the wavelet transform coefficients are shown. The graphs on the right list the kept coefficients and in the graphs on the left the corresponding signal reconstruction is plotted. The increasing compression very nicely yields into a graceful degradation of the conformity with the original signal. As one might expect - looking at the audio signals plot - auditory test series confirm that this directly results into a graceful degradation of the audio quality. The observed behaviour gives us everything that we wanted:

- graceful degradation: The quality of the audio signal can be controlled by the number of transform coefficients used for reconstruction. Using more/less coefficients directly increases/decreases the achieved quality. The level of quality is infinitely variable from excellent to mediocre.
- layering: As a direct outcome of graceful degradation layering is achieved when grouping the coefficients in order of their importance together. An encoding scheme with three layers for example can use the highest 10% of coefficients for the bottom most layer, the next 20% to build the second layer and group another 30% together for the third layer.

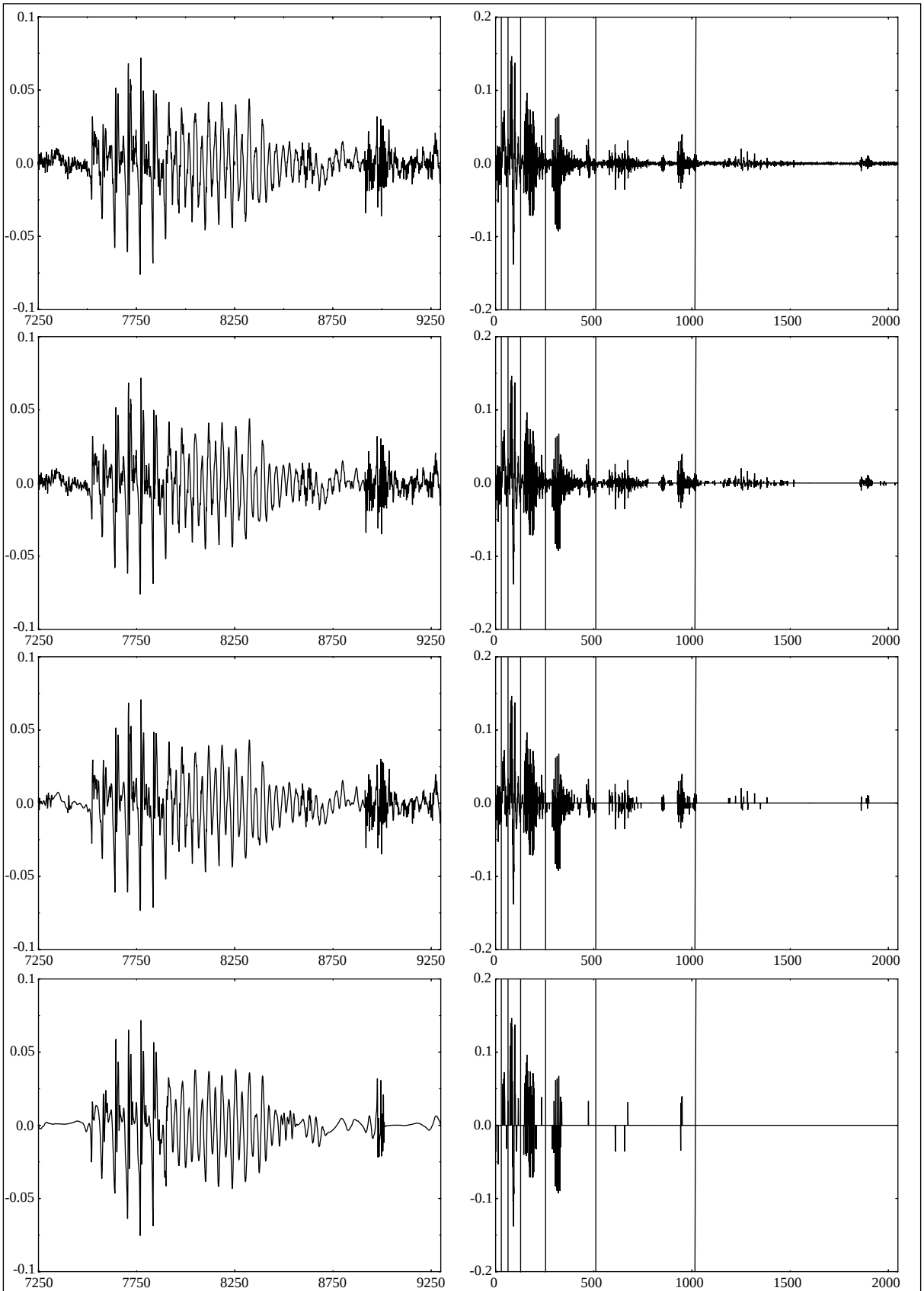


FIGURE 70. The top left graph shows the original audio signal with its Daubechies8 wavelet transform on the right. Below three different signal reconstructions that use the highest 40%, 20% or 5% of the wavelet transform coefficients are shown. The graphs on the right list the kept coefficients and in the graphs on the left the corresponding signal reconstruction is plotted.

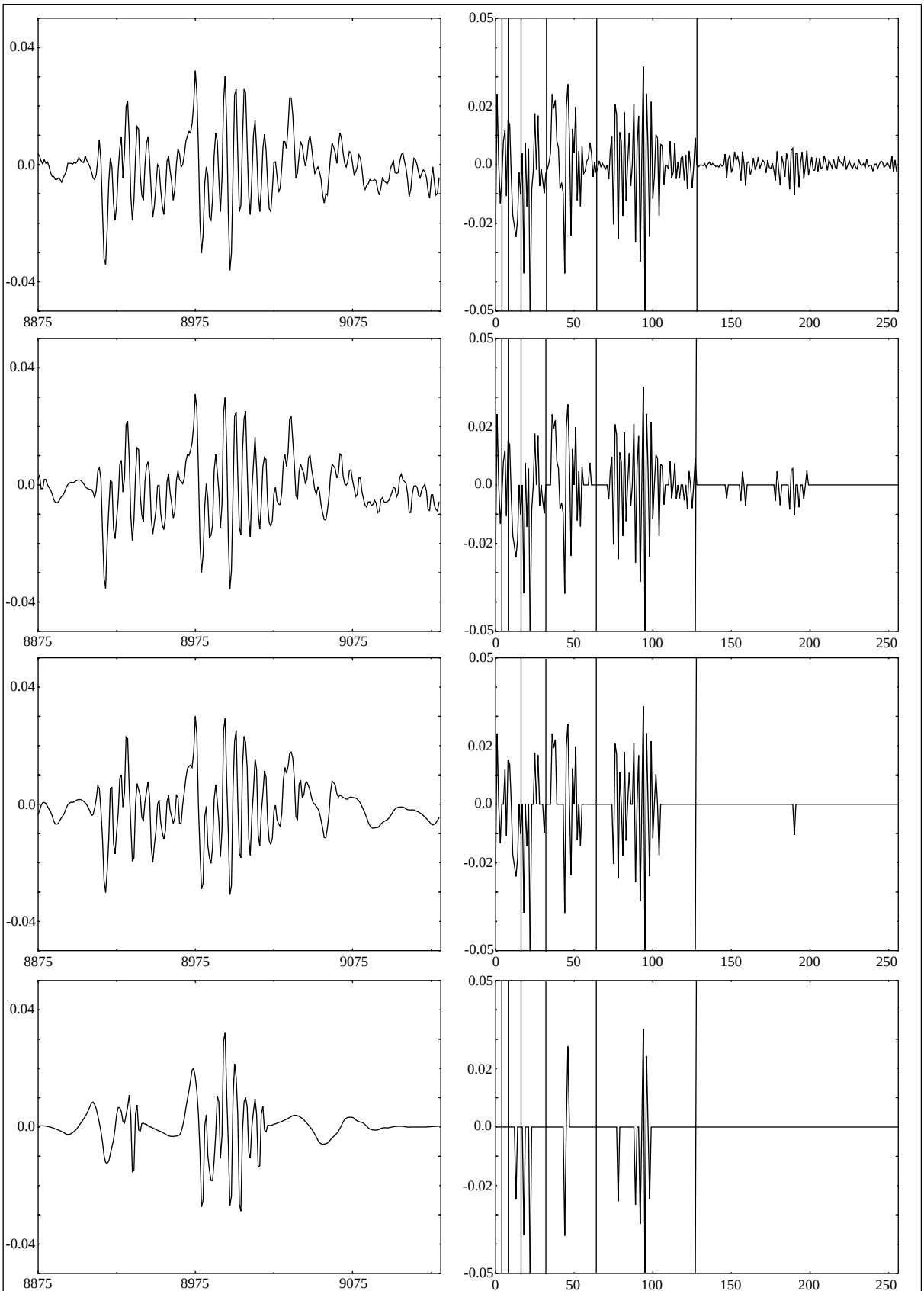


FIGURE 71. The top left graph shows the original audio signal with its Daubechies8 wavelet transform on the right. Below three different signal reconstructions that use the highest 40%, 20% or 5% of the wavelet transform coefficients are shown. The graphs on the right list the kept coefficients and in the graphs on the left the corresponding signal reconstruction is plotted.

- variable bit rate: The number of transmitted coefficients obviously influences the amount of encoding information. Analog to graceful degradation using more/less coefficients directly increases/decreases the needed bandwidth.

The Daubechies8 wavelet transform is just one of many different transforms that can be exploited for this purpose. The next step is to find the transform that outperforms all of the others given a set of criteria.

An objective criteria for a suitable transform basis is the slope of the graph of the sorted coefficients. A steep descent of this graph and thus a large number of coefficients close to zero will allow a high compression rate with almost no distortion of the audio signal. The computational complexity is also an important factor for our choice. The encoding scheme is thought to be applied towards real-time audio communication, thus the available CPU power acts as a constraint on our choice. Another and obviously even more important criteria is the subjective judge through the human ear.

In the next two sections several transforms are compared in respect to these criteria. The first section deals with the objective measurements, like the descent of the graph of the sorted coefficients or the computational complexity. The second section tries to validate these results with the help auditory test series. Two independent tools written by the author allow to compare the sound quality for speech communication using various transforms at different compression ratios.

3 Competing objectively

Recalling the observation from figure 69 on page 69 we use the slope of the graph of the sorted transform coefficients as an objective criteria for choosing the best basis transformation. More precisely we use the amount of the signals power that is represented by the highest coefficients as the criteria. Experimental evidence has shown that some of these transforms contain enough signal information within the highest 50% of the transform coefficients to allow a perfect reconstruction for speech signals. Perfect in the sense that there is no audible difference between the original and the reconstructed signal. Using a sample recording we will calculate the percentile amount of the signals power that is kept within the highest 5%, 10%, 20%, 30% and 50% coefficients of each representation.

When working on only one block of sample data our results will not allow general statements about the applied transforms. On the one hand the sample could be biased - meaning this specific segment of speech is not a good representative for common speech signals. On the other hand the resulting plots of the sorted transform coefficients look rather irregular. Therefore we do a certain number of consecutive transforms and calculate their average. However, this evaluation scenario is far from being perfect. The author - who recorded his voice as test data (see figure 72 on page 74) - does not claim to have specifically representative voice. Furthermore we were talking about audio compression, but then use only speech signals to judge the different transformations. We justify this with the fact that our objective statements will be proofed through auditory experiments. The focus on speech signals is due to time limitations and the certainty that the momentary research work of Hartmut Chodura at the Fraunhofer Institut for Computer Graphics applies the results achieved here towards high-quality audio.

In the following we are going to use the notations ‘*DWT <filter> (<minus>)*‘ and ‘*WLT <filter> (<minus>)*’ for denoting either a discrete wavelet transform or a wavelet level transform that decomposes the signal with the filter *<filter>* up to the bottom most minus

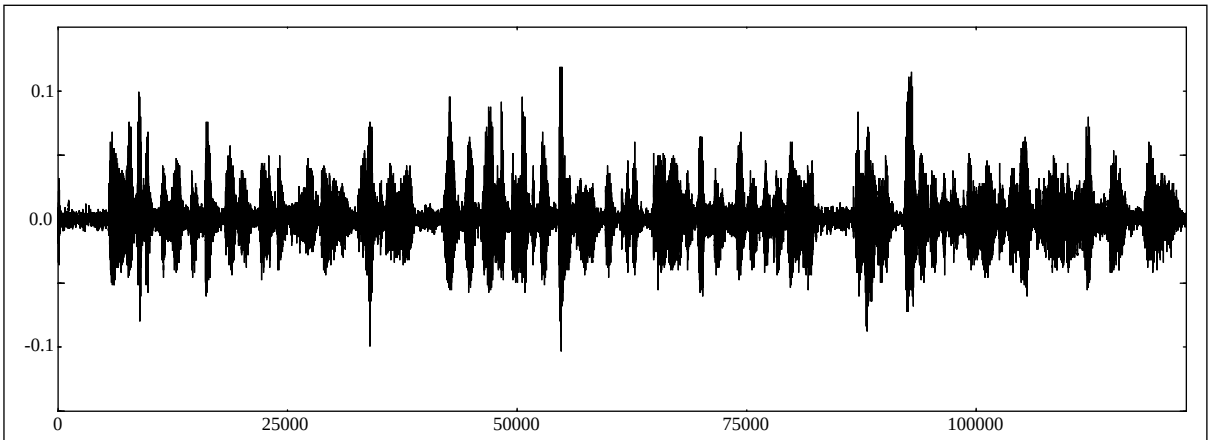


FIGURE 72. A recording with a duration of 15.36 seconds of the authors voice which equals 122880 samples values at a sampling rate of 8000 Hz is the test data for competing objectively.

<*minus*> decomposition level. For instance 'DWT Daub4 (0)' stands for a discrete wavelet transform with the Daubechies4 filter over all decomposition levels, whereas 'WLT Daub8 (2)' indicates that the Daubechies8 filter is used in a wave level transform which omits the last two decomposition steps. A similar notation for the WPT is '*WPT <filter> (<threshold>)*' meaning that a wave packet transform works with the filter <*filter*> and uses the threshold <*threshold*> for its cost-function in the best-basis search.

Like before we work with the two block length of 2048 and 256 samples. The whole sample segment contains 122880 samples which equals a recording duration of 15.36 seconds. The average over all transformations on this audio segment is made of 60 single transforms for the 2048 sample blocks and 480 independent transforms for the smaller 256 sample blocks. In

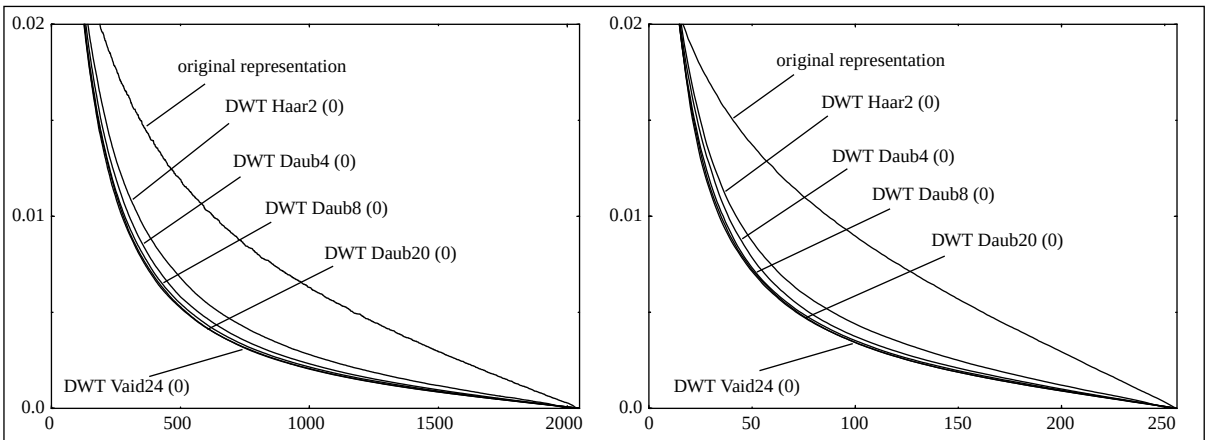


FIGURE 73. Compare to figure 69 on page 69. Here an average over 60 consecutive transformations of the speech segment (figure 72) with 2048 sample blocks and over 480 transform with 256 sample blocks is computed.

figure 73 the results of applying DWT with various filters to the sample segment are shown. It may be noted that all DWTs decompose the sample signal down to the bottom most level. For the average case it seems that the block size does not influence the outcome of the transform at all. As a first result we notify that with increasing filter length the plot of the sorted transform coefficients becomes steeper. However, the resulting difference between the Daubechies8, Daubechies20 and Vaidyanathan24 transform is fairly small compared to the difference in their complexity. A more exact picture about the performance of the different transforms is given in table 7 on page 75. The table makes it immediately clear why the highest 50% of the transform coefficients allow an almost perfect reconstruction of the speech signal. Our choice - assuming

transform	5%		10%		20%		30%		50%		CPU
none	39.14	27.61	56.30	44.57	75.52	65.95	85.88	78.64	95.69	92.91	140
DWT Haar2 (0)	68.55	59.12	82.98	76.19	93.14	89.26	96.77	94.34	99.22	98.47	373
DWT Daub4 (0)	73.25	63.59	86.58	79.98	95.08	91.66	97.78	95.79	99.59	98.90	471
DWT Daub8 (0)	75.49	66.31	88.11	82.04	95.72	92.57	98.07	96.22	99.55	99.01	533
DWT Daub20 (0)	76.42	67.14	88.71	82.61	96.03	92.82	98.23	96.43	99.59	99.08	2485
DWT Vaid24 (0)	76.63	67.74	89.02	83.00	96.12	92.88	98.26	96.40	99.60	99.07	2888

TABLE 7. The percentile amount of the total signal power contained in the highest 5%, 10%, 20%, 30% and 50% coefficients per block for the DWT transform with various filters. The first number results for a transform block length of 2048 samples taking 60 single transformations for the sample signal. Doing the same in 480 transformations with smaller 256 sample blocks yields into the second given value. The relative CPU costs are a coarse indication of the transforms complexity.

we are subject to computational constraints - could be the DWT with the Daubechies8 filter. But there are still a lot of other transformations to examine.

Using the same scenario we compare the WLT and the WPT with the DWT in figure 74. For all three transforms first the Daubechies8 and then the Vaidyanathan24 filter are the somewhat

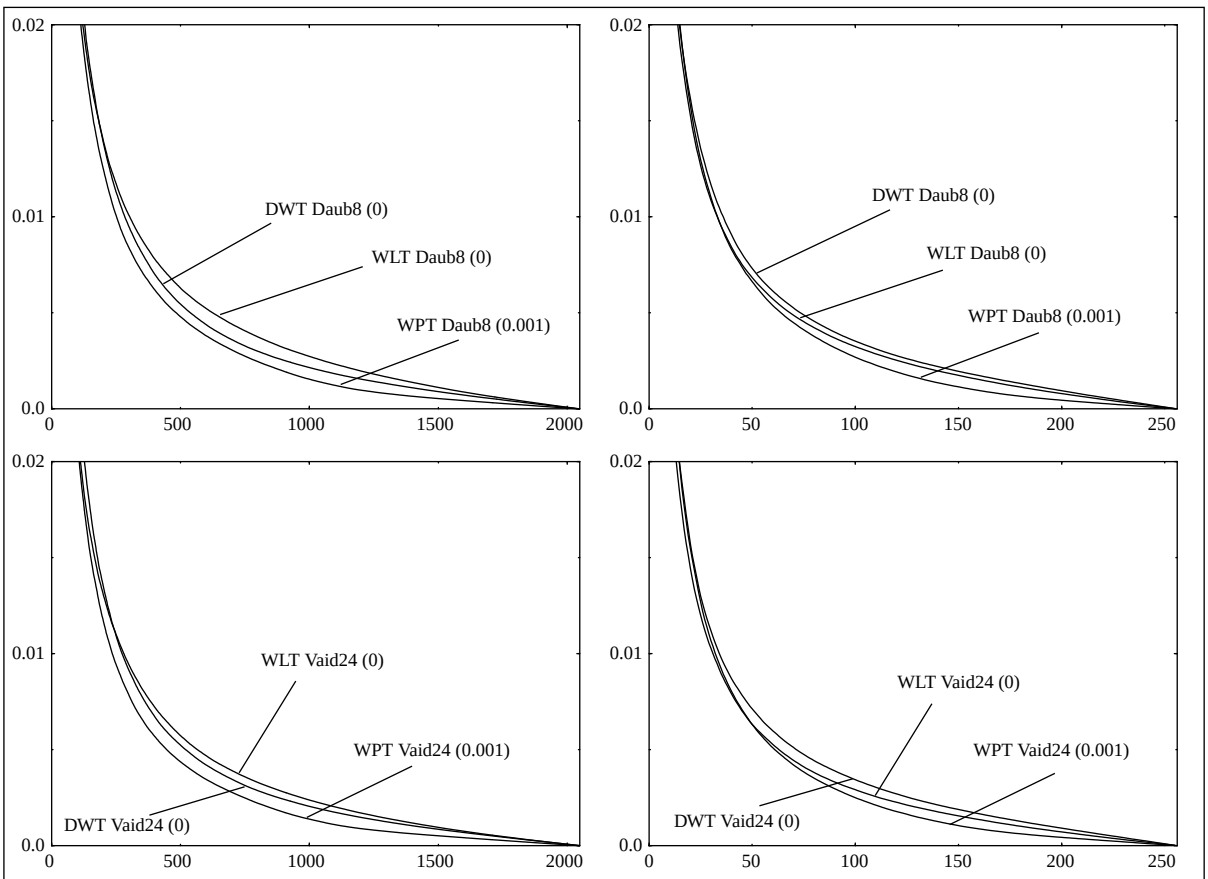


FIGURE 74. The same scenario as in figure 73 on page 74. The discrete wavelet transform, the wave level transform and the wave packet transform are compared using the Daubechies8 (top) and the Vaidyanathan24 filter (bottom).

arbitrary choices. In table 8 the graphical results are again underpinned by evaluating how much of the signal power the highest coefficients contain. We can state immediately that the wave packet transform - using the same filter - clearly outperforms all other transforms. This is not surprising since the WPT computes a library of transform bases that contains both - the basis of the DWT and the basis of the WLT. In regard to computational complexity the WPT

transform	5%		10%		20%		30%		50%		CPU
DWT Daub8 (0)	75.49	66.31	88.11	82.04	95.72	92.57	98.07	96.22	99.55	99.01	533
WLT Daub8 (0)	74.01	70.85	85.74	85.03	94.02	94.01	97.09	97.05	99.30	99.29	1758
WPT Daub8 (0.001)	79.35	72.27	90.13	86.47	96.63	95.16	98.65	97.91	99.81	99.67	2373
DWT Vaid24 (0)	76.63	67.74	89.02	83.00	96.12	92.88	98.26	96.40	99.60	99.07	2888
WLT Vaid24 (0)	77.71	74.35	88.07	87.23	95.28	95.15	97.79	97.69	99.49	99.45	14743
WPT Vaid24 (0.001)	83.21	71.93	92.48	87.18	97.57	95.83	99.05	98.29	99.86	99.74	15448

TABLE 8. The procentual amount of the total signal power contained in the highest 5%, 10%, 20%, 30% and 50% coefficients per block for the discrete wavelet transform, the wave level transform and the wave packet transform using the Daubechies8 and the Vaidyanathan24 filter. Compare to table 7 on page 75.

and the WLT are almost equivalent - at least for the case where the WLT does a full decomposition of the sample block down to the bottom most level. This is because then the wave level transform computes exactly the same amount of transform coefficients. Whereas the computational overhead for the best basis search is comparatively small, the performance is way better when using a best base instead of a fixed subband base.
Nevertheless the wave packet transform is neglected in the following considerations. This is due to the fact that the information about the actual chosen wave packet basis has to accompany the kept coefficients. This unavoidable overhead of information takes away the practical use of the WPT for compression purposes. The competing wave level transform for instance definitely outperforms the wave packet transform in respect to compression, when the amount of bits needed to specify the wave packet bases is invested into keeping a higher percentage of transform coefficients instead.

The expectation of the author - based on experimental knowledge - was that the WLT would always outperform the DWT. This holds true for the transforms of the 256 sample block, but for the transform of 2048 sample blocks the opposite is the case. In some situations the ‘WLT Daub8 (0)’ turns out to be even worse than the simple ‘DWT Daub4 (0)’. We suspect that the full decomposition of the sample block down to the bottom most level may not always be the best choice. In figure 75 we investigate the behaviour of the WLT for different depths of

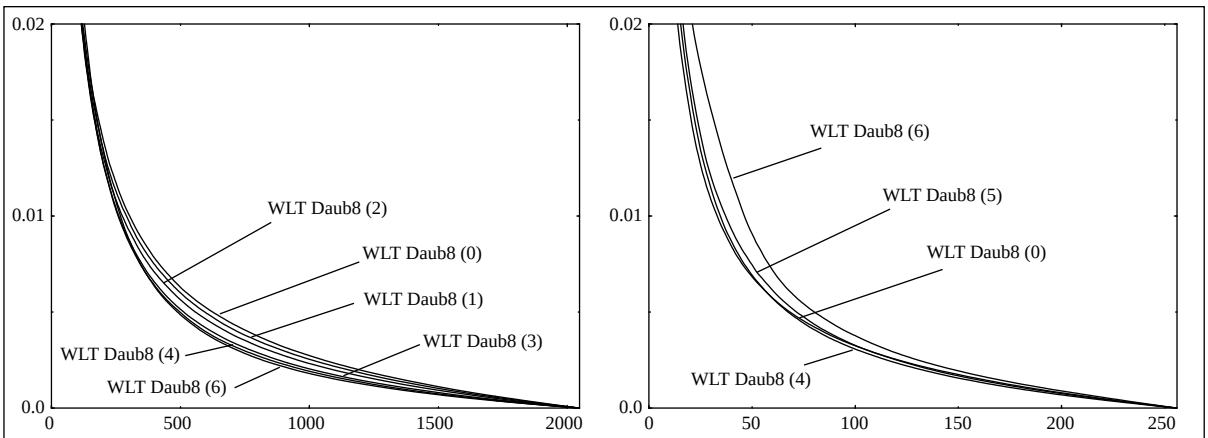


FIGURE 75. The same scenario again. Here wave level transforms with the Daubechies8 filter are compared at different decomposition depths

decomposition. Our assumptions are immediately confirmed. For the transform with large sam-

transform	5%		10%		20%		30%		50%		CPU
WLT Daub8 (0)	74.01	70.86	85.74	85.03	94.02	94.01	97.09	97.05	99.30	99.29	1758
WLT Daub8 (1)	75.89	69.74	87.07	84.64	94.69	93.95	97.46	97.07	99.40	99.31	1556
WLT Daub8 (2)	77.72	68.57	88.46	84.17	95.41	93.92	97.86	97.09	99.51	99.34	1357
WLT Daub8 (3)	79.38	68.75	89.97	84.51	96.29	94.29	98.32	97.35	99.63	99.42	1192
WLT Daub8 (4)	79.58	68.49	90.39	84.73	96.58	94.38	98.49	97.41	99.67	99.44	962
WLT Daub8 (5)	77.87	65.20	89.97	82.55	96.62	93.45	98.55	96.97	99.70	99.33	766
WLT Daub8 (6)	77.91	53.41	90.17	75.10	96.77	91.23	98.63	95.88	99.72	99.04	617
WLT Daub8 (7)	77.76	40.03	89.99	60.75	96.76	82.48	98.63	92.06	99.71	98.39	532

TABLE 9. The procentual amount of the total signal power contained in the highest 5%, 10%, 20%, 30% and 50% coefficients per block for WLT at different decomposition depths (compare to table 7).

ple blocks the ‘WLT Daub8 (0)’ has actually the worst results. The ‘WLT Daub8 (6)’ would be a much better choice that furthermore needs less than half the computation time. The plot on the right tells us that for the transform with small sample blocks the ‘WLT Daub8 (0)’ is already almost the best choice. For a more exact performance verification of the individual WLTs at different decomposition depths see table 9 on page 77.
Once the decision about the used filter is made both the DWT and the WLT have one degree of freedom left - the decomposition depth. Different decomposition depths result into different transform bases. Thus the DWT and the WLT can also be seen as a library of bases, although this library is very small. For sample blocks of length 2^n each of the transforms offers n transform bases to choose from. We could turn the discrete wavelet transform (DWT) into a best discrete wavelet transform (BDWT) and the wave level transform (WLT) into a best wave level transform (BWLT) by applying the best basis algorithm of the WPT to their small libraries of bases. In figure 76 we have done this for the WLT. The exact results can be obtained in table 10.

transform	5%		10%		20%		30%		50%		CPU
BWLT Daub8 (.0005)	77.92	67.60	90.16	83.94	96.79	94.15	98.64	97.36	99.72	99.45	2096
BWLT Daub8 (.001)	78.18	68.08	90.27	84.38	96.82	94.42	98.66	97.46	99.73	99.47	2096
BWLT Daub8 (.005)	79.18	70.76	90.69	85.84	96.91	94.76	98.67	97.60	99.72	99.46	2096

TABLE 10. The percentile amount of the total signal power contained in the highest 5%, 10%, 20%, 30% and 50% coefficients per block using the best wave level transform with a best basis search over all decomposition levels.

The performance of the BWLT is at least as good as the best possible WLT transform using a fixed decomposition level. The big advantage is that one does not have to worry whether the chosen decomposition level is suitable. The best level search takes care that a level is chosen that (almost) maximizes the performance of the WLT. The increase in computational effort is acceptable and can even be reduced because of the following observations. In table 11 we summarized some other results of the best level search. For the above scenario we counted the number of times each decomposition level was declared to be the best in respect to the used threshold for the cost function. The result is unmistakable and has been verified by using different filters and various transform block lengths. The decomposition depths 9, 10 and 11 have been a seldom choice of the best level search. The same measurements have been taken using the Vaidyanathan24 filter. They are listed in table 12 and show the same characteristics. A

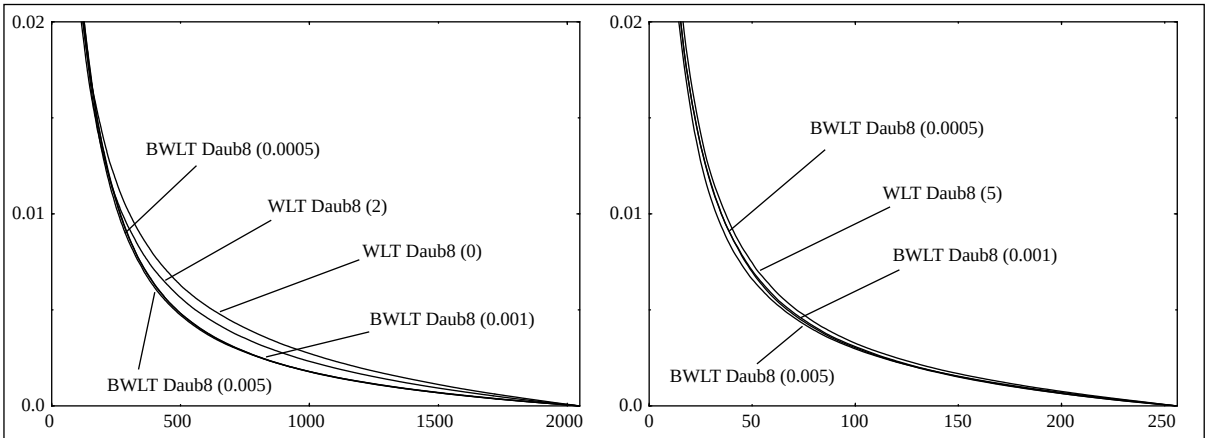


FIGURE 76. The wave level transform extended by a best basis algorithm over all possible decomposition depths is compared to wave level transforms at fixed decomposition levels.

transform	1	2	3	4	5	6	7	8	9	10	11
BWLT Daub8 (.0005)	0 3	0 22	3 62	19 100	18 114	13 61	4 55	2 63	0 ---	1 ---	0 ---
BWLT Daub8 (.001)	0 0	0 23	4 61	20 108	25 94	3 65	6 60	2 69	0 ---	0 ---	0 ---
BWLT Daub8 (.005)	0 1	0 4	3 44	18 87	16 76	11 63	5 91	5 115	1 ---	0 ---	1 ---

TABLE 11. The number of times each decomposition level was chosen by the best basis search algorithm of the best wave level transform. The left number corresponds to the 60 transforms with a block length of 2048 and the right number corresponds to the 480 transforms with 256 samples.

transform	1	2	3	4	5	6	7	8	9	10	11
BWLT Vaid24 (.0005)	0 3	0 39	8 85	15 84	16 77	17 63	2 64	1 69	0 ---	1 ---	0 ---
BWLT Vaid24 (.001)	0 0	0 22	6 77	18 78	23 90	9 58	2 70	1 80	0 ---	0 ---	1 ---
BWLT Vaid24 (.005)	0 1	0 6	3 50	17 66	18 59	10 70	7 80	3 148	0 ---	0 ---	2 ---

TABLE 12. The same measurements of the best basis search algorithm for the best wave level transform as in table 11 using the Vaidyanathan24 instead of the Daubechies8 filter.

transform	5%		10%		20%		30%		50%		CPU
BWLT Daub8 (.0005)	77.79	66.97	90.11	83.71	96.79	94.09	98.64	97.34	99.72	99.45	1217
BWLT Daub8 (.001)	77.89	66.92	90.17	83.85	96.81	94.27	98.65	97.42	99.72	99.47	1217
BWLT Daub8 (.005)	78.04	68.82	90.23	84.98	96.86	94.58	98.67	97.54	99.72	99.47	1217

TABLE 13. The procentual amount of the total signal power contained in the highest 5%, 10%, 20%, 30% and 50% coefficients per block using the best wave level transform with a best basis search that is restricted to the five first decomposition levels.

direct and useful result of these observations is to stop the BWLT algorithm after reaching the 8th decomposition level or even earlier. In table 13 the results are shown that were obtained with the best wave level transform decomposing only the first five levels. The decrease in performance is minimal and is confined to the 5%, 10% and 20% case. The achieved reduction in computational complexity on the other hand is significant.

The advantage we draw from the fact that the bottom most decomposition levels do not yield into suitable transform bases is the reduced computational effort. This statement is already pretty satisfying, nevertheless we want to spend some thoughts on the reason why it is like that. As we learned earlier, basis vectors from the bottom of the decomposition tree have a good localization in frequency but are little localized in time, whereas the basis vectors after only a

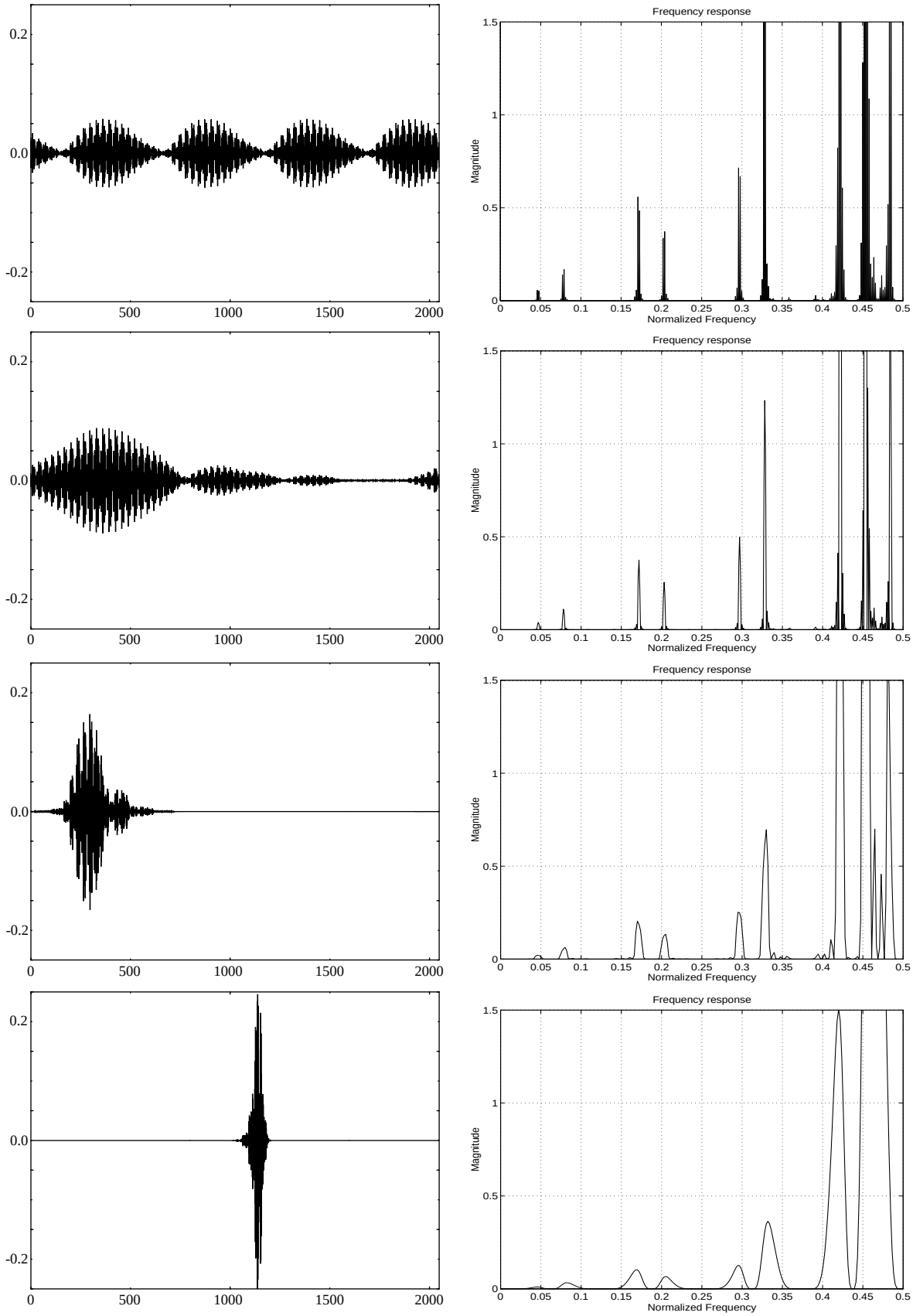


FIGURE 77. On the left basis vectors from the 11th (top), from the 9th, from the 7th and from the 5th (bottom) decomposition level of a wave level transform are illustrated with the corresponding frequency response illustrated on the right.

few iterations of decomposition are less frequency localized for the benefit of a good time localization. To make this more clear we have illustrated the time localization and the frequency localization of several basis vectors from the Daubechies8 wave level transform in figure 77 on page 79. Calculated on a 2048 sample block these are starting from the top a basis vector of the - bottom most - eleventh decomposition level, one of the ninth, one of the seventh and the last one of the fifth. The difference in time location between the basis vector from the eleventh and the basis vector from the fifth decomposition level is obvious. Having in mind the shape of an average speech signal as it is depicted in figure 66 on page 67 it is intuitively understandable that time localized basis vectors are more suited to represent speech signals than the non time localized ones.
Later - when we investigate the performance of the Fourier transform in respect to compression of speech signals - our conclusion will be validated. The basis vectors of the Fourier transform have no time localization at all. Nevertheless the Fourier transform will yield into a good decomposition of the speech signal when the transform block length is small. Because then it is the small block length which assures a sufficient time localization in regard to the complete signal. The same reason caused the good performance of the WLT using the bottom most decomposition level for the transform block length of 256 samples. Even though the basis vectors were not time localized within the transform interval, in respect to the speech signal their time range was limited to the transform block length.

Traditionally the Fourier transform has either been simply an efficient tool for accomplishing certain common manipulations of data or been of intrinsic interest itself (or the related frequency spectrum). Using the Fourier transform for a layered audio compression scheme we combine these two usages. The audio data is manipulated in the Fourier domain in awareness of the related frequency spectrum.
The next step is to carry out the same measurements which we have done for the various wavelet transforms with the FFT (and for the sake of completeness with the DCT). The performance

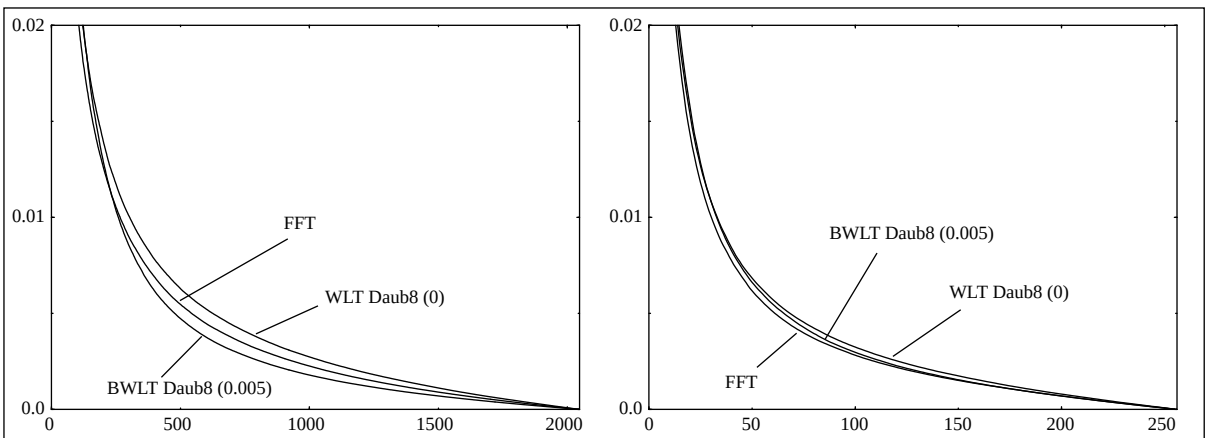


FIGURE 78. The absolute values of the transform coefficients - averaged over many consecutive transformations - of the FFT, the WLT Daub8 (0) and the BWLT Daub8 (0.005) are plotted in decreasing order.

of the FFT - illustrated in figure 78 - is more or less like we predicted earlier. For the large transform block of 2048 samples a representation of the signal through just a few of the highest FFT transform coefficients does not seem to be efficient. A surprise is the excellent result for the 256 sample block case. In concentrating the power of the signal within a small number of

coefficients the FFT easily outperforms the WLT. The exact results are given in table 14

transform	5%		10%		20%		30%		50%		CPU
WLT Daub8 (0)	74.01	70.86	85.74	85.03	94.02	94.01	97.09	97.05	99.30	99.29	1758
BWLT Daub8 (.005)	79.18	70.76	90.69	85.84	96.91	94.76	98.67	97.60	99.72	99.46	2096
FFT	79.35	75.28	89.08	87.86	96.70	95.44	97.99	97.81	99.54	99.47	617
DCT	79.08	75.88	89.01	88.38	95.70	95.75	98.00	98.03	99.55	99.56	1348

TABLE 14. The percentile amount of the total signal power contained in the highest 5%, 10%, 20%, 30% and 50% coefficients per block. The results for FFT and for the DCT are compared with earlier measurements

together with some earlier taken measurements. The FFT on small transform block performs way better than all the variations of wavelet transforms we have investigated so far. In particular we can observe that the FFT captures already very much signal information within the highest 5% or 10% of the transform coefficients. This leads to the assumption that the FFT may be well suited for extremely high compression rates. Later we will introduce an FFT based encoding scheme that confirms this assumption. The only competing transform is the WPT or the WLT using the Vaidyanathan24 filter. But this transform uses more than twenty times the computation time of the FFT. As an example, even the most efficient implementation of the WLT algorithm has not been fast enough to transform 8 kHz sampled speech in real-time on a SUN sparc station 10 with the Vaidyanathan24 filter.
To be complete the performance of the discrete cosine transform (DCT) was included in table 14 on page 81. The discrete cosine transform is said to outperform the fast Fourier transform in terms of speech compression [42]. The results for the transform of 256 sample blocks verify this statement. The basis vectors of the DCT have - like the FFT basis vectors - absolutely no time localization within the transform interval. This explains the similar behaviour of DCT and FFT in regard to the transform block length.

4 Competing subjectively

The measurements taken so far resulted in a lot of different numbers and a couple of nice graphics. How steep the decline of a plotted graph was and how precise the statements about the capture of signal power have been, can at most impress our eyes and our brain - not our ears. Even though all numerical results forcible suggest that especially transform X will perform outstanding, the human ear - little impressed - may favor transform Y with the not very scientific proof: it simply sounds better.
The amount of signal power that is captured by a certain number of coefficients in respect to the chosen transform basis is definitely a rational criteria to judge about the remaining amount of signal information. But it gives no evidence about the importance of the lost information for the audible quality of the signal. The best way to prove our theoretical considerations is to use our ears - or better the ears of a number of listeners - as a jury.

The author wrote a small audio tool that allows to perform all transforms we dealt with either to a prerecorded or to real-time audio signal. We give a brief introduction to the possibilities offered by the tool and discuss the auditory results subsequently.

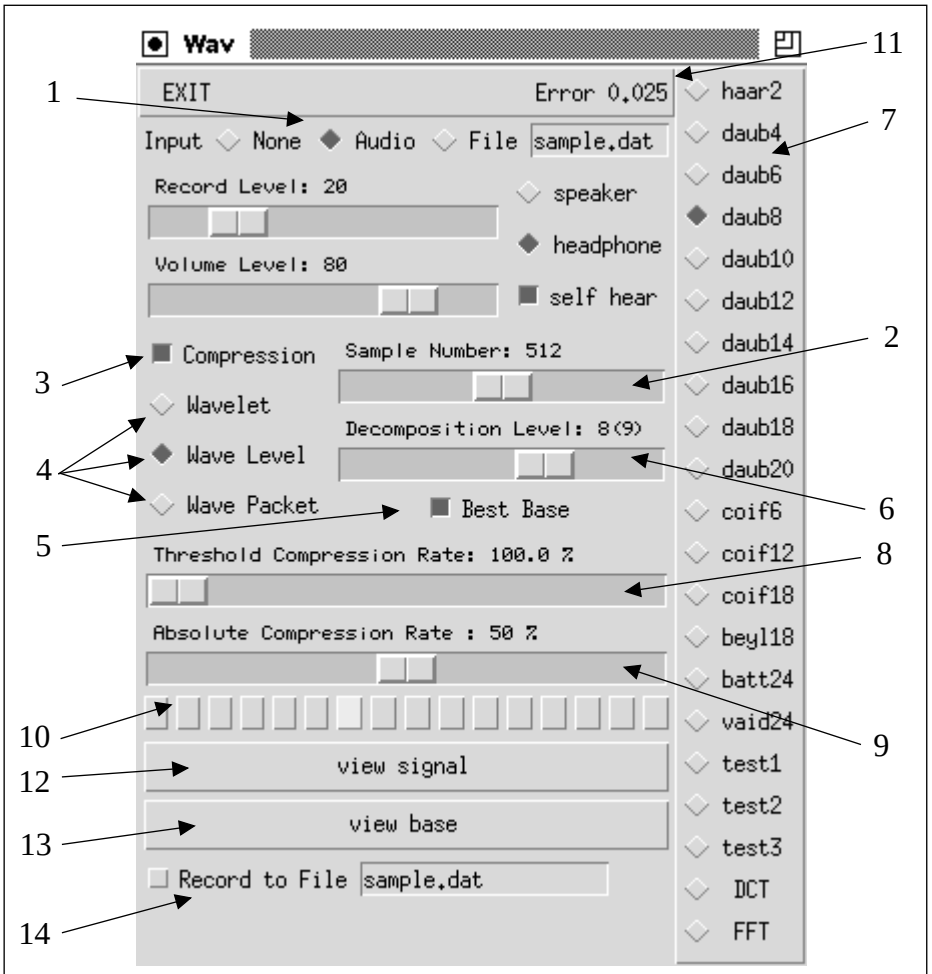


FIGURE 79. the user interface of WAV

In figure 79 the user interface of this tool called WAV is shown. The *input*-buttons (1) determine whether the audio samples are read from a file or taken from the audio device. Only files containing an audio signal in 8 bit µlaw PCM sampled at 8 kHz can be played back. The control elements below for configuring the audio device are self-describing. The *sample number*-slider (2) allows to choose the number of samples that make one transform block. It is a constraint of all possible transforms that the block size is a power of two. The *compression*-button (3) activates and deactivates the transform and the compression of the audio signal. The buttons (4) + (5) determine together with the slider (6) and the button bar (7) the performed transform. Either a wavelet transform (DWT), a wave level transform (WLT) or a wave packet transform (WPT) decompose each transform block using the number of decomposition levels as determined by the slider (6). The number in brackets informs about the possible decomposition depth. If a higher level of decomposition is chosen the transform is done down to the bottom most level. Selecting the *best base*-button (5) activates the best basis search for each transform. The QMF filter for the three transforms DWT, WLT and WPT is chosen on the button bar (7). Choosing either FFT or DCT makes all other settings insignificant. In this case either the fast Fourier transform or the discrete cosine transform is performed.

Unless some of the transform coefficients are discarded none of the transforms will alter the output signal. What we call compression is (up to now) nothing but discarding of coefficients. The *compression*-sliders (8) + (9) allow position independent and the button bar beneath (10) allows position dependent discarding of coefficients. The *threshold compression* (8) discards all coefficients below a certain threshold which results in a varying compression rate. The *absolute compression* (9) discards a constant percentage of the lowest coefficients. With the button bar (10) the user is able to experience how position dependent discarding of coefficients affects the

output signal. The 16 buttons together correspond to the complete transform block. Activating the first button on the left will set one sixteenth of the transform coefficients to zero. In order to use this kind of compression in a meaningful way, it is necessary to know how the transform coefficients are arranged within the transform interval. When a signal is compressed its reconstruction from the remaining transform coefficients differs from the original signal. In the top right corner of the user interface the error label (11) gives a coarse measurement of a mean error history. We calculate the mean error with

$$e = \frac{1}{n}\sum_{k=0}^{n-1} |u_k - \hat{u}_k|$$

where n is the number of samples, u_i the i-th coefficient of the original sample vector and $\hat{u}_i$ the i-th coefficient of the reconstructed vector. Since this value varies too quickly with every other transformation, we rather display an error value that is an average over the history of the mean error with

$$e_h = 0,99e_h + 0,01e$$

The error label (11) shows the value e_h multiplied with a scaling factor of 100.

The two big buttons (12) + (13) open two separate windows. The *signal viewer* (12) is a small strip chart for a real-time visualization of either the signal coefficients - button (2) deactivated - or the transform coefficients - button (2) activated. In figure 80 two screen shots of the signal

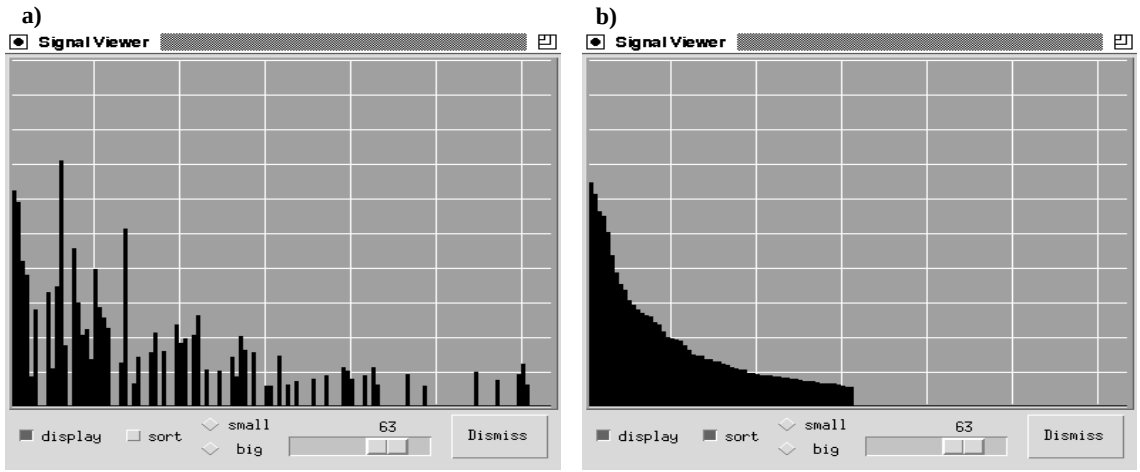


FIGURE 80. The signal viewer shows the coefficients of a Daubechies8 wavelet transform with 50% absolute compression (a) without and (b) with activated sort option.

viewer are shown. It should be noted that the strip chart depicts only absolute values. The *base viewer* (13) is of relevance for the wave packet transform. On one hand it allows to illustrate the wave packet bases that are result of the best bases search. On the other hand it is the means for explicitly selecting a certain basis for a wave packet transform and use it as fixed basis. See figure 81 on page 84 for an example screen shot. The same illustration with rectangles was already used when the wavelet theory was introduced, see “The wave packet transform” on page 58The two top rectangles correspond to the first decomposition level and from the top to the bottom seven decomposition levels are shown. The coefficients of the eight decomposition level are used when none of the above levels was selected. With simply clicking on the appropriate rectangles the user can compose an arbitrary basis for the wave packet transform. The library of bases of the wave packet transform includes the transform basis of both the wavelet

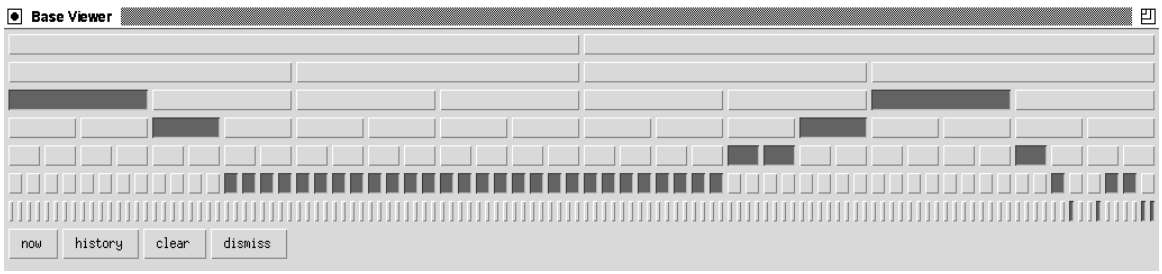


FIGURE 81. The base viewer used to select a certain basis from the wave packet library of bases. This decomposition has eight levels. The bottom most level is not shown, but is always used when no higher level was selected.

transform and the wave level transform. The selected basis of figure 82 corresponds to a wavelet basis at the sixth decomposition level. Using this selection as a fixed basis for the wave packet transform calculates exactly the same transform as a simple wavelet transform down to the sixth decomposition level. A wave level transform is calculated when a complete row of

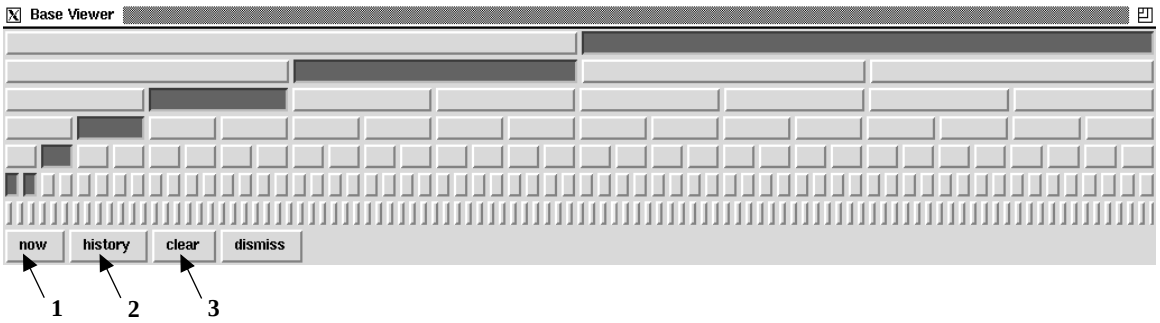


FIGURE 82. The base viewer used to select the wavelet basis at the sixth decomposition level.

rectangles in the base viewer window is activated. In respect to computational complexity these simulations are much more expensive than the original transform algorithms.

When wave packet transforms are performed in the best base mode, the buttons (1) + (2) allow to keep track on the best base choice. Pressing the *now*-button (1) captures and displays the chosen basis of the last wave packet transform. With the *history*-button (2) the frequency each of the rectangle components was chosen to belong to the best basis is illustrated. According to the number of times recently used the rectangle is colored in different red shades (see

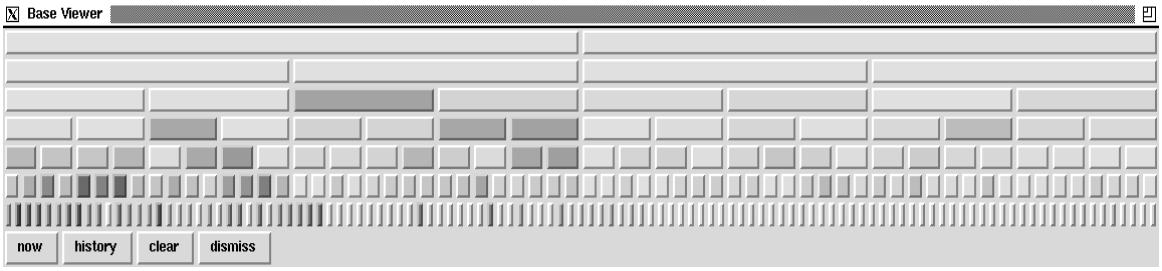


FIGURE 83. The base viewer illustrates the frequency with which a rectangle was chosen to belong to the best basis.

figure 83). This will allow statements about bases that are a preferred choice from the library of wave packet bases. The clear button (3) resets the state of the base viewer - all rectangles are deactivated.

The auditory results fulfilled the expectations we had after the investigations about the capture of signal power, but were also surprising in some points. It is very difficult to make any statements about the quality of the audio signal in the self hear mode. The test persons - who simul-

taneously heard their own voice and the compressed signal reconstruction via the headphones - always overestimated the intelligibility of the audio signal. Therefore we used a prerecorded speech sample of the authors voice and taped the results from applying various transforms with different absolute compression rates. In the test scenario each of the resulting audio files was played back several times to give the listening test persons time to compare the different transforms. The competing transforms were judged in three independent runs at three different compression ratios.

As a rule over the thumb it can be said that the transforms that capture more signal power within the highest coefficients result in better signal reconstructions. But in most cases - especially when these difference is small - the improvement was not noticeable. The best wave packet transform for instance did not perform better than the wave level transform. To make this more immediate we tried the following: After using the best wave packet transform for a short moment an arbitrary best basis was captured and retained for consecutive transforms. In most cases there was no audible decrease in audio quality. The discrete wavelet transform produced for all compression ratios unmistakable the worst results of all tested transforms. For the remaining wave level and wave packet transforms the result was - in respect to computational constraints - very promising. Even though the objective comparison before predicted better results for longer quadrature mirror filters (which enormously increase the computational complexity), the auditory experiments did not verify this. The Daubechies8 filter performed quite well and even the long Vaidyanathan24 filter, which is said to be optimized for speech compression [48], had almost never evidently better results. Anyhow, with a computational complexity that - for an average work station - does not allow its calculation in real-time the Vaidyanathan24 filter is of no practical use for our audio codec. In table 15 the results of the auditory test series are briefly summarized. The length of the transform block was 256 sample

transform	%	quality[a]	intelligible	description
DWT Daub8 (0)	50	++	yes	slight scratching
	20	-	hardly	scratching, hissing, hollow
	5	---	no	noisy
WLT Daub8 (0)	50	+++	yes	perfect
	20	++	yes	hissing, some bell like sounds
	5	-	almost	scratching, hissing, some bell like sounds, hollow
BWPT Daub8 (0.005)	50	+++	yes	perfect
	20	++	yes	like WLT Daub8 (0)
	5	-	almost	like WLT Daub8 (0)
FFT	50	+++	yes	perfect
	20	+	yes	many bell like sounds
	5	0	yes	strong bell like sounds, hollow

TABLE 15. The quality of the reconstructed audio signal after discarding all but 50%, 20% and 5% of the coefficients for certain transforms. The same measurements with the Vaidyanathan24 filter were slightly better for the DWT and almost identical for the WLT and the BWPT.

a. subjective judgement of quality by the test persons

values.

As a first conclusion of the obtained results we will now concentrate on two transforms: the wave level transform with the Daubechies8 filter and the FFT transform. With the Daubechies8 filter the WLT offers a fairly good compression while the computational effort remains accept-

able. The FFT is interesting because it seems to be extremely well suited to allow intelligible signals at very high compression rates.

5 Compressing the transform coefficients

We were talking about compression ratios although we actually talked about the percentage of not discarded transform coefficients. In this section we learn about this difference and about the difficulties involved in achieving a real compression. The real compression ratio ρ [47] for a block of audio samples is calculated with:

$$\rho = \frac{\text{number of bits needed to store the compressed block of audio samples}}{\text{number of bits needed to store the uncompressed block of audio samples}}$$

For our experiments the used audio samples are 8 bit μlaw PCM values which are simply stored one after one in a block of audio samples. In this case the number of bits needed to store the uncompressed block of n audio samples is $8n$.
In order to have a uniform scaled representation of the audio signal the 8 bit μlaw PCM audio samples are transformed linear into floating point values between -1.0 and 1.0. Applying one of the introduced transforms towards a block of n audio samples yields into a block of n transform coefficients in floating point representation. A certain amount of these transform coefficients is discarded. Now the value of the remaining coefficients and their position within the block needs to be stored with less than $8n$ bits. With the idea of a layered encoding in mind this storage should furthermore allow to separate the encoding into pieces of different priority.

Efficient coding of the coefficient positions. For each of the kept coefficients the position has to be transmitted along with the value. For a block of n transform coefficients there are two ways to do this:

1. A bit field of n bits informs which coefficients are kept and which coefficient were discarded. It is immediate that the coefficients have to be arranged in the same order as the set bits in the bit field. This approach always uses the same amount of bits for positioning regardless from the number of kept coefficients. Thus it is inefficient when the number of kept coefficients is very small. For a layered encoding scheme each layer needs its own bit field of length n.
2. Specifying the position of a coefficient within a block of n transform coefficients can be done with $\log_2 n$ bits per coefficient. For large block sizes this approach is only attractive if the number of coefficients is very small. It is a big advantage that the storage order of the coefficients remains a free choice. This makes it possible to store the coefficients sorted by their value - a chance for further compression as we will see later. There is no extra information needed for layered encoding schemes.

Efficient coding of the coefficient values. Storing the transform coefficients in their present representation as float values will definitely not result in any compression. The two possibilities for a more compact storage of these values are described here. The coefficients that are subject to transmission are a certain amount of those with the highest absolute value. Therefore both approaches start with an array that contains the absolute value of the coefficients while the sign is stored somewhere else. For layered codecs the described procedure is applied to the coefficients of each layer separately.

1. Suppose max is the biggest and min the smallest value in the array of coefficients. The straight forward approach is to map the range $dif = max - min$ linear onto an integer number i of b bits. Then for some coefficient c this mapping is calculated with:

$$i = \frac{c - min}{dif} 2^b$$

Of course the offset values min and dif or min and max have to be kept as well. Extending the integer number by a proceeding bit allows to specify the sign of the respective coefficient. In case the coefficient was negative we simply add 2^{b+1} to the number i.
The following small example shows that the linear quantization of transform coefficients into an b-bit integer number yields into a poor exploitation of the 2^n possible code words. Top left in figure 84 the sorted coefficients of an averaged WLT for a transform block length

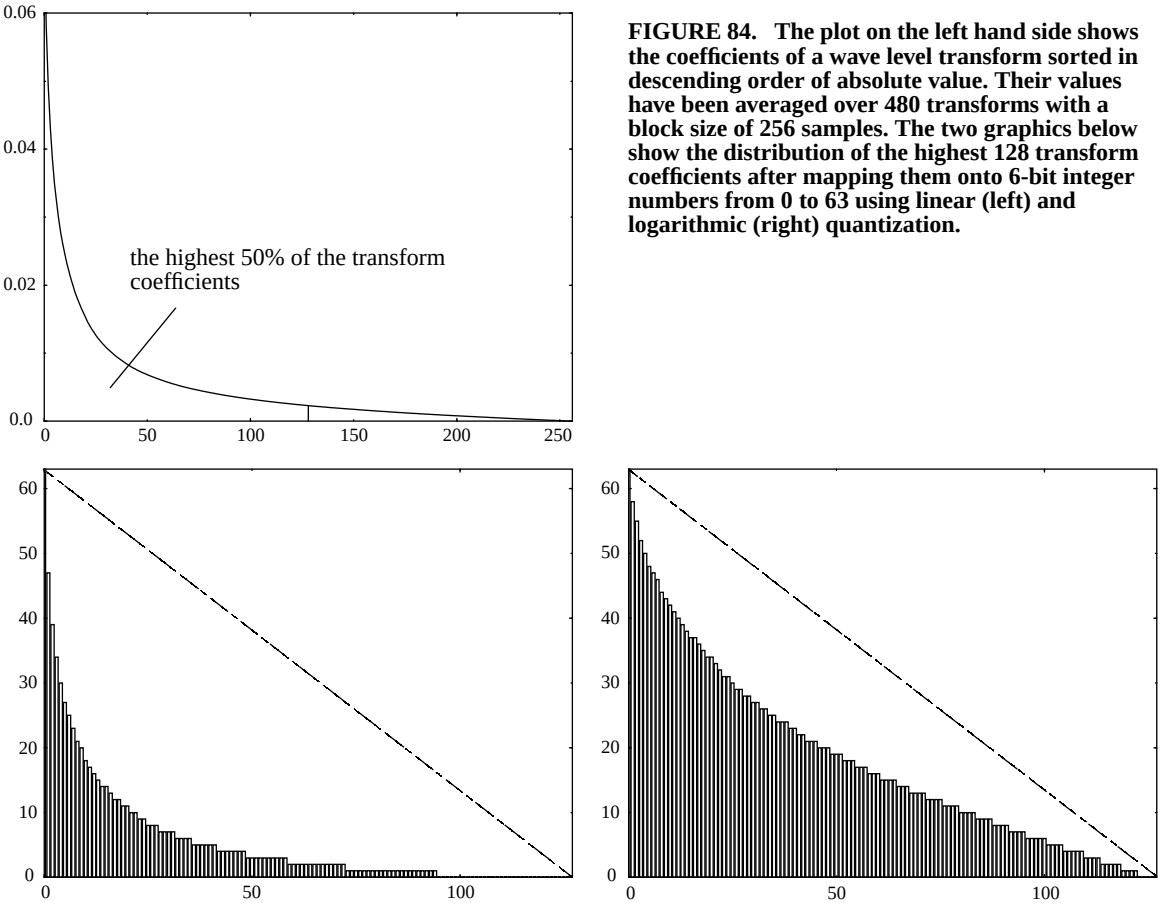


FIGURE 84. The plot on the left hand side shows the coefficients of a wave level transform sorted in descending order of absolute value. Their values have been averaged over 480 transforms with a block size of 256 samples. The two graphics below show the distribution of the highest 128 transform coefficients after mapping them onto 6-bit integer numbers from 0 to 63 using linear (left) and logarithmic (right) quantization.

of 256 samples are shown. Applying the introduced method of linear quantization with 6 bits towards the highest 50% of the coefficients distributes the 128 quantized values rather irregular over the 64 possible code words. The graph in the bottom left of figure 84 illustrates this distribution. In the right picture this distribution is more uniform. It is much closer to the ideal distribution that is denoted by the dotted line in both graphics. Here we used a logarithmic quantization to map the coefficients onto the integer numbers.
Let max contain the natural logarithm of the highest coefficient and consequently be min the natural logarithm of the lowest coefficient from all that are subject to quantization. With $dif = max - min$ each coefficient c is mapped onto an integer number i of b bits using

the formula

$$i = \frac{\ln(c) - min}{dif} 2^b$$

Is is immediate that again the offset values min and dif or min and max are necessary for the inverse calculation. To record the sign of each coefficient the same method as before may be used.

When the quantized transform coefficients are arranged in a sorted order transmitting only the small difference between two consecutive values can decrease the needed bit rate. Three bits per coefficient would be enough for our example since the biggest variation between two successive numbers is seven.

2. This method introduced here depends on the approach used to specify the coefficient positions. It only works with the coefficient values arranged in a sorted order. Thus the bit field approach for positioning is out of the question.

   The idea is to approximate the slope of the graph of the sorted transform coefficients. Using some interpolating functions that describe the curve drawn by the coefficients in sorted order is one possibility. We instead exploit the already quantized and sorted array of transform coefficients. Selecting a number of representative points of support reduces the amount of storage data enormously. In figure 85 on page 88 a very coarse approximation with four supporting point is illustrated. The points are chosen at fixed positions at 10%, 20%, 40% and 70% of the interval length starting at zero. The highest (at 0%) and lowest (at 100%) coefficient can always be reconstructed from the offset values. Fixed positions for the supporting points do not always give the best results. In the example we reach a better approximation if the first interpolating point would be at 5% and not at 10%. A more sophisticated approach would flexible adapt the actual positions of these points to the signal so that the approximation error is minimized. This is subject of further investigation.

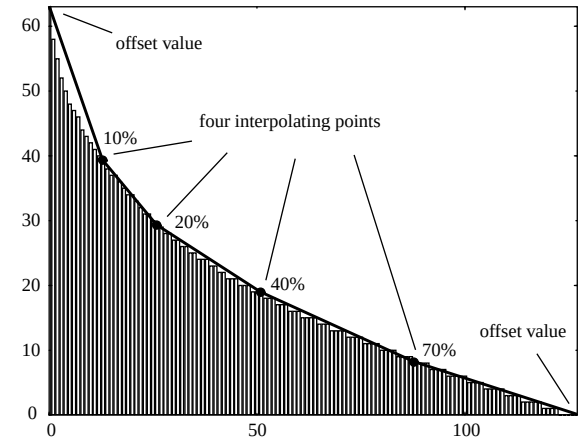


FIGURE 85. After quantizing the highest 128 transform coefficients logarithmic onto 6-bit integer numbers from 0 to 63 the resulting graph is interpolated with four supporting points. These are at fixed positions at 10%, 20%, 40% and 70% of the interval length.

6 The new audio codecs

Here we finally present three audio codecs that are result of our investigations. All three audio encoders have been implemented and perform extraordinary well in real-time on audio signals sampled at 8 kHz. Our research group is presently applying these coders towards high-quality audio with sampling rates up to 32 kHz.

6.1 One layer wave compression

This audio codec is completely independent from the chosen transform basis. However, with the given computational constraints it looses real-time capability on a SUN sparc station 10 if

the length of the quadrature mirror filter for a WLT exceeds 12. Together with either a WLT using the Daubechies8 filter at eight decomposition levels or a simple FFT the coder seems to perform at its best. For the WLT this performance is independent from the transform block size, while the results for the FFT get worse for transforms with more than 512 samples (at a 8 kHz sampling rate).
The audio codec provides an audio stream at a flexible bit rate ranging from 64 kbps down to less than 6 kbps. It exploits a combination of these three ideas:

1. For specifying the coefficient positions the bit field approach is used.
2. A free eligible percentage of the highest coefficients is logarithmically quantized into an integer number with a selectable number of bits.
3. Position dependent discarding of coefficients is offered. Then less bits are necessary to specify the positions. For high compression ratios it makes sense to omit the high frequencies components of the audio signal. Then the highest valued coefficients are not selected out of all coefficients (range 0) from the transform block but of those from the left half (range 1) or even only from the left most quarter (range 2) of the transform coefficient array as described in figure 86 on page 89. This is because for all wavelet transforms the resulting coefficients are arranged in the array in a way that more left ones address lower frequency components and more right ones address higher frequency components of the signal.

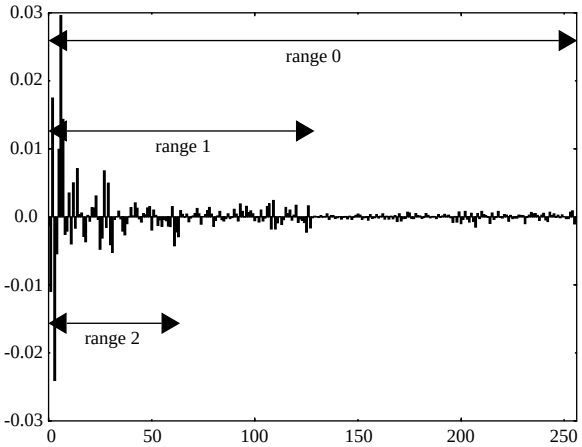


FIGURE 86. A WLT transform using the Daubechies8 filter and 8 decomposition levels of an array of 256 audio samples. The range factor determines the section where the highest valued coefficients are chosen from.

The audio coder offers three ways to vary the bit rate:

- the percentage of kept coefficients
- the number of bits for storing each coefficient
- the range the coefficients are selected from.

In table 16 some example calculations with different parameters are listed.

%	b	r	quantized coefficients	offset values	position field	total	ρ	bit rate
50	5	0	256 * (5+1) = 1536 bits	2 * 8 = 16 bits	512 bits	2064 bits	0.504	32.25 kbps
30	5	0	154 * (5+1) = 924 bits	2 * 8 = 16 bits	512 bits	1452 bits	0.354	22.69 kbps
30	3	0	154 * (3+1) = 616 bits	2 * 8 = 16 bits	512 bits	1144 bits	0.279	17.86 kbps
20	3	0	102 * (3+1) = 408 bits	2 * 8 = 16 bits	512 bits	936 bits	0.229	14.63 kbps
20	3	1	102 * (3+1) = 408 bits	2 * 8 = 16 bits	256 bits	680 bits	0.166	10.63 kbps

TABLE 16. When coding audio signals in blocks of 512 samples the following bit rates are achieved by varying the percentage (%) of kept coefficients, the number of bits (b) for storing each coefficient and the range (r) the coefficients are selected from. The reference for the compression ratio ρ are 8 bit μlaw PCM values with a bit rate of 64 kbps.

%	b	r	quantized coefficients	offset values	position field	total	ρ	bit rate
10	3	1	51* (3+1) = 204 bits	2 * 8 = 16 bits	256 bits	476 bits	0.116	7.44 kbps
10	3	2	51* (3+1) = 204 bits	2 * 8 = 16 bits	128 bits	348 bits	0.085	5.44 kbps

TABLE 16. When coding audio signals in blocks of 512 samples the following bit rates are achieved by varying the percentage (%) of kept coefficients, the number of bits (b) for storing each coefficient and the range (r) the coefficients are selected from. The reference for the compression ratio ρ are 8 bit μlaw PCM values with a bit rate of 64 kbps.

6.2 Multi layer wave compression

Here we present a layered audio encoding scheme with three layers. However, two, four, five or more layers can be realized in the same manner. Again the coder is independent from the chosen transform basis.

A possible approach is to have the bottom most layer containing the highest 10%, the second layer the next highest 20% and the last layer another 20% of the coefficients. The exact percentages are not of immediate interest here and could be arranged to yield the quality layering intelligible/good/perfect. Each layer is organized in the same way. The position of every coefficient is specified with $\log_2 n$ bits where n is the transform block length. An extra bit indicates the sign of each coefficient. The used approach for positioning allows us to arrange the absolute values of the coefficients in decreasing order. As before logarithmic quantization is used to map the values onto an integer number of b bits. Since the coefficients are in sorted order interpolation can be used to reduce the amount of storage data. In figure 87 on page 91 the method of quantizing and interpolation is shown for two layers. The original sorted transform coefficients and their segmentation into three layers are illustrated in the upper graph. The the first two layers are quantized separately into 3-bit integer numbers. The somewhat arbitrary approximation with four supporting points makes it clear that the interpolation method yields into more accurate results than absolute quantizing. This is because linear interpolation brings back all the intermediate value steps between the fixed number of quantization levels.

layer	%	b	i	supporting points	offset values	sign	positions	total
0	10	3	2	8 * 3 = 24 bits	2 * 8 = 16 bits	51 bits	51 * 9 = 459 bits	550 bits
1	20	2	2	8 * 2 = 16 bits	2 * 8 = 16 bits	102 bits	102 * 9 = 918 bits	1052 bits
2	20	2	2	8 * 2 = 16 bits	2 * 8 = 16 bits	102 bits	102 * 9 = 918 bits	1052 bits
				compression ratio ρ	0.648	**bit rate**	41.47 kbps	2654 bits

TABLE 17. For a block of 512 audio samples encoded with three layers the resulting bit rate is calculated. For each layer the percentage (%) of kept coefficients, the number of bits (b) for storing each supporting point and an interpolation factor (i) are eligible. The interpolation factor 0 means no interpolation, then all quantized coefficients are kept. A factor 1 corresponds to an approximation with 16 supporting points, with factor 2 only 8 points are used and for factor 3 the interpolation is done with 4 points.

Some calculations about the bit rate which was achieved using this encoding scheme are presented in table 17. Compared to the one layer approach we need less bits for quantizing of the coefficients. The reason is that the complete value range of the highest 50% of the coefficients is divided in three section of which each is quantized separately with its own offset values.

Although the resulting bit rates of this encoder are not as low as the rates of the codec introduced before, it offers a very nice property. The codification of layer 0 is the most important part of the encoding. Adding layer 1 and furthermore layer 2 increases the quality of the audio signal but in worst case situations a signal reconstructed using only layer 0 is acceptable. For

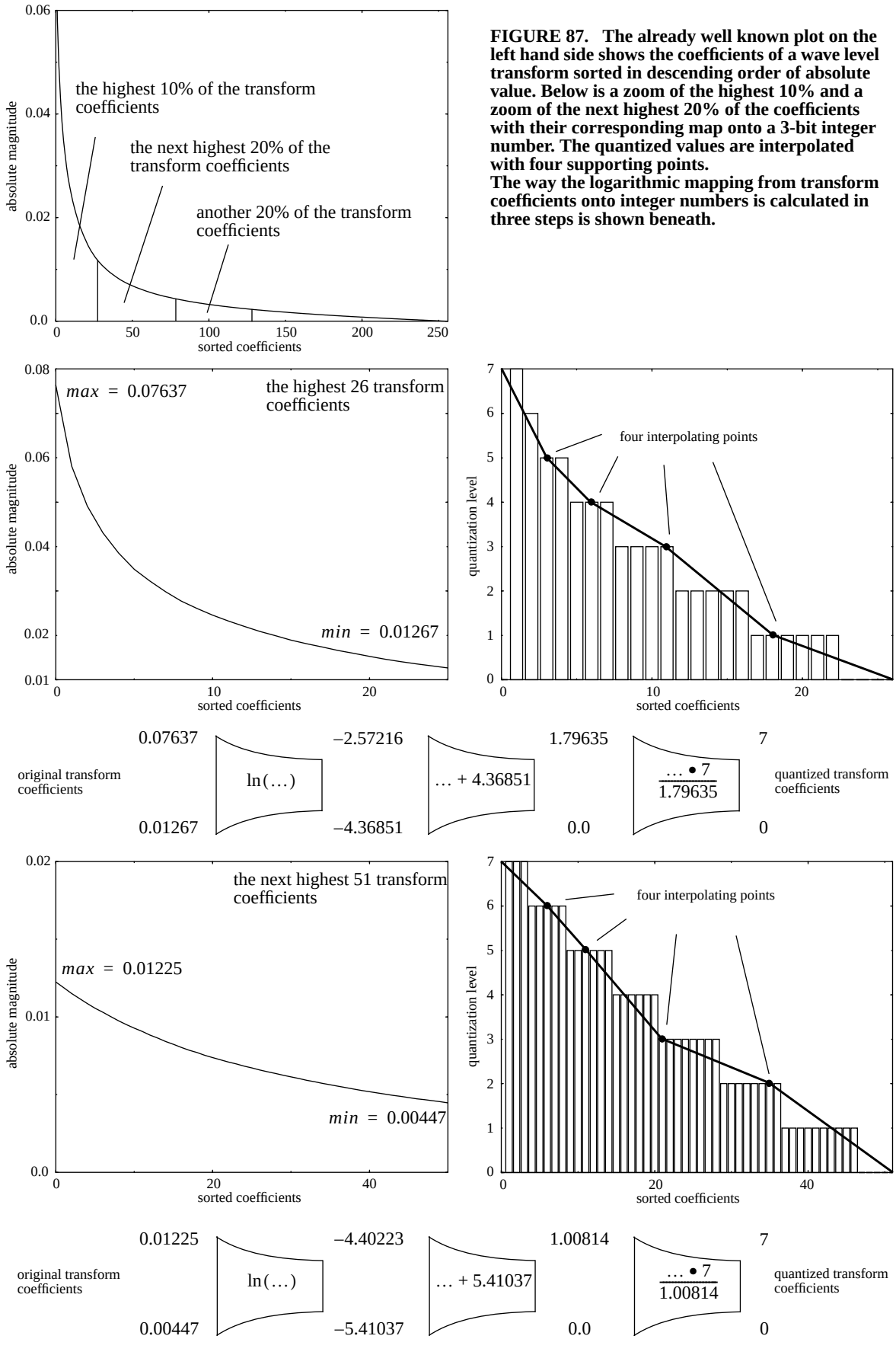


FIGURE 87. The already well known plot on the left hand side shows the coefficients of a wave level transform sorted in descending order of absolute value. Below is a zoom of the highest 10% and a zoom of the next highest 20% of the coefficients with their corresponding map onto a 3-bit integer number. The quantized values are interpolated with four supporting points.
The way the logarithmic mapping from transform coefficients onto integer numbers is calculated in three steps is shown beneath.

lossy data transmission this yields to an almost natural approach that protects this layer with a higher priority against loss than the remaining two layers.

6.3 High FFT compression

The convincing performance of this encoding scheme was a big surprise for all of us. Rainer Storn of SIEMENS, whose dissertation was about various FFT algorithms, said he never heard of an approach that used the fast Fourier transform the way we do it for audio compression. For us it was surprising to see that such a simple and straight forward exploitation of a FFT algorithm would perform that well. In general the same methods as before are used to compress the transform coefficients that represent the audio signal. The difference is that we take advantage of some special characteristic of the fast Fourier transform that allow further compression. We first introduce this special characteristic and explain the codec later.
The discrete Fourier transform and its inverse from a vector $\bar{h}$ with n coefficients into a vector $\bar{H}$ with as well n coefficients are defined as:

$$H_m = \sum_{k=0}^{n-1} h_k e^{-2\pi ikm/n} \qquad h_k = \frac{1}{n} \sum_{m=0}^{n-1} H_m e^{-2\pi imk/n}$$

In our case the vector $\bar{h}$ represents the audio signal $h(t)$ sampled in regular time steps Δt. For a vector with n coefficients the length of the sample interval is $n\Delta t$ and $h_k = h(t_k)$ with $t_k = k\Delta t$ for $k = 0, 1, \ldots, n-1$. Consequently computing the Fourier transform $\bar{H}$ of $\bar{h}$ results into the frequency representation of audio signal $H(f)$ sampled in regular frequency steps Δf. Then $H_m = H(f_m)$ with $f_m = m/(n\Delta t)$ for $m = -n/2, \ldots, n/2$. The extreme values of m correspond exactly to the Nyquist critical frequency range $-f_c$ to f_c. If you are really on the ball, you will have noticed that there are $n+1$ and not n values of m. It turns out that the two extreme values of m are equal - this reduces the count to n.
Up to now we have taken the view that the index m varies from $-n/2$ to $n/2$. You can easily see that the discrete Fourier transform is periodic in m with the period n. Therefore $H_{-m} = H_{n-m}$ and with this convention in mind, one generally lets the m in H_m vary one complete period from 0 to $n-1$. Then the m of the H_m's and the k of the h_k's vary over the same range. Following this convention the zero frequency corresponds to $n = 0$, positive frequencies $0 < f < f_c$ correspond to the values $1 \le m \le n/2 - 1$, while negative frequencies $-f_c < f < 0$ correspond to $n/2 + 1 \le m \le n-1$. The value $m = n/2$ corresponds to both $f = f_c$ and $f = -f_c$.
In all these definitions the h_k's and the H_m's are thought to be complex values. In the case of audio signals the function $h(t)$ in the time domain happens to have a special symmetry. It is purely real, meaning $im(h_k) = 0$ for $k = 0, 1, \ldots, n-1$. In the frequency domain these symmetry leads to the properties $im(H_0) = 0$, $im(H_{n/2}) = 0$ and $H_m = \overline{H_{n-m}}$ for $1 \le m \le n/2 - 1$ [31]. Exploiting these characteristics a vector of n real numbers - even though they are thought to be complex numbers to compute a complex-valued Fourier transform - is

mapped onto another vector of only n real numbers. In figure 88 these mapping is illustrated

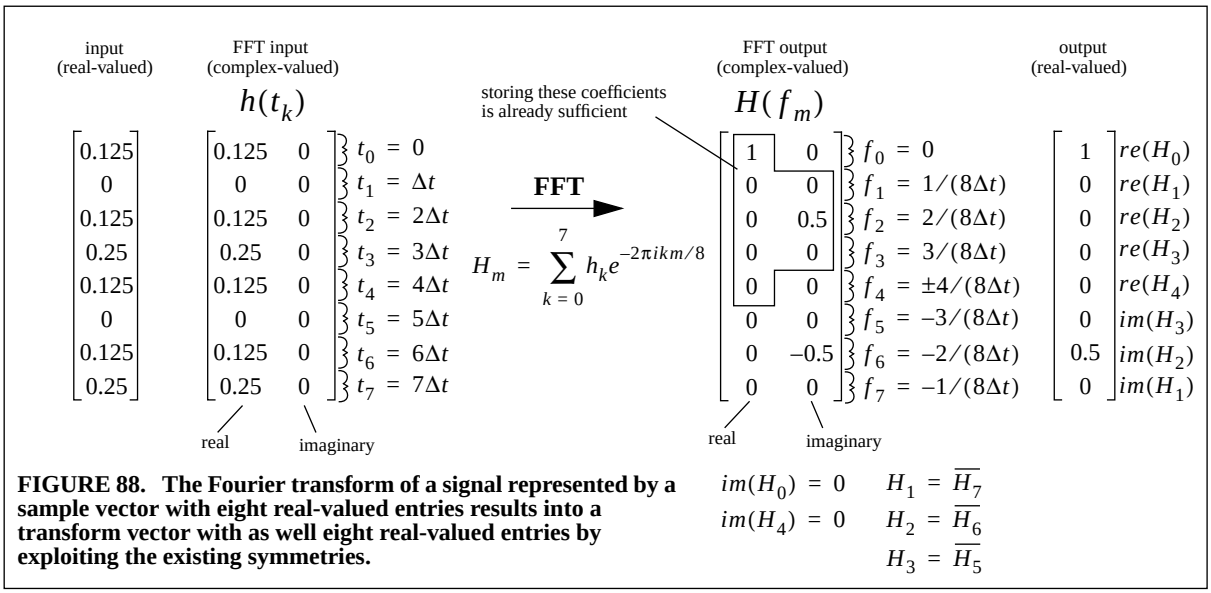


FIGURE 88. The Fourier transform of a signal represented by a sample vector with eight real-valued entries results into a transform vector with as well eight real-valued entries by exploiting the existing symmetries.

with a small example.

For high audio compression the zero frequency f_0 and the Nyquist frequency $f_{n/2} = \pm f_c$ are omitted from the beginning. The contribution of the remaining frequencies f_1 to $f_{(n/2)-1}$ is determined through the complex coefficients H_1 to $H_{(n/2)-1}$, whose real and imaginary components are stored separately in the output array. Discarding transform coefficients the one layer wave compression scheme together with the FFT transform did not take this relation into account. There a coefficient was discarded according to his absolute value so that real components could be omitted while corresponding imaginary parts remained and vice versa. Here we compute the absolute value of each pair of real and imaginary coefficient and keep the highest complex numbers. What we do is transforming the complex numbers H_1 to $H_{(n/2)-1}$ into the polar representation with:

$$amp(H_m) = \sqrt{re(H_m)^2 + im(H_m)^2} \qquad phs(H_m) = \operatorname{atan}\left(\frac{im(H_m)}{re(H_m)}\right)$$

For both - the rectangular and the polar representation - the $(n-2)/2$ complex coefficients are stored in $n-2$ real numbers. The profit we get from either keeping a complex coefficient complete or discarding the real and the imaginary part concerns the positioning. Instead of distinguishing $n-2$ real numbers it is enough to specify the position of $(n-2)/2$ complex coefficients which needs only half the amount of information.
The coder uses the bit field approach for positioning and quantizes the absolute values of the kept complex coefficients - the amplitude - logarithmically into integer numbers of a bits. The phase information of each coefficient is mapped linear onto an integer number of p bits. The

%	a	p	r	quantized amplitude	offset	quantized phase	positions	total	ρ	bit rate
25	4	4	0	64 * 4 = 256 bits	16 bits	64 * 4 = 256 bits	256 bits	784 bits	0.191	12.25 kbps
20	3	3	0	51 * 3 = 153 bits	16 bits	51 * 3 = 153 bits	256 bits	578 bits	0.141	9.03 kbps

TABLE 18. These calculation are based on encoding blocks of 512 audio samples. The percentage (%) of kept coefficients (real and imaginary component count as one coefficient each), the number of bits (a) for storing each amplitude, the number of bits (p) for storing the corresponding phase and the range (r) the coefficients are selected from are the parameters to vary the bit rate.

%	a	p	r	quantized amplitude	offset	quantized phase	positions	total	ρ	bit rate
15	2	3	1	39 * 2 = 78 bits	16 bits	39 * 3 = 117 bits	128 bits	339 bits	0.083	5.30 kbps
10	2	3	1	26 * 2 = 52 bits	16 bits	26 * 3 = 78 bits	128 bits	274 bits	0.067	4.28 kbps
10	2	3	2	26 * 2 = 52 bits	16 bits	26 * 3 = 78 bits	64 bits	210 bits	0.051	3.28 kbps

TABLE 18. These calculation are based on encoding blocks of 512 audio samples. The percentage (%) of kept coefficients (real and imaginary component count as one coefficient each), the number of bits (a) for storing each amplitude, the number of bits (p) for storing the corresponding phase and the range (r) the coefficients are selected from are the parameters to vary the bit rate.

resulting compression ratios yields into audio streams at bit rates as low as are as high 3.28 kbps. In the next section we investigate the quality and the intelligibility of the audio signal at these compression rates.

7 Auditory testing

In our experience the best test scenario for investigations about the distortion and intelligibility of an audio signal compressed through the introduced methods is a full duplex point-to-point communication. To compare the quality of different encoding schemes the play back of prerecorded sound files is well suited. Especially to allow statements about the intelligibility at very high compression ratios this approach fails when the test person has heard the audio samples before.

This and the fact that our encoders were designed to work hand in hand with protection schemes that make the audio stream robust against transmission errors were the motivation for implementing WAU.

7.1 The Wave Audio Unit (WAU)

The Wave Audio Unit is an audio tool for full duplex point-to-point communication. It acts as an experimental test-bed for both investigating the audio codecs introduced above and applying different protection schemes towards them. Rather than explaining implementation details we give an overview about the concepts and the design of the tool. The cooperation of the single

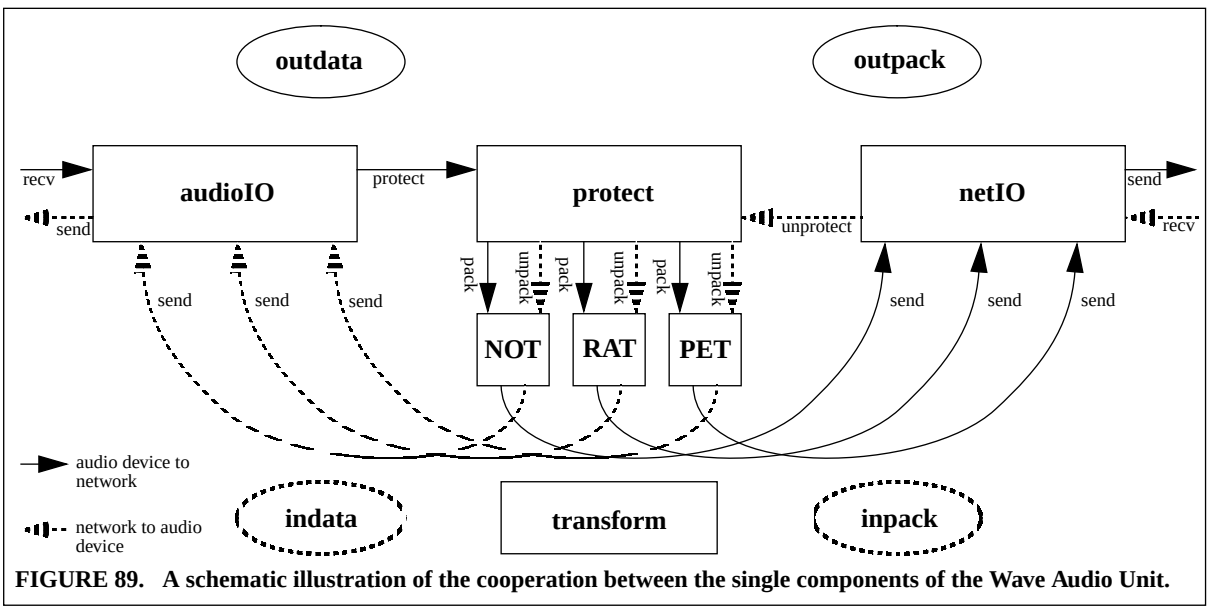


FIGURE 89. A schematic illustration of the cooperation between the single components of the Wave Audio Unit.

components is depicted graphically in figure 89. The user interface directly interacts with each of the components according to the actions by the user. Other events are triggered on arrival of data either at from the network or from the audio device.

Audio device to network: Arriving data at the audio device triggers an event that calls the *recv* function of the *audioIO*. The data is read into the *outdata* object and in case enough data is present the *protect* function of the *protect* object is called. Depending on the chosen protection scheme the *pack* function of either the *NOT*, the *RAT* or the *PET* object is called. The respective objects stand for different schemes of protection. *NOT* - the currently only already implemented object - means no protection, *RAT* incorporates the approach from the MICE project and *PET* applies the message striping scheme towards the encoding. The three objects use the functionality of the *transform* object to carry out the selected transform, compress this representation according to the chosen degree and fill one packet (or more for *PET*) from the *outpack* object with the codification. With the *send* command of the *netIO* object the contents of the *outpack* object are transmitted over the network.

Network to audio device: When packets arrive from the network they are read by the *recv* function of the *netIO* object into the *inpack* object and *unprotect* is called. The *protect* object detects the protection method and calls the *unpack* function of the corresponding object. The packet is uncompressed and - using the methods of the *transform* object - retransformed and stored into the *indata* object. The final call of the *send* function from the *audioIO* object hands the reconstructed audio signal over to the audio device.

The user interface of the Wave Audio Unit - which is very similar to the one of WAV - can be seen in figure 90. The number of various transformations was reduced to those that make sense for compression. The wave packet transform for instance was omitted because of the enormous amount of additional information about the wave packet basis which makes it inefficient for our compression schemes. The remaining adjustments offer the same possibilities to select a transformation as in WAV.
The innovations are the protection control (1) and the net control (2). At this point only the protection less NOT option is available. Future releases will offer the two protection schemes PET and RAT as they were marked out before. The straight forward design of the net control allows unidirectional and bidirectional network connections to another host. The tool receives data on the local port (3) and sends it to the remote host (4).

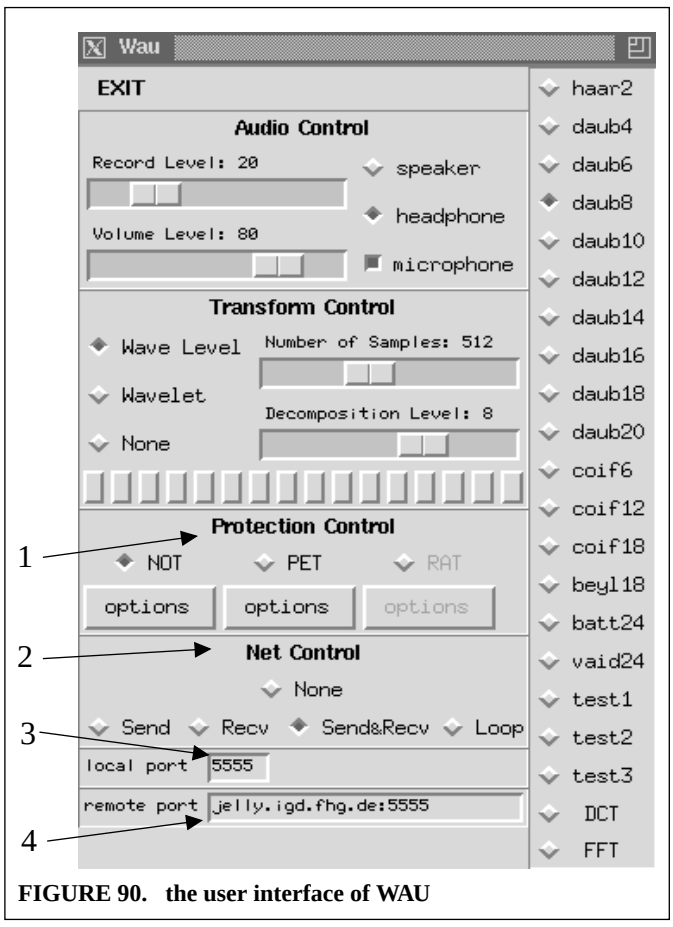


FIGURE 90. the user interface of WAU

The three different codecs that were introduced in the sections before have been implemented exactly as described and are accessible via the options window of NOT (see figure 91). Furthermore standard 8 bit μlaw PCM (1) and an ADPCM coder (2) are offered to allow quality comparisons. The naming of the control panel corresponds to the names used when the codecs were described. For the high FFT compression (3) it may be noted that the left *bit*-slider determines the number of bits used to quantize the amplitude, while the right *bit*-slider is for the number of bits for the phase.
Some statistics are displayed at the bottom of the window (4). These inform about the time interval length and the number of bytes in one packet, the achieved bit rate and the time needed for computation.

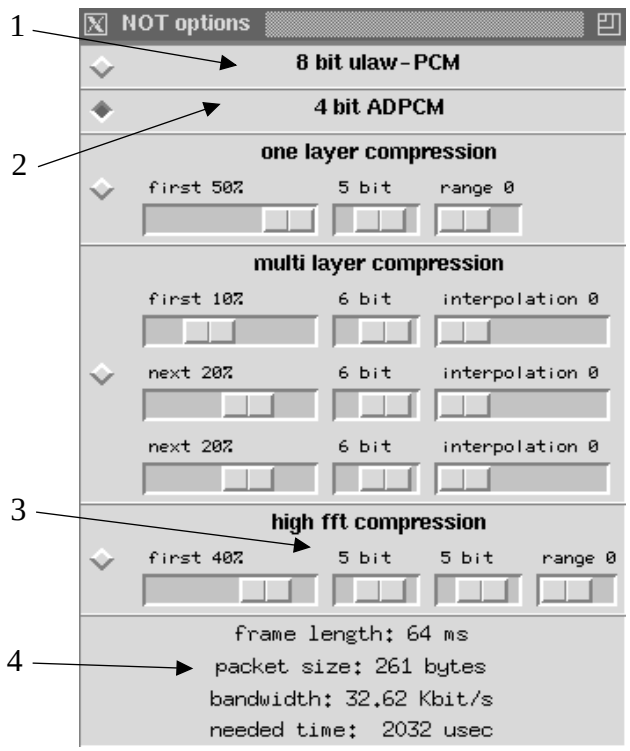


FIGURE 91. The NOT options window

7.2 Auditory results

The quality evaluations were done between two SUN sparc 10 stations using head sets and room microphones. A later release of the Wave Audio Unit does also includes a GSM and a LPC coder at 13.2 kbps and 4.8 kbps bit rate respectively. This enabled the test persons to make a direct comparison between some state of the art speech codecs and our new developed encodings.

one layer compression. This encoder came up with the most impressing results. Together with either the wave level transform using the Daubechies8 filter or the discrete cosine transform there was no audible difference in quality at 32 kbps compared to ADPCM codec with the same bit rate. The computational complexity which is roughly ten times higher is the price we have to pay for the infinitely variable graduation in both bit rate and quality. Increasing reduction of the encoders bit rate results in graceful degradation audio quality with an understandable signal down to 5.44 kbps. In contrast to LPC where the encoded voice sounds like ‘Mickey Mouse‘ is talking with our codec the speakers characteristic voice can be recognized even at high compression ratios. The degradation of the signals quality is - especially for the WLT - very natural. Noise and distortion is added in the same manner as in a walkie-talkie communication that exceeds the maximal distance.

multi layer compression. At a bit rate of roughly 42 kbps this encoding yields into a transparent compression of a 8 bit μlaw PCM signal sampled at 8 kHz. The big advantage are the three independent layers that can be stored, protected and transmitted separately. Using only the bottom most layer results in an already understandable audio signal at 8.59 kbps. When adding layer 1 the necessary bandwidth increases to 25.03 kbps with a signal quality that is already

fairly good. The used transformations were again the wave level transform with the Daubechies8 filter and the fast Fourier transform.

high FFT compression. In respect to intelligible audio signals at lowest bit rates and smallest computational complexity we were able to beat the LPC coder with 4.8 kbps. With almost the same computational power our coder produces (sometimes rather hard) understandable audio signals at 3.28 kbps. Increasing the bit rate to 4.28 kbps results into a throughout intelligible voice signal with strange background noise. It was the test persons experience that the communication was better after the ear became used to this compression method. The distortion is easier to distinguish from the speech signal because of its abnormal nature. The ringing bell-like background noise led a test person to say: ‘Hey, are you calling from outer space?‘.
Generally the LPC coder was the preferred choice - not for reasons of understandability, but because of less disturbing noise. The LPC compressed signal sounds better was a common statement. Nevertheless as soon as the bit rate for high FFT compression was increased to 5.30 kbps and above the audio quality improved a lot - something which is not possible with the LPC coder. Furthermore we are able to apply the same compression algorithm to audio streams with a higher sampling rate - again something which is not possible with the LPC coder.

Summing up the new encoders outperform their rivals through flexibility. When a fine granularity of different bit rates is needed the one layer compression is the first choice. Otherwise the panoply of audio codecs like 8 bit μlaw PCM (at 64 kbps), ADPCM (at 32 kbps), GSM (at 13.2 kbps) and LPC (at 4.8 kbps) may be better. When a layered encoding scheme is needed the multi layer compression is the first choice. However, combining ADPCM or GSM with LPC simulates a two layer encoding, that allows the reconstruction of the audio signal at two quality levels using either the one or the other layer. When compression at lowest bit rates is needed the high FFT compression is an alternative to LPC when some flexibility in bit rate is desired. Two very important advantages of our codecs are independence from the sample rate and unlimited capability to with any kind of audio signal. Moving towards higher sampling rates and/or music signals all three coders outperform ADPCM, GSM and LPC which have been designed to compress speech signals at 8 kHz.

VII

Résumé

1 Conclusion

This thesis addressed quality orientated improvements for multimedia connections over packet switched and lossy networks. The problems involved in establishing real-time communication over networks such as the Internet have been investigated and the definite network characteristics that cause these problems have been clearly marked out. The quality of audio communication essentially depends on the number of packets lost and on the variation in packet arrival times. Efficient mechanism to minimize the impact of delay jitter have already been proposed in literature, whereas dealing with packet loss remains an active research area.

The measurements about the packet loss rate for audio streams over the Internet showed that the number of consecutive lost packet usually is small. This rigorously proved that open loop mechanisms that add redundancy on the sending side are suited to cope with the loss of information. We presented two transmission concepts that overcome these network limitations using forward error correction schemes.
The ‘piggyback protected transmission‘ was introduced - a resilient scheme that has already showed its usefulness in improving full duplex audio communication. The ‘priority encoded transmission‘, which had never been applied to audio streams before, was examined for its capability in protecting the transmission of audio data over lossy networks.
We showed that for time critical point-to-point communication the comparatively simple ‘pick-a-back protected transmission‘ is a better choice than ‘priority encoded transmission‘. In a broadcast scenario on the other hand where large delays are acceptable the PET approach will yield in better results because of its robustness against long packet loss periods and its capacity to transmit to receivers with widely different network bandwidth.

In order to apply the ‘priority encoded transmission‘ towards audio streams, it was necessary to develop a layered audio encoding scheme. A major part of thesis is concerned with discussing and analyzing different transformations of an audio signal in respect to time and frequency. Finally we are able to present an audio codec that we have developed from scratch and that yields into a compressed and layered representation of the audio signal. In contrast to common standard codecs this encoding scheme is well suited to work together with PET.
Furthermore we demonstrated how our new encoding scheme improves the performance of the ‘piggyback protected transmission‘. Through diminishing the redundancy in the redundant information a better audio quality can be achieved in case of isolated packet losses.

2 Current work

The Wave Audio Unit (WAU) - written to allow quality test for the developed audio codecs and to investigate the improvement in audio quality through different protection schemes - is currently enhanced for allowing various sampling rates. We look into robust transmission of audio sampled at higher rates with a higher frequency bandwidth for high-quality audio communication. The promising results of our audio encoding schemes for 8 kHz sampled audio want to be validated with sampling rates up to 32 kHz. A mechanism as proposed in literature to eliminate the impact of delay jitter is about to be implemented. Furthermore we look into speeding up the encoding algorithms and into minimizing the achievable bit rates. Other standard audio codecs are integrated in the Wave Audio Unit to simplify comparisons of audio quality.

3 Future work

A very interesting task is the integration of psychoacustical techniques like frequency and time masking into our audio codecs. With these methods the auditory model of the human ear can be exploited for further compression. There is a limit to the sensitivity of the ear and if sounds are too weak they will not be detected. This is known as the threshold of audibility. This threshold varies with frequency and it can be increased at any given frequency by the presence of a large signal at a nearby lower frequency. This phenomenon is called masking and it is already widely used in speech coding. Sinha and Tewfik have used such techniques for transparent compression of high-quality audio [40]. They employed the wave packet transform to split the audio signal in the necessary *critical frequency bands* [3].
Furthermore the piggyback protected transmission as proposed in the papers of Bolot should be implemented. Subsequently the improvement in audio quality using our encoding schemes instead of standard codecs can be evaluated.
Enhancing the Wave Audio Unit for a broadcasting option, so that we can apply the PET protection towards our layered audio representation, is an urgent task. The PET code definitely needs to be cleaned up and we think about optimizing it for our requirements. Right now the overhead in both - computation time and additional coding information - is too high. We will need a scenario that simulates the packet loss behaviour of the Internet so that applications like ‘Internet radio‘ using our protection schemes can be tested. It will be interesting to compare the achieved results to applications that are already available. ‘RealAudio™‘ for example is a commercial product that already allows radio like audio connections over the Internet at bit rates of 14.4 and 28.8 kbps. However the quality - especially for music transmissions - is still at best mediocre. We think that with our approach much better audio quality is possible.

A

Appendix
A short tutorial on linear algebra

This chapter is not meant to have the reader going through all the following definitions. It is thought to serve as a reference for the used nomenclature. Simply reading through the given examples may be an adequate way to refresh the necessary mathematical knowledge.

A.1 Vector space

A collection $\{\bar{v}_1, \bar{v}_2, \bar{v}_3, \ldots\}$ of elements is a vector space V over the real numbers R if:

1. for all $a, b \in R$ and for all $\bar{u}, \bar{v} \in V$ holds $a\bar{u} + b\bar{v} \in V$. This provides closure under linear combinations.
2. there exists an unique element $\bar{0} \in V$ such that for all $\bar{u} \in V$ holds $\bar{0}\bar{u} = \bar{0}$ and for all $\bar{u} \in V$ holds $\bar{0} + \bar{u} = \bar{u}$.

The elements of V are called vectors, and the element $\bar{0}$ of V is called the zero vector.

> *Example:* The vector spaces R^2 and R^3 and their geometrical meaning should be familiar to the reader. The elements of the more general vector space R^n are sequences of n real numbers $(u_1, u_2, \ldots, u_n)$.
> Another vector space is the space of all square integrable functions $L_2(R)$, where the vectors are functions $f(x)$ that satisfy $\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty$.

A.2 Inner product and orthogonality

An inner product $\langle \ldots | \ldots \rangle$ on a vector space V is any map from $V \times V$ to R that is:

1. symmetric, meaning for all $\bar{u}, \bar{v} \in V$ holds $\langle \bar{v} | \bar{u} \rangle = \langle \bar{u} | \bar{v} \rangle$
2. bilinear, meaning for all $a, b \in R$ and all $\bar{u}, \bar{v}, \bar{w} \in V$ holds $\langle a\bar{u} + b\bar{v} | \bar{w} \rangle = a\langle \bar{u} | \bar{w} \rangle + b\langle \bar{v} | \bar{w} \rangle$
3. positive definite, meaning for all $\bar{u} \neq \bar{0} \in V$ holds $\langle \bar{u} | \bar{u} \rangle > 0$

The most important use of the inner product is to formalize the idea of orthogonality. Two vectors $\bar{u}, \bar{v}$ of a vector space V with the inner product $\langle \ldots | \ldots \rangle$ are said to be orthogonal if $\langle \bar{u} | \bar{v} \rangle = 0$.

> *Example:* Dealing with vectors $\bar{v} = (v_1, v_2, \ldots, v_n)$ and $\bar{u} = (u_1, u_2, \ldots, u_n)$ of the vector space R^n the common used inner product is $\langle \bar{u} | \bar{v} \rangle = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$.

The inner product of two vectors $\bar{u} = f(x)$ and $\bar{v} = g(x)$ of the vector space $L_2(R)$ is defined as $\langle \bar{u} | \bar{v} \rangle = \int_{-\infty}^{\infty} f(x)g(x)dx$.

A.3 Norms and normalization

A norm $||\ldots||$ on a vector space V is any map from V to R with:

1. for all $\bar{u} \neq \bar{0} \in V$ holds $||\bar{u}|| > 0$
2. for all $a \in R$ and for all $\bar{u} \in V$ holds $||a\bar{u}|| = a||\bar{u}||$
3. for the null vector $\bar{0} \in V$ holds $||\bar{0}|| = 0$

The most important use of the norm is to formalize the idea of the length of a vector. A vector $\bar{u}$ with $||\bar{u}|| = 1$ is said to be normalized.

Example: For the vector space R^n and the vector space $L_2(R)$ the most frequently used norm is the L_2 norm. The definition of the L_2 norm uses the inner product of a vector space resulting in $||\bar{u}||_2 = \sqrt{\langle \bar{u} | \bar{u} \rangle}$.

A.4 Linear independence

A collection $\{\bar{b}_1, \bar{b}_2, \bar{b}_3, \ldots\}$ of vectors from a vector space V is said to be linear independent if:

$$(c_1\bar{b}_1 + c_2\bar{b}_2 + c_3\bar{b}_3 + \ldots = 0) \Leftrightarrow (c_1 = c_2 = c_3 = \ldots = 0)$$

with $c_1, c_2, c_3, \ldots \in R$.
It is easy to show that orthogonal vectors must be linear independent, suggesting that orthogonality is a strong form of linear independence.

A.5 Basis and dimension

A collection $\{\bar{b}_1, \bar{b}_2, \bar{b}_3, \ldots\}$ of linear independent vectors from a vector space V forms a basis B for the vector space V, if every vector $\bar{u} \in V$ can be written as $\bar{u} = c_1\bar{b}_1 + c_2\bar{b}_2 + c_3\bar{b}_3 + \ldots$ with $c_1, c_2, c_3, \ldots \in R$. The scalars $c_1, c_2, c_3, \ldots$ are the coordinates and the vector $(\bar{u})_B = (c_1, c_2, c_3, \ldots)$ is the corresponding coordinate vector for the vector $\bar{u}$ in the vector space V in respect to the basis B. It should be noted that coordinate vectors depend not only on the basis B but also on the order in which the basis vectors are written. A change in the order of the basis vectors result in a corresponding change of order for the entries in the coordinate vectors. When talking about a coordinate vector $(\bar{u})_B$ of a vector space V, one must be aware of the regarding basis B.

A basis B is an orthogonal basis if all basis vectors are mutually orthogonal. It is orthonormal if in addition all basis vectors are normalized.
If a basis B for a vector space V has a finite number of basis vectors $\bar{b}_1, \bar{b}_2, \bar{b}_3, \ldots, \bar{b}_n$, then V is finite-dimensional and its dimension is n. Otherwise the vector space V is said to be infinite-dimensional.

Example: For the vector space R^n the collection $S = \{\bar{e}_1, \bar{e}_2, \bar{e}_3, \ldots, \bar{e}_n\}$ with $\bar{e}_1 = (1, 0, 0, \ldots, 0)$, $\bar{e}_2 = (0, 1, 0, \ldots, 0)$, ... , $\bar{e}_n = (0, 0, 0, \ldots, 1)$ forms a orthonormal basis. The basis S is called the standard basis S of R^n. The coordinate vec-

tor of any vector $u \in R$ with $u = (u_1, u_2, u_3, \ldots, u_n)$ relative to the standard basis S is $(\bar{u})_S = (u_1, u_2, u_3, \ldots, u_n)$, because every vector $\bar{u} \in R^n$ can be written as $\bar{u} = u_1\bar{e}_1 + u_2\bar{e}_2 + \ldots + u_n\bar{e}_n$. Regarding to the standard basis S every vector $\bar{u} \in R^n$ and its coordinate vector $(\bar{u})_S$ are the same, resulting in $\bar{u} = (\bar{u})_S$. It should be emphasized that in general a vector and its coordinate vector are not the same. The equality is a special situation that occurs only with the standard basis S for R^n. Since the vector space R^n has a finite number of basis vectors, it is finite dimensional.

A.6 Vector subspace

A subset U of a vector space V is said to be a subspace of V, written $U \subseteq V$, if U itself is a vector space. Given a vector space V and a subspace U, there always exists a vector subspace W of V such that:

1. every element $\bar{v}$ of V can be written as a linear combination $\bar{v} = \bar{u} + \bar{w}$ of an element $\bar{u}$ in U and element $\bar{w}$ in W.
2. the representation in 1 is unique $\Leftrightarrow U \cap W = \{0\}$. This is called the direct sum decomposition of V and is written as $V = U + W$.
3. one can choose W so that $U \perp W$. In this case every element $\bar{u}$ in U is orthogonal to every element $\bar{w}$ in W. This is called the orthogonal decomposition of V and is written as $V = U \oplus W$.

A.7 Basis transformation

Recall that if $B = \{\bar{b}_1, \bar{b}_2, \ldots\}$ is a basis for a vector space V, then every vector $\bar{u} \in V$ can be expressed uniquely as a linear combination $\bar{u} = c_1\bar{b}_1 + c_2\bar{b}_2 + \ldots$ of the basis vectors. The scalars $c_1, c_2, \ldots$ are the coordinates of $\bar{u}$ relative to B and the vector $(\bar{u})_B = (c_1, c_2, \ldots)$ is the coordinate vector of $\bar{u}$ relative to B. We define

$$[\bar{u}]_B = \begin{bmatrix} c_1 \\ c_2 \\ \ldots \end{bmatrix}$$

to be the coordinate matrix of $\bar{u}$ relative to B.

If we change the basis for a vector space V from some old basis $B = \{\bar{b}_1, \bar{b}_2, \ldots\}$ to some new basis $B' = \{\bar{b}_1', \bar{b}_2', \ldots\}$, then the old coordinate matrix $[\bar{u}]_B$ of a vector $\bar{u}$ is related to the new coordinate matrix $[\bar{u}]_{B'}$ by the equation

$$[\bar{u}]_B = P[\bar{u}]_{B'}$$

where the columns of P are the coordinate matrices of the new basis vectors relative to the old basis. The matrix P is called the transition matrix from the basis B' to the basis B and can be expressed in terms of its column vectors as

$$P = \begin{bmatrix} [\bar{b}_1']_B & [\bar{b}_2']_B & [\bar{b}_3']_B & \ldots \end{bmatrix}$$

If the matrix P is the transition matrix from a basis B' to a basis B then P is invertible and its inverse P^{-1} is the transition matrix from the basis B to the basis B'. This relation is expressed by the analog equation.

$$[\bar{u}]_{B'} = P^{-1}[\bar{u}]_B$$

A change from an orthonormal basis B to another orthonormal basis B' results into a transformation matrix P that has a nice property making its inverse P^{-1} easy to find. The inverse P^{-1} of this matrix is its simply its transpose P^T resulting in $P^{-1} = P^T$.

R

References

[1] A. Albanese, J. Bloemer, J. Edmonds and M. Luby, '*Priority encoding transmission*', International Computer Science Institute, August 1994

[2] E. Biersack, '*Performance evaluation of FEC in ATM networks*', Proceedings ACM Sigcomm 92 in Baltimore, pp. 248-257, August 1992

[3] J. Benedetto and A. Teolis, '*A wavelet auditory model and data compression*', Applied and Computational Harmonic Analysis, vol. 1, December 1993

[4] J. Bolot, '*Characterizing the end-to-end behaviour of the Internet: measurements, analysis and applications*', ITC specialists seminar on traffic, modeling and measurement, 1995

[5] J. Bolot, H. Crepin and A. Vega García, '*Analysis of audio packet loss in the Internet*', INRIA, 1993

[6] J. Bolot and A. Vega García, '*The case for FEC based error control for packet audio in the Internet*', INRIA, 1996

[7] J. Bolot and A. Vega García, '*Control mechanisms for packet audio in the Internet*', Proceedings IEEE Infocom 96 in San Francisco, pp. 232-239, April 1996

[8] H. Chodura and R. Storn, '*Audio communication for distributed collaborative systems*', International Computer Science Institute, January 1996

[9] R. Chua, '*The ICSI audio laboratory*', International Computer Science Institute, May 1996

[10] M. Cody, '*The fast wavelet transform*', Dr. Dobb's Journal, April 1992

[11] M. Cody, '*The wavelet packet transform*', Dr. Dobb's Journal, April 1994

[12] R. Coifman and V. Wickerhauser, '*Entropy-based algorithms for best basis selection*', Yale University, New-Haven, Connecticut

[13] R. Coifman and V. Wickerhauser, '*Best-adapted wave packet basis*', Yale University, New-Haven, Connecticut

[14] R. Coifman, Y. Meyer and V. Wickerhauser, '*Wavelet analysis and signal processing*', Yale University, New-Haven, Connecticut

[15] R. Coifman, Y. Meyer, S. Quake and V. Wickerhauser, '*Signal processing and compression with wave packets*', Yale University, New-Haven, Connecticut

[16] ‘*Free Phone -- Telephony over the Internet*’
http://www.inria.fr/rodeo/fphone/

[17] A. Graps, ‘*An introduction to wavelets*’, IEEE Computational Science and Engineering, Los Almanitos, Summer 1995

[18] V. Hardman, M. Sasse and A. Watson, ‘*Successful voice reconstruction for packet networks using redundancy*’, University College London, April 1995

[19] V. Hardman and M. Handley, ‘*Reliable audio for use over the Internet*’, University College London, 1995

[20] F. Halsall, ‘*Data communications, computer networks and open systems*’, Addison-Wesley Publishing Company, 1995

[21] N. Hess-Nielsen and M. Wickerhauser, ‘*Wavelets and time-frequency analysis*’, Proceedings IEEE Infocom 96 in San Francisco, pp. 523-540, April 1996

[22] C. Hicks, ‘*Aspects of sampling, oversampling, quantization, dither and noise-shaping, as applied to digital audio*’, November 1994

[23] M. Isenburg, ‘*Comm++ 3.1 - An object orientated communication platform for distributed systems*’, Fraunhofer-IGD, Internal Report, August 1995

[24] IVS, ‘*INRIA Videoconferencing System*’,
http://www.inria.fr/rodeo/ivs.html

[25] B. Jawerth and W. Sweldens, ‘*An overview over wavelet based multiresolution analysis*’, Department of Mathematics, University of South Carolina

[26] B. Lamparter, A. Albanese, M. Kalfane and M. Luby, ‘*PET - priority encoded transmission: A new, robust and efficient video broadcast technology*’, International Computer Science Institute, August 1995

[27] C. Leicher, ‘*Hierarchical encoding of MPEG sequences using priority encoded transmission*’, International Computer Science Institute, November 1994

[28] M. Mathog and E. Knorr, ‘*Reviews - 12 web phones to make net calls*’,
http://www.cnet.com/Content/Reviews/Compare/Wphone/index.html

[29] MERCI, ‘*Multimedia European Research Conferencing Integration*’,
http://www.cs.ucl.ac.uk/mice/merci/index.html

[30] MICE, ‘*Multimedia Integrated Conferencing for Europe*’,
http://www.cs.ucl.ac.uk/mice/mice_home.html

[31] ‘*Numerical recipes in C: The art of scientific computing*’, Cambridge University Press, 1992

[32] V. Paxton, ‘*An analysis of end-to-end Internet dynamics*’, Ph.D. Dissertation Seminar, University of California, Berkeley, Mai 1996

[33] PET, ‘*Priority Encoded Transmission*’,
http://www.icsi.berkeley.edu/PET/icsi-pet.html

[34] K. Ramchandran, M. Vetterli and C. Herley, ‘*Wavelets, subband coding, and best bases*’, Proceedings of the IEEE Infocom 96 in San Francisco, pp. 541-560, April 1996

[35] R. Ramjee, J. Kurose and D. Towsley, '*Adaptive play-out mechanisms for packetized audio applications in wide-area networks*', Proceedings IEEE Infocom 94 in Toronto, pp.680-688, June 1994

[36] RAT, '*Robust Audio Tool*', http://www.cs.ucl.ac.uk/mice/rat/index.html

[37] ReLaTe, '*Remote Language Teaching over SuperJANET*', http://www.ex.ac.uk/pallas/relate/welcome.html

[38] T. Robinson, '*Speech analysis - a post graduate course in computer speech and language processing*', Cambridge University, 1996

[39] N. Shacham and P. McKenney, '*Packet recovery in high-speed networks using coding and buffer management*', Proceedings IEEE Infocom 90 in San Francisco, pp.124-131, May 1990

[40] D. Sinha and A. Tewfik, '*Low bit rate transparent audio compression using adapted wavelets*', IEEE Transactions on signal processing, pp.3463-3479, December 1993

[41] E. Stollnitz, T. DeRose and D. Salesin, '*Wavelets for computer graphics: a primer*', University of Washington, Seattle

[42] R. Storn, SIEMENS, International Computer Science Institute, '*personal conversation*'

[43] M. Ueda and S. Lodha, '*Wavelets: An elementary introduction and examples*', University of California, Santa Cruz, January 1995

[44] VIC, '*Video Conferencing Tool*', http://www-nrg.ee.lbl.gov/vic/

[45] B. Vidakovíc and P. Müller, '*Wavelets for kids - a tutorial introduction*', Duke University, 1991

[46] R. Warren, '*Auditory perception*', Pergamon Press

[47] V. Wickerhauser, '*Acoustic signal compression with wave packets*', Yale University, New-Haven, Connecticut

[48] V. Wickerhauser, '*Adapted wavelet analysis from theory to software*', AK Peters, Boston, 1994

[49] R. Zelinski and P. Noll, '*Adaptive transform coding of speech signals*', Heinrich-Hertz-Institut Berlin, October 1976