Rate-Distortion Optimal Vector Selection in Frame Based Compression

by

Tom Ryen

Submitted in partial fulfillment
of the requirements for the degree of
DOKTOR INGENIØR

University of
Stavanger

Norway

2005

"Now I know in part; then I shall know fully,
even as I am fully known."

(1 Corinthians 13, 12b)

Abstract

In signal compression we distinguish between lossless and lossy compression. In lossless compression, the encoded signal is more bit efficient than the original signal and is exactly the same as the original one when decoded. In lossy compression, the encoded signal represents an approximation of the original signal, but it has less number of bits. In the latter situation, the major issue is to find the best possible rate-distortion (RD) tradeoff. The rate-distortion function (RDF) represents the theoretical lower bound of the distortion between the original and the reconstructed signal, subject to a given total bit rate for the compressed signal. This is with respect to any compression scheme. If the compression scheme is given, we can find its operational RDF (ORDF).

The main contribution of this dissertation is the presentation of a method that finds the operational rate-distortion optimal solution for an overcomplete signal decomposition. The idea of using overcomplete dictionaries, or frames, is to get a sparse representation of the signal. Traditionally, suboptimal algorithms, such as Matching Pursuit (MP), are used for this purpose. Given the frame and the Variable Length Codeword (VLC) table embedded in the entropy coder, the solution of the problem of establishing the best RD tradeoff has a very high complexity. The proposed method reduces this complexity significantly by structuring the solution approach such that the dependent quantizer allocation problem reduces into an independent one. In addition, the use of a solution tree further reduces the complexity. It is important to note that this large reduction in complexity is achieved without sacrificing optimality. The optimal rate-distortion solution depends on the frame selection and the VLC table embedded in the entropy coder. Thus, frame design and VLC optimization is part of this work.

Extensive coding experiments are presented, where Gaussian AR(1) processes and various electrocardiogram (ECG) signals are used as input signals. The experiments demonstrate that the new approach outperforms Rate-Distortion Optimized (RDO) Matching Pursuit, previously proposed in [17], in the rate-distortion sense.

Preface

This dissertation is submitted in partial fulfilment of the requirements for the degree of *doktor ingeniør* at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. Professor Sven Ole Aase and professor John Håkon Husøy at University of Stavanger (UiS), Norway, and associate professor Dag Haugland at University of Bergen, Norway, have been my supervisors.

The work has been carried out at the Department of Electrical and Computer Engineering of UiS from August 1999 to March 2005. Included in the work are compulsory courses corresponding to one year full time studies. Five months were spent as a visiting scholar at the Image and Video Processing Lab, Department of Electrical and Computer Engineering, Northwestern University, Evanston, USA. Three weeks were spent at HSR Hochschule für Technik Rapperswil, Rapperswil, Switzerland. The last three years have included graduate and undergraduate lecture duties at UiS. The work has been funded by UiS. The abroad stays were funded by a scholarship from the Norwegian Research Council.

Parts of the work leading to this dissertation have been published in [47, 46, 45, 44].

Acknowledgments

First of all, I would like to thank my three supervisors Professor Sven Ole Aase, Professor John Håkon Husøy and Associate Professor Dag Haugland. Thank you, Prof. Aase, for giving me insight into linear algebra, particularly, and for arranging my work situation, generally. Thank you, Prof. Husøy, for your encouraging guidance and for giving me advices through your knowledge in signal processing. Thank you, Dr. Haugland for sharing your enthusiasm and your insight into optimization theory with me.

I am very grateful to all my colleagues at the Department of Electrical and Computer Engineering at University of Stavanger for contributing to a highly appreciated job atmosphere. Particularly I would like to thank Dr. Karl Skretting for valuable help and useful discussions.

Some of the most important moments of bringing results to this work has taken place at the Northwestern University in Evanston, USA. Thanks to Professor Aggelos K. Katsaggelos for giving me a pleasant stay and valuable advices. Thanks to Professor Guido M. Schuster for creative suggestions and inspiring guidance, and for inviting me to Switzerland.

Thanks to my parents Gudrun and John for giving me the best start and for supporting me all my life. Thanks to my daughter Hanna for bringing joy and hope. Last but not least I would like to thank my wife Turid for love, patience and support in all ups and downs connected to this work, and in my life in general.

Contents

Nomenclature

$\|\cdot\|$	l_2-norm
$\langle a, b\rangle$	the inner product of a and b
$\mathbf{a}$	vector of constants symmetric distributed around zero
$\alpha_{l,M}$	for signal block l, the product of λ and the sum of bit rates of M *indices* and EOB symbol
B	number of nodes to be selected at each solution tree level in a partial search algorithm
B_m	number of nodes to be selected at solution tree level m in a partial search algorithm
$\beta_{l,M}$	for signal block l, sum of M minimum rate-distortion solution of added quantized coefficient values
C	total cost, which is $D + \lambda R$
C_l	cost of signal block l, which is $D_l + \lambda R_l$
C_l^*	so far minimum cost of signal block l
$C_{l,m}$	cost of signal block l at iteration m in the RDO MP algorithm
$C_{l,M}$	for signal block l, the cost at a solution tree node at level M
C^i	total cost at training procedure iteration no. i
C^+	total cost when $\mathbf{F}^+$ is frame
D	total distortion
D_l	distortion of signal block l
D_{vs}	total distortion introduced by vector selection
$D_{vs,l}$	for signal block l, distortion introduced by vector selection
$D_{RDF}(R)$	distortion of the rate-distortion function as a function of rate R
D^+	total distortion when $\mathbf{F}^+$ is frame
δ	small constant used to update the frame elements in the convergent frame design loop

δ_m	rate at iteration m of RDO BMP or RDO ORMP algorithms
Δ	quantization step size
ΔC	difference in total cost from one training procedure iteration to the next
$\mathbf{e}_{l,M}$	difference between the original signal vector l and reconstructed signal vector l of unquantized coefficients, when M frame vectors are selected
$\mathbf{E}$	difference between the original signal, $\mathbf{X}$, and the reconstructed preprocessor signal, $\breve{\mathbf{X}}$
$\tilde{\mathbf{E}}$	reconstruction of $\mathbf{E}$
ε	small constant used as a stop constraint in an iterative algorithm
f_{nk}	element n in column k in the frame matrix $\mathbf{F}$
$\mathbf{f}_k$	column k in the frame matrix $\mathbf{F}$
$\mathbf{f}_k^+$	column k in the frame matrix $\mathbf{F}^+$
$\mathbf{f}_{k_m}$	selected frame vector in iteration m
$\hat{\mathbf{f}}_{k_m}$	orthogonalization of selected frame vector in iteration m with respect to previously selected frame vectors
$\mathbf{F}$	frame matrix with dimensions $N \times K$
$\mathbf{F}'$	in coarse frame design, the frame matrix where a not chosen column vector is removed from the matrix
$\mathbf{F}^+$	in fine frame design, the frame matrix where a small real valued constant, δ, is added to one of the frame elements
$\mathbf{F}^*$	frame after the last training iteration
$\mathbf{F}^i$	frame after training iteration no. i
$g(n)$	low-pass filter impulse response
$h(n)$	anti-aliasing filter impulse response
$i_{l,m}$	index of the m^{th} selected frame vector for signal block l
i_{new}	lowest index of the so far not selected frame vectors
$\mathbf{i}_l$	vector containing the indices of the selected frame vectors for signal block l
$(\mathbf{i}_l)^*$	vector containing the indices of the final selected frame vectors for signal block l
$(\mathbf{I})^*$	matrix where the column l is $(\mathbf{i}_l)^*$
indices	table containing frame indices and their corresponding binary codewords
j	*values* codeword table index

j_m	index of the element of the *values* table to be the coefficient of the m^{th} selected vector
J	number of *values* codewords
k	frame vector index
K	number of frame vectors in $\mathbf{F}$
κ	number of quantization steps in the DPCM encoder of the pre-processor
l	signal block index
L	number of signal blocks
LB	lower bound of cost when the set of selected vectors is known
λ	the Lagrangian multiplier
$\boldsymbol{\lambda}$	vector of different Lagrangian multipliers
m	frame vector selection index
M	number of selected frame vectors
M_{max}	upper bound on the number of selected frame vectors per signal block
μ_l	mean value of the samples in signal block l
$\breve{\mu}_l$	reconstructed mean value of the samples in signal block l
$\boldsymbol{\mu}$	vector of mean values of all signal blocks
$\breve{\boldsymbol{\mu}}$	vector of reconstructed mean values of all signal blocks
n	signal sample index
N	number of samples per signal block
ξ	index used as needed
P	down- and up-sampling factor
$\mathbf{q}_{l,m}$	column vector m of matrix $\mathbf{Q}_l$
$Q[\cdot]$	scalar quantizer
$\mathbf{Q}_l$	matrix with orthonormal column vectors, QR-decomposition of $\boldsymbol{\Phi}_l$
$\mathbf{Q}_{l,M}$	matrix with M orthonormal column vectors
$\mathbf{Q}_l^+$	matrix with orthonormal column vectors, QR-decomposition of a subset of $\mathbf{F}^+$
$\mathbf{r}_m$	residual vector at iteration m
R	total bit rate
R_{budget}	total bit budget
R_l	bit rate of signal block l
R^+	total bit rate when $\mathbf{F}^+$ is frame

R^{EOB}	number of bits for EOB symbol
$R^{ind}_{l,m}$	number of bits for the *index* codeword of m^{th} coefficient of signal block l
R^{ind}_{min}	minimum *index* codeword bit length
$R^{run}_{l,m}$	number of bits for the *run* codeword of m^{th} coefficient of signal block l
R^{run}_{min}	minimum *run* codeword bit length
$R^{val}_{l,m}$	number of bits for the *value* codeword of m^{th} coefficient of signal block l
R^{val}_{min}	minimum *value* codeword bit length
$\mathbf{R}_l$	the upper triangular matrix in the QR-decomposition of $\mathbf{\Phi}_l$
$\mathbb{R}^+$	space of all real and non-negative numbers
$\mathbb{R}^N$	N-dimensional real Hilbert space
$runs$	table containing run symbols and their corresponding binary codewords
$runs^*$	*runs* codeword table after the last training iteration
$runs^i$	*runs* codeword table after training iteration no. i
ρ	adjacent-sample correlation coefficient in an AR(1) process
S	number of nonzero coefficients in the representation of the input signal
S_{budget}	upper bound on the total number of nonzero coefficients
$S_{budget,l}$	upper bound on the number of nonzero coefficients of signal block l
S_l	number of nonzero coefficients of signal block l
$S_{xx}(e^{j\omega})$	power density spectrum of the signal $\mathbf{x}$
SNR_{RDF}	signal-to-noise ratio for the rate-distortion function
σ_x^2	the variance of the signal $\mathbf{x}$
$\mathbf{T}$	transform matrix
$\mathbf{T}^T$	transpose of $\mathbf{T}$
$\mathbf{T^{-1}}$	inverse of $\mathbf{T}$
$\mathbf{\Theta}_l$	binary (0,1)-matrix with dimensions $M \times J$
$\mathbf{x}$	input signal vector of length NL
$\mathbf{x}_l$	input signal vector no. l, of length N
$\mathbf{x}_s$	input signal vector of length N, with corresponding coefficient vector with the highest number of nonzero coefficients

$\hat{\mathbf{x}}_l$	reconstruction of $\mathbf{x}_l$ using not quantized coefficients
$\tilde{\mathbf{x}}_l$	reconstruction of $\mathbf{x}_l$ using quantized coefficients
X	source signal
$\tilde{X}$	reconstructed signal
$\mathbf{X}$	input signal matrix with dimension $N \times L$, where column l is signal vector $\mathbf{x}_l$
$\tilde{\mathbf{X}}$	reconstructed signal matrix
$\breve{\mathbf{X}}$	reconstructed preprocessor signal matrix
$\mathbf{X}_{test}$	test signal matrix
$\mathbf{X}_{tr}$	training signal matrix
Y	source coding output signal
$\tilde{Y}$	channel coding output signal
Y^{opt}	optimal source coding output signal
$v_{l,m}$	unquantized coefficient corresponding to the m^{th} selected frame vector for signal block l
$\tilde{v}_{l,m}$	quantized coefficient corresponding to the m^{th} selected frame vector for signal block l
$v^o_{l,m}$	unquantized coefficient corresponding to the orthogonalization of the m^{th} selected frame vector for signal block l
$\tilde{v}^o_{l,m}$	quantized coefficient corresponding to the orthogonalization of the m^{th} selected frame vector for signal block l
$\mathbf{v}_l$	vector of unquantized coefficients corresponding to the selected frame vectors for signal block l
$\mathbf{v}^o_l$	vector of unquantized coefficients corresponding to the orthogonalization of the selected frame vectors for signal block l
$\tilde{\mathbf{v}}_l$	vector of quantized coefficients corresponding to the selected frame vectors for signal block l
$\tilde{\mathbf{v}}^o_l$	vector of quantized coefficients corresponding to the orthogonalization of the selected frame vectors for signal block l
$(\tilde{\mathbf{v}}^o_l)^*$	optimal vector of quantized coefficients corresponding to the orthogonalization of the selected frame vectors for signal block l
$(\tilde{\mathbf{V}}^o)^*$	matrix of L column vectors, where column no. l is $(\tilde{\mathbf{v}}^o_l)^*$
val	vector containing J coefficient representation values
values	table containing coefficient representation values and their corresponding binary codewords

$values^*$	*values* codeword table after the last training iteration
$values^i$	*values* codeword table after training iteration no. i
ϕ	parametric variable
$\boldsymbol{\phi}_{l,M}$	for signal block l, the M^{th} selected frame vector
$\boldsymbol{\Phi}_l$	matrix where the columns are the selected frame vectors for signal block l
$\boldsymbol{\Phi}_{l,M}$	matrix where the columns are M selected frame vectors for signal block l
$\tilde{w}_{l,k}$	the k^{th} element in $\tilde{\mathbf{w}}_l$
$\tilde{w}^o_{l,k}$	the k^{th} element in $\tilde{\mathbf{w}}^o_l$
$\mathbf{w}_l$	unquantized coefficient vector for signal block l
$\tilde{\mathbf{w}}_l$	quantized coefficient vector for signal block l
$\tilde{\mathbf{w}}^o_l$	unquantized coefficient vector for signal block l, where the nonzero elements correspond to the orthogonalization of the selected frame vectors
$\tilde{\mathbf{w}}_s$	coefficient vector corresponding to $\mathbf{x}_s$
$\mathbf{W}$	matrix with dimensions $K \times L$, where the columns are unquantized coefficient vectors
$\tilde{\mathbf{W}}$	matrix with dimensions $K \times L$, where the columns are quantized coefficient vectors
$\tilde{\mathbf{W}}'$	in coarse frame design, the matrix with dimensions $(K-1) \times L$, where row k in $\tilde{\mathbf{W}}$ is removed
z	objective function in quantization step size optimization
$\emptyset$	empty set

List of Abbreviations

AR(1)	Autoregressive process, first order Markov
BMP	Basic Matching Pursuit, a greedy vector selection algorithm
dB	decibel
DCT	Discrete Cosine Transform
DFS	Depth-First-Search
DPCM	Differential Pulse Code Modulation
ECG	Electrocardiogram
EOB	End Of Block
FOCUSS	FOCal Underdetermined System Solver
FOLS	Fast Orthogonal Least Squares
FOMP	Fast Orthogonal Matching Pursuit, a greedy vector selection algorithm
GHz	Giga Hertz
GMP	Global Matching Pursuit, a greedy vector selection algorithm
GS	Gram-Schmidt, an orthogonalization algorithm
IDCT	Inverse Discrete Cosine Transform
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loeve Transform
LB	Lower bound
MIT	Massachusetts Institute of Technology
MIT100	Signal 100 from the MIT arrhythmia database of ECG signals
MIT103	Signal 103 from the MIT arrhythmia database of ECG signals
MIT113	Signal 113 from the MIT arrhythmia database of ECG signals
MIT207	Signal 207 from the MIT arrhythmia database of ECG signals
MIT217	Signal 217 from the MIT arrhythmia database of ECG signals
MMP	Modified Matching Pursuit, a greedy vector selection algorithm
MOD	Method of Optimal Directions, a frame design algorithm
MP	Matching Pursuit, a group of greedy vector selection algorithms
MP3	Moving Picture Experts Group Audio Layer 3
MPEG	Moving Picture Experts Group

OLS	Orthogonal Least Squares
OMP	Orthogonal Matching Pursuit, a greedy vector selection algorithm
ORD	Operational Rate-Distortion
ORDE	Operational Rate-Distortion Encoding/ Encoder
ORDF	Operational Rate-Distortion Function
ORMP	Order Recursive Matching Pursuit, a greedy vector selection algorithm
QMP	Quantized Matching Pursuit, a greedy vector selection algorithm
RD	Rate-Distortion
RDF	Rate-Distortion Function
RDO	Rate-Distortion Optimal/ Rate-Distortion Optimized
RDO MP	Rate-Distortion Optimized Basic Matching Pursuit
RDO BMP	Rate-Distortion Optimized Matching Pursuit
RDO ORMP	Rate-Distortion Optimized Order Recursive Matching Pursuit
RDT	Rate-Distortion Theory
RLC	Run-Length Coding
SNR	Signal to Noise Ratio
SSQ	Simultaneous Selection and Quantization
VLC	Variable Length Codeword

Chapter 1

Introduction

Signal compression is an important subject in digital communication and data storage systems today. By signals, we mean all kind of digitalized data. The signals can be one-dimensional such as digitalized waveforms of speech and music, two-dimensional like digital images and contours, or multi-dimensional like video and seismic data. The purpose of signal compression is to generate a more bit efficient representation of the presented information. Shannon called it source coding in his fundamental work in information theory [51, 52]. In the illustration in Figure 1.1, the source encoder generates the bit stream Y, which is a more bit efficient representation of the source signal, X, before it is transmitted through the channel. The channel could be the Internet, e.g., and would normally include error control coding, or in other words, channel coding. In this work the focus is on source coding, and we expect the channel to be free of error. In other words, the channel output signal, $\tilde{Y}$, is expected to be identical to the channel input signal, Y. In lossless source coding, the reconstructed signal, $\tilde{X}$, is identical to the original signal, X. The goal is to reduce the number of bits required for the coded data [16], i.e. to remove redundant information. In lossy source coding, the reconstructed signal, $\tilde{X}$, is an approximation to the source signal, X. By allowing lack of perfect reconstruction, the compression potential is much higher than in lossless coding. In order to get a more bit efficient representation, we allow some loss of information, or some *distortion* between X and $\tilde{X}$. If we can accept a high distortion, the bit rate can be small, and the compression gain is large. If not, the compression gain will be smaller. This tradeoff between a lowest possible bit rate and lowest possible distortion is an essential issue in lossy coding. Whether the distortion is at an acceptable level or not, depends on the application. For example, a digital photograph used in a slide show

requires a much higher quality than the same photograph as a thumbnail image at a web site. In the latter case most of the detail information is irrelevant, due to the fact that the human eye is not capable of perceiving details in a small image, and the image can be represented by a much smaller number of bits.

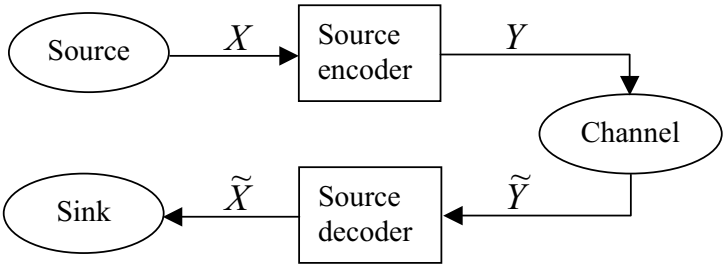


Figure 1.1: Communication system.

The focus of this thesis is on finding the optimal rate-distortion (RD) tradeoff in a frame based coding scheme. Frame based coding is a generalization of transform coding. These coding schemes are explained in Section 1.1. Section 1.2 includes a brief introduction to rate-distortion theory. Section 1.3 summarizes the contribution of this thesis.

1.1 Lossy signal compression

There are many different lossy compression techniques available today, and still considerable research is put into this field. Most of these techniques involve three steps at the encoding side: 1) A transforming step, where blocks of signal samples are mapped into blocks of coefficients, 2) a quantization step, where the coefficients are quantized to discrete values, and 3) an entropy encoding step, where the discrete coefficients are converted to a sequence of bits, by a lossless encoder. The lack of perfect reconstruction is caused by the two first steps, and for some lossy encoders, only by the quantizer. The latter is the situation in *transform coding* [16], which is one of the most popular techniques known today. The famous still image compression standard, JPEG (Joint Photographic Experts Group), is based on transform coding. So is also the video compression standard MPEG (Moving Picture Experts Group), and the audio compression standard, MP3 (MPEG Audio Layer 3).

Our work focuses on *frame based coding* [12]. This is a technique in closely related to transform coding. For this reason, we will have a closer look at both of these techniques.

1.1.1 Transform coding

Transform coding is a popular compression technique. This is due to its simplicity and efficiency [20]. Many variants exist. Irrespective of particulars, basic transform encoders consist of three steps: The first step is the forward *transform*, denoted $\mathbf{T^{-1}}$ for notational convenience. The transform produces coefficients, that in step two are quantized by a scalar quantizer. Each quantized value corresponds to a discrete symbol. In step three these symbols are translated to a string of bits, Y. A schematic view of a transform coding system is shown in Figure 1.2. We expect the channel to be error free, thus $Y = \tilde{Y}$. As the figure shows, the transform decoder consists of three steps, as well. First, an entropy decoder, converting the bit string to symbols. Second, an inverse quantizer, generating discrete valued coefficients of these symbols. And, third, an inverse transform, or a *reconstruction* transform, $\mathbf{T} = (\mathbf{T^{-1}})^{-1}$, making an approximation of the original signal by using the quantized coefficients as input.

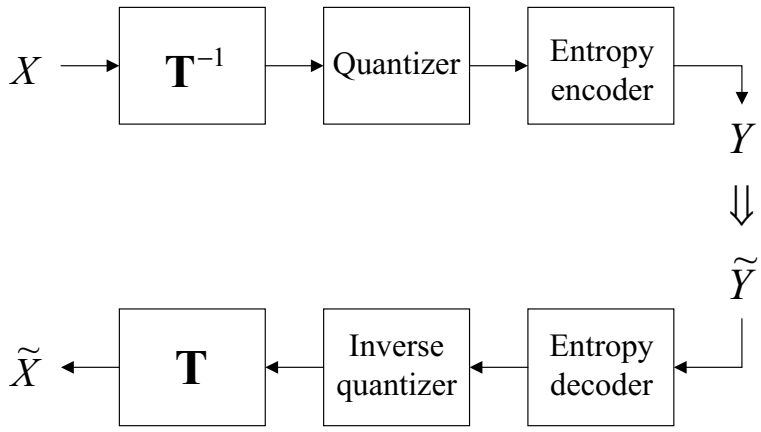


Figure 1.2: Transform coding.

Let a one-dimensional signal, $\mathbf{x}$, represented by an $NL \times 1$ vector, be divided into L blocks, each consisting of N samples. The l^{th} signal block, $\mathbf{x}_l$, is a column vector of length N. The reconstruction transform, $\mathbf{T}$, can be represented

as a matrix of dimensions $N \times N$. The idea in transform coding is to decorrelate the data and to represent the signal energy in few coefficients. This is done by the forward transform, $\mathbf{T^{-1}}$, if its row vectors are chosen properly. Let the coefficient set $\{w_j\},\ j = 1, 2, \ldots, N$, constitute the N-dimensional vector $\mathbf{w}_l$. The *analysis* and *synthesis* equations are

$$\mathbf{w}_l = \mathbf{T}^{-1}\mathbf{x}_l \tag{1.1}$$

and

$$\begin{aligned} \mathbf{x}_l &= \mathbf{T}\mathbf{w}_l & (1.2)\\ \tilde{\mathbf{x}}_l &= \mathbf{T}\tilde{\mathbf{w}}_l, & (1.3) \end{aligned}$$

respectively. $\tilde{\mathbf{x}}_l$ is the reconstructed version of $\mathbf{x}_l$, and $\tilde{\mathbf{w}}_l$ is the quantized version of $\mathbf{w}_l$. If $\mathbf{T}$ is orthogonal, that is, all column vectors of $\mathbf{T}$ are orthonormal and real, the inverse transform is the transposed transform matrix: $\mathbf{T}^{-1} = \mathbf{T}^T$. The transform is energy preserving, which implies that $\|\mathbf{w}_l\|^2 = \|\mathbf{x}_l\|^2$. Since most of the signal energy is collected in a small number of coefficients, most of the coefficients will have a small value, and consequently will be put to zero by the quantizer. This is what we call energy packing. It is easy to show that the squared error between the original and the reconstructed signal vectors is equal to the squared error between the original and the quantized coefficient vectors [16]:

$$\left\|\mathbf{x}_l - \tilde{\mathbf{x}}_l\right\|^2 = \left\|\mathbf{w}_l - \tilde{\mathbf{w}}_l\right\|^2 = \sum_{k=1}^{K}(w_{l,k} - \tilde{w}_{l,k})^2. \tag{1.4}$$

The minimum squared error can be found by minimizing coefficient errors independently. The discrete Karhunen-Loeve Transform (KLT) is an orthogonal transform, consisting of the eigenvectors of the autocorrelation matrix of the input signal. If the input signal is a statistically stationary Gaussian process [27], KLT is optimal in the energy packing sense [48]. Unfortunately, the KLT is signal dependent, thus it has to be computed for each class of input signals. In addition, practical signals are not always stationary, resulting in frequent recalculations of the KLT, in order to keep it optimal. The Discrete Cosine Transform (DCT) [16] is a *fixed* orthogonal transform. DCT achieves performance very close to KLT, and yet it has the advantage of being signal independent. The DCT is easy to implement in both software and hardware.

It is part of the JPEG standard [48], where image blocks of 8×8 pixels are transformed by an 8×8 forward DCT. At low bit rate coding, very few of the coefficients are nonzero when quantized. We say that the representation of the signal is *sparse*. This sparsity is utilized in the entropy encoder, in order to get a bit efficient representation.

1.1.2 Frame based coding

Consider that we have more than N column vectors in the reconstruction part of the transform coding scheme of the previous section. The transform matrix, $\mathbf{T}$, is replaced by a frame[1] matrix [22], $\mathbf{F}$, with dimensions $N \times K$, where $K > N$. The signal expansion [53], represented by the coefficient vector, $\mathbf{w}_l$, will now have K terms.

Given that the set of column vectors, $\{\mathbf{f}_k\}$, $k = 1, \ldots, K$, span the space to which $\mathbf{x}_l$ belongs, $\mathbf{x}_l$ can be written as a combination of these vectors, and the *synthesis* equation is now

$$\mathbf{x}_l = \mathbf{F}\,\mathbf{w}_l \tag{1.5}$$

$$\tilde{\mathbf{x}}_l = \mathbf{F}\tilde{\mathbf{w}}_l = \sum_{k=1}^{K} \tilde{w}_{l,k}\mathbf{f}_k, \tag{1.6}$$

where $\tilde{w}_{l,k}$ is the k^{th} element in $\tilde{\mathbf{w}}_l$. The column vectors constituting the frame are all assumed to be of unit length.

The rationale for replacing the transform by a frame is to get a more bit efficient representation without further loss of information. Having more than N column vectors to choose from increases the flexibility: There is a better chance of finding a sparse representation with a good approximation of the signal. A sparse representation is the same as having just a few nonzero coefficients in the coefficient vector. An effective entropy coding scheme would in this case use the quantized *values* of the nonzero coefficients and their corresponding *indices* as input symbols. According to the symbol probabilities, the number of bits for each of the symbol codewords will vary. These codewords are stored in a *variable length* codeword (VLC) table. A symbol with a high probability will have a binary codeword with few bits, while a symbol with low probability will have a large number of bits. All in all, the total number

[1] A frame is also called an overcomplete dictionary.

of bits in the final binary string will be smaller when using variable length codewords than by using fixed length codewords.

A price we have to pay when introducing an overcomplete dictionary, is the loss of an *analysis* equation in the encoding part of the compression scheme. The analysis equation for transforms, (1.1), is not valid for frames, simply because there can not exist any inverse of the non-quadratic matrix $\mathbf{F}$. In Figure 1.3 there is no $\mathbf{F}^{-1}$, but instead a "frame vector selection" block. The easily solved problem of finding a set of unique coefficients using a transform is replaced by an NP-hard problem [11], when dealing with frames. This is the reason why fast suboptimal vector selection algorithms are used in order to find a sparse $\tilde{\mathbf{w}}_\mathbf{l}$ that generates a good approximation to the original signal, $\mathbf{x}_\mathbf{l}$. Some of the most important vector selection algorithms are discussed in Chapter 2.

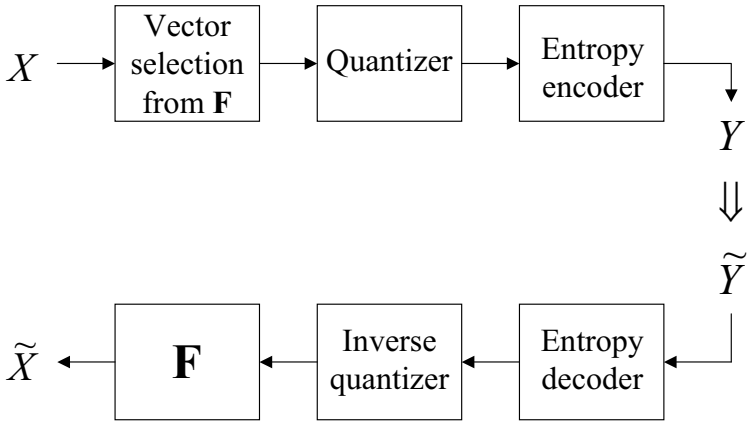


Figure 1.3: Frame based coding.

Another disadvantage with frame based coding, compared to transform coding, is the loss of energy preservation in the signal block coefficients. In fact, the energy in the coefficient vector can be larger than the energy in the signal vector.

The main focus in frame based coding research has been in the first phase, to find a sparse representation that minimize the distortion, and in the next phase, to quantize and to entropy encode this representation. In this work, we focus on finding a sparse representation that minimize the *rate-distortion tradeoff*. Rate-distortion optimization is discussed in the following section.

1.2 Rate-distortion optimization

The major issue in *rate-distortion optimization* is to find the representation of a signal with the fewest number of bits possible for a given reconstruction quality. Rate-Distortion Theory (RDT) [4] has been essential in many applications, such as in Video compression [49, 38, 35], Shape Coding [32] and Compression of electrocardiogram (ECG) data [37]. The central entity in RDT is the Rate-Distortion Function (RDF), which given a signal source, is the lower bound of the distortion that is obtainable with a given bit rate. When the bit lengths of the codewords are known, we can find the *Operational Rate-Distortion Function* (ORDF) [49]. When using frame based coding, all possible ways of quantizing each combination of coefficients will result in a rate R and a distortion D, which can be shown as an (R, D)-point in a rate-distortion diagram. An (R, D)-point is a part of the ORDF if there is no other (R, D)-points with a smaller distortion using the same or a smaller rate. A simple example of a rate-distortion diagram is shown in Figure 1.4. All (R, D)-points are indicated as plus signs. The circled ones are the members of the ORDF, while the solid line represents the ORDF's *convex envelope*. Note that these (R, D)-points are operational in that they are directly achievable with the chosen coding scheme and for the given set of test data. While the bound given by Shannon's theoretical RDF [51, 52] gives no constructive procedure for attaining that optimal performance, in the operational RD case we always deal with achievable points.

A much used method to find members of the operational (R, D)-points' convex envelope is the *Lagrangian Multiplier Method* [49, 38]. This method is essential in the way we formulate and solve our problem in Chapters 3 and 4. The basic idea of this technique is as follows: We introduce a Lagrangian multiplier, λ, which is a non-negative real number. The Lagrangian cost is $C(\lambda) = D + \lambda R$, where D and R are the distortion and the rate for a particular set of quantized coefficients, respectively. Minimizing $C(\lambda)$ for a given value of λ, results in finding one of the (R, D)-points on the convex envelope. For a set of different λ-values, where $\lambda \geq 0$, we find a set of (R, D)-points constituting the convex envelope when minimizing $C(\lambda)$. When $\lambda = 0$, minimization of the Lagrangian cost is equivalent to minimizing the distortion only, i.e., it selects the point closest to the y-axis in Figure 1.4. Conversely, minimizing $C(\lambda)$ when λ becomes arbitrary large is equivalent to minimizing the rate only, and thus we find the point closest to the x-axis. Intermediate values of λ determine intermediate points on the convex envelope.

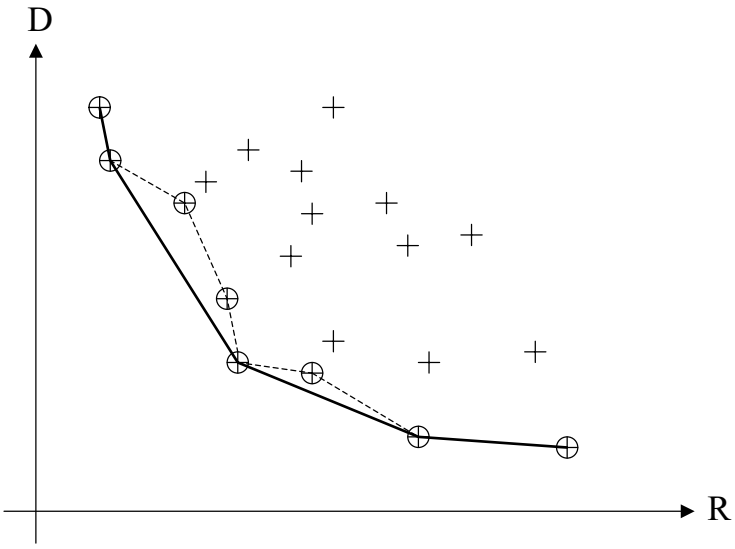


Figure 1.4: Operational rate-distortion function (ORDF). All (R, D)-points are indicated as plus signs. Members of ORDF are circled. The solid line is the convex envelope for the ORDF.

1.3 Contributions of this work

The focus of this work is on the development of an operational rate-distortion optimal frame based encoder. In traditional frame based compression the suboptimal vector selection, the quantization, and the entropy encoding is performed in a sequential order. Previously [43], we have developed the Simultaneous Selection and Quantization (SSQ) algorithm. As the name of the algorithm indicates, the vector selection and the quantization are performed in the same operation. The entropy encoding is made subsequently. The SSQ algorithm finds the minimum distortion representation, subject to a *sparsity* constraint[2]. This work presents a method for finding the binary representation of a one-dimensional input signal in *one operation*. Since the entropy encoding is made in the same step, we can change the optimization problem to be *bit rate* constrained instead of being sparsity constrained. In a lossy compression approach, distortion and rate are two of three important entities, thus finding the optimal solution of the distortion-rate problem is highly relevant. The third important entity in lossy compression is the *time* consumed by the encoder and decoder. In frame based coding, the time needed to decode the binary representation is negligible, as long as the encoder parameters are known. The time needed to find the optimal binary representation of the signal can be large, due to the fact that it increases exponentially with respect to increasing frame and signal block dimensions. Nevertheless, the problem is not unsolvable. As this work shows, the optimal solution can be found, by setting the right parameter values and by using search techniques that saves time.

The major contributions of this thesis are:

- Development of a theoretical framework for rate-distortion optimal frame based compression.
- Development of an operational rate-distortion encoding (ORDE) algorithm. The optimality depends on the given frame, the given codeword tables for allowable coefficient values and indices, and the Lagrangian multiplier constant. In addition, it depends on an upper bound on the number of selected vectors per signal block, in order to keep the complexity at an acceptable level.
- Development of an iterative training scheme that designs a well fitted frame and optimizes the variable codeword tables with respect to the presented class of signal.

[2]The distortion in [43] was the l_1-norm of the error signal, and not the l_2-norm as in this work.

- Demonstration of capabilities by presenting experiments on AR(1) processes and ECG waveforms.
- Comparisons to Rate-Distortion Optimized Matching Pursuit.

A major part of the listed contributions are published in [47, 46, 45, 44].

Chapter 2

Frame based compression and Matching Pursuit

In this chapter we present frame based coding in more detail. In Section 2.1 we start by considering the minimization problem for a traditional block oriented frame based coding scheme. We also examine the complexity involved in finding the optimal solution to the introduced minimization problem, which is an NP-hard problem. Due to the complexity, a class of suboptimal algorithms, Matching Pursuit (MP), is used in the vector selection process. MP is introduced in Section 2.2. Three MP algorithms are presented: Basic Matching Pursuit (BMP), Orthogonal Matching Pursuit (OMP) and Order-Recursive Matching Pursuit (ORMP). In Section 2.3, previous work on frame based coding is presented.

2.1 The minimization problem and its complexity

In most publications dealing with frame based coding, the attention has been fixed on finding the *sparse* representation that minimizes the overall signal distortion, D. We have a sparse representation when the number of nonzero coefficients is much smaller than the number of zero valued coefficients. A sparse coefficient vector can be found by using a vector selection procedure, where the nonzero coefficients belong to the selected frame column vectors. Let D_{vs} be the distortion introduced by the vector selection, and let S be the number of nonzero coefficients in the representation of the input signal. The optimization problem can be formulated as

$$\begin{aligned} \min \quad & D_{vs} \\ \text{s.t.} \quad & S = S_{budget}, \end{aligned} \tag{2.1}$$

where S_{budget} is an upper bound on the total number of nonzero coefficients. D_{vs} is the sum of the distortions for all signal blocks considered. There are no quantization error included to the distortion in the objective function, since quantization is not taken into consideration in this chapter. For the same reason, bit rate is not included in (2.1). Let the input signal consist of L blocks. The distortion of block l, $D_{vs,l}$ is defined by

$$D_{vs,l}(\mathbf{w}_l) = \left\|\mathbf{x}_l - \hat{\mathbf{x}}_l\right\|^2 = \left\|\mathbf{x}_l - \mathbf{F}\,\mathbf{w}_l\right\|^2. \tag{2.2}$$

The approximation signal, $\hat{\mathbf{x}}_l$, is the reconstruction of the original signal using not quantized coefficients. Now, the problem in 2.1 can be written as

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_L} \quad & \sum_{l=1}^{L} \left\|\mathbf{x}_l - \mathbf{F}\,\mathbf{w}_l\right\|^2 \\ \text{s.t.} \quad & \sum_{l=1}^{L} S_l(\mathbf{w}_l) = S_{budget}, \end{aligned} \tag{2.3}$$

where $S_l(\mathbf{w}_l)$ denotes the number of nonzero coefficients in $\mathbf{w}_l$. This problem is extremely hard to solve, due to the nonlinearity in the objective function of (2.3) and the dependency between the signal blocks in the constraint. We "relax" the problem, which means that we replace it by a simpler problem, by introducing a coefficient budget for each signal block, $S_{budget,l}$, such that $\sum_{l=1}^{L} S_{budget,l} = S_{budget}$.[1] The relaxation will make the signal blocks independent, and the problem of (2.3) can be decomposed into L independent subproblems, where subproblem l can be written as

$$\begin{aligned} \min_{\mathbf{w}_l} \quad & \left\|\mathbf{x}_l - \mathbf{F}\,\mathbf{w}_l\right\|^2 \\ \text{s.t.} \quad & S_l(\mathbf{w}_l) = S_{budget,l}. \end{aligned} \tag{2.4}$$

[1] Whether the number of coefficients per signal block is the same for all blocks or not, is not discussed here.

The problem in (2.4) is still hard to solve. An optimal solution could be found by doing a full search among all $\binom{K}{S_l}$ solutions per signal block. K is the number of frame column vectors. Assume that $K = 32$ and $S_l = \{1, 2, 3, 4\}$. The number of solutions to be computed *per* signal block would be $\{32, 496, 4960, 35960\}$, respectively. In most practical situations, the computational effort associated with the full search for the optimal solution is too high. Thus, fast heuristic algorithms are introduced.

2.2 Matching Pursuit

Some of the most important vector selection heuristics is the family of Matching Pursuit (MP) algorithms. Originally, MP was the algorithm that in this work is presented as Basic Matching Pursuit (BMP). All MP algorithms are greedy algorithms, where frame vectors are selected one by one, iteratively. Here, we will examine Orthogonal Matching Pursuit (OMP) and Order Recursive matching Pursuit (ORMP), but first a brief introduction to the Basic MP algorithm.

2.2.1 Basic Matching Pursuit

Basic Matching Pursuit (BMP), introduced in a digital signal processing context in [29], works as follows: First, the input signal vector is noted as the initial *residual* vector, $\mathbf{r}_0 \triangleq \mathbf{x}_l$. At the first iteration, the frame column vector that best fits the input signal vector, $\mathbf{f}_{k_1}$, is selected: $k_1 \in \{1, \ldots, K\}$ is chosen such that $|\mathbf{x}^T \mathbf{f}_{k_1}|$ is maximum. All frame vectors are of unit length. The coefficient, $v_{l,1}$, is found by taking the inner product between the initial residual, $\mathbf{r}_0$, and the selected vector, $\mathbf{f}_{k_1}$. The new residual, $\mathbf{r}_1 = \mathbf{r}_0 - \langle \mathbf{r}_0, \mathbf{f}_{k_1} \rangle \mathbf{f}_{k_1}$, is the input vector for the next iteration. $\langle \mathbf{r}_0, \mathbf{f}_{k_1} \rangle = \mathbf{r}_0^T \mathbf{f}_{k_1}$ is the inner product of $\mathbf{r}_0$ and $\mathbf{f}_{k_1}$. For each iteration, the coefficient and the frame vector index is stored. After m iterations, the nonzero coefficient vector is defined as $\mathbf{v}_l = [\ v_{l,1}\ \ v_{l,2}\ \ \cdots\ \ v_{l,m}\ ]^T$, and the index vector is $\mathbf{i}_l = [\ i_{l,1}\ \ i_{l,2}\ \ \cdots\ \ i_{l,m}\ ]^T = [\ k_1\ \ k_2\ \ \cdots\ \ k_m\ ]^T$. The relation between $\mathbf{v}_l$ and $\mathbf{w}_l$ is simply that $\mathbf{v}_l$ holds the nonzero values of $\mathbf{w}_l$. $\mathbf{i}_l$ indicates where these values are placed in $\mathbf{w}_l$. In Figure 2.1, a pseudo code of the BMP algorithm is presented. The condition in line 9 is a consequence of the sparsity constraint of (2.4). In many cases the condition is replaced by $\|\mathbf{r}_m\| < \varepsilon$, where ε is a small, positive scalar. In these cases, the optimization problem is changed: To minimize the number of selected vectors, S_l, subject to a distortion constraint, ε.

Basic Matching Pursuit

input : $\mathbf{x}_l$, $\mathbf{F}$, S_l
output: $\mathbf{v}_l$, $\mathbf{i}_l$

$\mathbf{r}_0 \leftarrow \mathbf{x}_l$
$m \leftarrow 1$
while $m \leq S_l$ **do**
    $k_m \leftarrow \arg\max_{k\in\{1,\ldots,K\}} |\langle \mathbf{r}_{m-1}, \mathbf{f}_k \rangle|$
    $i_{l,m} \leftarrow k_m$
    $v_{l,m} \leftarrow \langle \mathbf{r}_{m-1}, \mathbf{f}_{k_m} \rangle$
    $\mathbf{r}_m \leftarrow \mathbf{r}_{m-1} - v_{l,m}\mathbf{f}_{k_m}$
    $m \leftarrow m + 1$
end

Figure 2.1: A pseudo code of the Basic Matching Pursuit algorithm.

In a compression scheme, the index vector, $\mathbf{i}_l$, and a *quantized* version, $\tilde{\mathbf{v}}_l$, of the coefficient vector $\mathbf{v}_l$, is entropy encoded. The reconstruction of the signal vector is simple and consists of a very few steps, as shown in the pseudo code in Figure 2.2.

2.2.2 Orthogonal Matching Pursuit

Basic MP has a drawback. It repeatedly optimizes over all frame vectors, which can result in reselecting a vector already selected in the past iterations. Orthogonal Matching Pursuit (OMP) [11, 39] is an improvement of BMP avoiding this problem. The difference lays in the way the residual is computed: In each iteration the residual is made orthogonal to *all* previously selected frame vectors, and not only to the last one added. The principle of vector selection in each iteration is the same, where the frame vector that reduces the residual the most is selected. But, due to the residual orthogonalization, only the so far non-selected frame vectors are of interest. In addition, OMP always converges after a finite number of iterations in finite dimensional spaces [11], which is not necessarily the case for BMP. In the literature, OMP is sometimes called Modified Matching Pursuit (MMP) [2].

BMP Reconstruction

input : $\tilde{\mathbf{v}}_l$, $\mathbf{i}_l$, $\mathbf{F}$
output: $\tilde{\mathbf{x}}_l$

$\tilde{\mathbf{w}}_l \leftarrow \mathbf{0}$
$S_l \leftarrow$ length of $\mathbf{i}_l$
for $m = 1$ **to** S_l **do**
    $\tilde{w}_{l,i_{l,m}} \leftarrow \tilde{v}_{l,m}$
end
$\tilde{\mathbf{x}}_l \leftarrow \mathbf{F}\tilde{\mathbf{w}}_l$

Figure 2.2: The reconstruction procedure when BMP is used as the vector selection algorithm.

2.2.3 Order Recursive Matching Pursuit

Order Recursive Matching Pursuit (ORMP) was introduced in [18] as Fast Orthogonal Matching Pursuit (FOMP). Even though ORMP includes orthogonalization, it is not a fast implementation of OMP, but rather a fast version of the "orthogonal least squares method" in [6]. Fully Orthogonal MP is another name of ORMP. While OMP orthogonalize the residual with respect to the space spanned by the selected vectors, ORMP orthogonalize *the remaining set of frame vectors* with respect to the set of previously selected vectors. In each iteration this is done *prior* to the selection, and the coefficient connected to the orthogonalized frame vector, $v^o_{l,m}$, is kept: At iteration m, $v^o_{l,m} = \langle \mathbf{r}_{m-1}, \hat{\mathbf{f}}_{k_m} \rangle$, where $\hat{\mathbf{f}}_{k_m}$ is the orthogonalized frame vector. In Figure 2.3 the ORMP algorithm is shown. The first vector selected is, as in BMP, the frame vector that best fits the input signal, $\mathbf{x}_l$. In each of the following iterations, the Gram-Schmidt (GS) Process [3] is used to find the orthonormalization of a so far not selected frame vector, $\mathbf{f}_k$, with respect to the selected set of vectors. The algorithm terminates after S_l iterations. Note that the coefficient vector to be quantized and entropy encoded, $\mathbf{v}^o_l = [\ v^o_{l,1} \quad v^o_{l,2} \quad \cdots \quad v^o_{l,S_l}\ ]^T$, is the one connected to the orthogonalized set of selected vectors. This involves an orthonormalization procedure to be implemented in the decoder, resulting in an increase of computational effort in the reconstruction scheme. Figure 2.4 shows the steps needed to find the reconstructed signal vector, $\tilde{\mathbf{x}}_l$, when $\mathbf{i}_l$ and $\tilde{\mathbf{v}}^o_l$ are the representation data transmitted. If the increase in the decoding computational time is unacceptable, we could move the computational effort to the encoding side, and let the original frame vector coefficients, $\mathbf{v}_l$, be quan-

tized and entropy encoded. The reconstruction procedure would then be equal to the one in Figure 2.2. As for BMP, this would involve a major disadvantage: The coefficients of the original frame vector can become arbitrarily large and cause problem for the quantizer and the entropy encoder.
In general, the ORMP gives better results than OMP, and especially, the BMP algorithm [2, 9].

Order Recursive Matching Pursuit

input : $\mathbf{x}_l$, $\mathbf{F}$, S_l
output: $\mathbf{v}_l^o$, $\mathbf{i}_l$

$\mathbf{r}_0 \leftarrow \mathbf{x}_l$
$m \leftarrow 1$
while $m \leq S_l$ **do**
    $k_m \leftarrow \arg\max_{k\in\{1,\ldots,K\}\setminus k\in\mathbf{i}_l} |\langle \mathbf{r}_{m-1}, \hat{\mathbf{f}}_k\rangle|$
    where $\hat{\mathbf{f}}_k = \frac{\mathbf{f}_k - \sum_{\xi=1}^{m-1}\langle \mathbf{f}_k, \hat{\mathbf{f}}_{k_\xi}\rangle \hat{\mathbf{f}}_{k_\xi}}{\|\mathbf{f}_k - \sum_{\xi=1}^{m-1}\langle \mathbf{f}_k, \hat{\mathbf{f}}_{k_\xi}\rangle \hat{\mathbf{f}}_{k_\xi}\|}$
    $i_{l,m} \leftarrow k_m$
    $v_{l,m}^o \leftarrow \langle \mathbf{r}_{m-1}, \hat{\mathbf{f}}_{k_m}\rangle$
    $\mathbf{r}_m \leftarrow \mathbf{r}_{m-1} - v_{l,m}^o \hat{\mathbf{f}}_{k_m}$
    $m \leftarrow m + 1$
end

Figure 2.3: A pseudo code of the Order Recursive Matching Pursuit algorithm.

2.3 Previous work on frame based coding

Considerable research has been performed in the area of frame based coding. The most important papers, and the papers most relevant to this work, will be reviewed in the following. First, we should repeat for ourselves the motivation of using frames, or overcomplete dictionaries: *To get a sparse representation of the input signal, that when decoded results in a good approximation of the signal.* Using an overcomplete dictionary increases the likelihood of finding a sparse representation resulting in a lower distortion, compared to using a transform. Thus, the focus of the research has mainly been: 1) Vector selection: Given a frame $\mathbf{F}$, how to get a good sparse representation in a reasonable

ORMP Reconstruction

input : $\tilde{\mathbf{v}}_l^o$, $\mathbf{i}_l$, $\mathbf{F}$
output: $\tilde{\mathbf{x}}_l$

$S_l \leftarrow$ length of $\mathbf{i}_l$
if $S_l = 0$ **then**
  $\tilde{\mathbf{x}}_l \leftarrow \mathbf{0}$
else
  $\mathbf{q}_{l,1} \leftarrow \hat{\mathbf{f}}_{i_{l,1}} \leftarrow \mathbf{f}_{i_{l,1}}$
  for $m = 2$ **to** S_l **do**
    $\mathbf{q}_{l,m} \leftarrow \hat{\mathbf{f}}_{i_{l,m}} \leftarrow \frac{\mathbf{f}_{i_{l,m}} - \sum_{\xi=1}^{m-1} \langle \mathbf{f}_{i_{l,m}}, \hat{\mathbf{f}}_{i_{l,\xi}} \rangle \hat{\mathbf{f}}_{i_{l,\xi}}}{\| \mathbf{f}_{i_{l,m}} - \sum_{\xi=1}^{m-1} \langle \mathbf{f}_{i_{l,m}}, \hat{\mathbf{f}}_{i_{l,\xi}} \rangle \hat{\mathbf{f}}_{i_{l,\xi}} \|}$
  end
  $\tilde{\mathbf{x}}_l \leftarrow \mathbf{Q}_l \tilde{\mathbf{v}}_l^o$ where $\mathbf{Q}_l = [\, \mathbf{q}_{l,1} \;\; \mathbf{q}_{l,2} \;\; \cdots \;\; \mathbf{q}_{l,S_l} \,]$
end

Figure 2.4: The reconstruction procedure when ORMP is used as the vector selection algorithm.

amount of time, and 2) Frame improvement: How to find a frame $\mathbf{F}$, that best fits a given class of signals.

2.3.1 Vector selection

BMP was developed by Mallat and Zhang [29] in 1993. After this, many MP variants has been developed. We have already mentioned Orthogonal MP by Davis [11] and in parallel by Pati et al. [39]. ORMP was proposed in different journals with different names: In 1989, Chen et al. [6] presented the Orthogonal Least Squares (OLS) method, and a fast implementation of this (FOLS) in 1995 [7]. In the same year, Natarajan [34] proposed a greedy algorithm in an applied mathematics journal. Gharavi-Alkhansari and Huang presented ORMP as Fast Orthogonal Matching Pursuit (FOMP) in 1998. The subject of finding the least computational complex version of the MP algorithms has resulted in several papers. Cotter et al. [9, 8] presents fast versions of BMP, OMP, and ORMP, and compare the performance of these algorithms. OMP and ORMP are more complex than BMP, but they have a better sparsity performance subject to a distortion constraint.

In addition to obtaining a well selected set of vectors, it is important to look at the coefficient quantization, since both selection and quantization introduce

distortion to the signal. This is more important when using coarse quantization. Quantized Matching Pursuit (QMP) was proposed by Vetterli and Kalker [57]. The difference between BMP and QMP is: In each iteration of QMP, the coefficient of the selected vector is quantized. The computation of the residual (to be the input vector of next iteration) involves the quantized coefficient, and not the continuous valued one. In Figure 2.1, this would be the same as replacing line 5 by $\mathbf{r}_m \leftarrow \mathbf{r}_{m-1} - Q[\langle \mathbf{r}_{m-1}, \mathbf{f}_{k_m} \rangle]\mathbf{f}_{k_m}$, where $Q[\cdot]$ represents a scalar quantization. The motivation for using QMP is to reduce the propagation of the quantization error to subsequent iterations. Goyal et al. [22] focus on the *consistency* of the reconstruction algorithm. Let $\tilde{\mathbf{w}}_{ex}$ be the quantized coefficient vector of signal vector $\mathbf{x}_{ex}$, for an arbitrary frame based encoder. The reconstruction of $\mathbf{x}_{ex}$ is $\tilde{\mathbf{x}}_{ex}$. A reconstruction algorithm is called a *consistent* reconstruction algorithm if $\tilde{\mathbf{w}}_{ex}$ would be the quantized coefficient vector, both with $\mathbf{x}_{ex}$ and $\tilde{\mathbf{x}}_{ex}$ as inputs. This is not always the situation when quantizing coefficients of nonorthogonal frame vectors.[2] By making a consistent reconstruction algorithm for QMP, Goyal and Vetterli [21] shows experimentally a rate-distortion improvement. Frossard et al. [15] look at quantization of coefficients in another way. They use *a posteriori* quantization, which implies that the quantization does not influence the MP expansion. The new idea in this work is on how the coefficients are quantized with respect to their placement in the coefficient vector; The quantization region decreases exponentially, according to the decay of coefficient value range in each iteration of MP. The quantization steps are made smaller for each iteration, and the quantization noise is reduced. Another method on MP quantization is proposed by Caetano et al. [5]. In this work, there are no coefficients to be quantized. They use generalized bit-plane decomposition, where the representation of the signal vector is pairs of selected vector indices and bit-plane integers. The selection and quantization are made simultaneously. The algorithm has a slightly better rate-distortion performance than BMP, and a much better rate control.

The Matching Pursuit procedure can be viewed as a greedy Depth-First-Search (DFS) [30] in a solution tree. How this tree is organized is described in Section 4.3. You start in the root node. At each iteration of MP, you go one step down in the tree, to the node that represent the best fitting frame vector according to the current residual. When the stop criterion of MP is met, the search is terminated. The sequential and greedy nature of MP prevents the possibility of getting a better solution when using the second best solution in first iteration, that in the next phase is a better selection when two or

[2] Not a problem when using ORMP, when the coefficients connected to the *orthogonalized* frame vectors are used.

more frame vectors are selected. Both Cotter and Rao [10], and Skretting and Husøy [55] propose tree-based partial search algorithms based on MP. But, instead of keeping only the best solution at each iteration, a small set of the better solutions is kept. When the stop criterion is met, the best combination of frame vector is returned. The time complexity increases, but the performance is better than MP, especially for the algorithms based on OMP and ORMP.

Skretting et al also introduced Global Matching Pursuit (GMP) [54] in an image compression scheme. Images are largely non-stationary, and parts with many details should be allocated more representation vectors than relatively smooth parts, in order to put the total distortion to a minimum. In GMP, the first iteration of MP is done for all L signal block, before starting the second iteration, and so on. The inner products between the residual and all frame vectors are found. The inner products with the largest absolute values, but maximum one per signal block, are stored in a priority queue of length S_{budget}. For each iteration, the new inner products are compared to the queue, and replaced by inner products with a smaller absolute values. GMP requires more memory space than BMP. Without a large increase in computation time, GMP provides a better sparsity performance through its well defined distribution of selected vectors among the L signal blocks.

Up to now, only MP or similar sequential selection algorithms are mentioned. Another heuristic, based on a parallel selection principle, is the FOCal Underdetermined System Solver (FOCUSS) algorithm [19]. Intuitively, the algorithm can be explained by noting that there is a competition between the columns of $\mathbf{F}$ to represent $\mathbf{x}_l$. In each iteration, certain columns get emphasized while others are deemphasized. In the end a few columns survive to represent $\mathbf{x}_l$ providing a sparse solution. FOCUSS is computational more expensive than BMP, OMP and ORMP, but has a better performance. A comparison between the MP algorithms and FOCUSS is presented by Adler et al. [2]. Extensive experiments on speech signals, ECG signals, and images where these four selection algorithms are used, are performed by Engan [12].

2.3.2 Frame improvement

In order to get a well fitted sparse representation, it is important to design a frame that is well suited to the current class of signals. Engan et al. [13, 14] developed the Method of Optimal Directions (MOD). Let $\mathbf{X}$ be an input signal matrix consisting of L column vectors, $\mathbf{X} = [\ \mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_L\ ]$, and let $\mathbf{W} = [\ \mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_L\ ]$ be the corresponding set of unquantized coefficient

vectors. MOD is an iterative procedure consisting of two steps: 1) $\mathbf{F}$ and $\mathbf{X}$ are known. Find $\mathbf{W}$ by using a vector selection algorithm. 2) $\mathbf{W}$ and $\mathbf{X}$ are known. Find the new frame by $\mathbf{F} = \mathbf{X}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}$. MOD is described in more detail in Section 5.2.2. A more general view and use of MOD is given by Aase et al. [1].

Through experiments, finding the most suitable frame depends on the number of vectors selected per signal block. Thus, Multi Frame Compression [14] is developed. Consider a set of S frames; $\mathbf{F}_1, \ldots, \mathbf{F}_S$. There are S entropy coders, one corresponding to each frame. For a particular class of signals, the frames and the entropy coders are designed to fit the selection of $1, \ldots, S$ frame vectors, respectively. In the designed scheme there is not only selection of vectors, but a *selection of frame*, as well. This implies an increase in the side information, due to the need of frame number information at the decoder, but the rate-distortion performance is better.

So far, we have talked about *block oriented* frames. Skretting et al. [56, 53] have introduced the concept of Overlapping Frames. An Overlapping Frame is actually a generalization of critical sampled filter banks and wavelets. The size of $\mathbf{F}$ is now $NO \times K$, where O is a positive integer, or actually the number of signal vectors covered at each time.

Chapter 3

Rate-distortion optimal (RDO) compression

The main idea behind rate-distortion optimized compression is to find the minimum distortion representation of a signal, subject to a given bit budget. Or, the inverse problem; Finding the minimum number of bits required, subject to an upper bound of the distortion. The definitions of distortion and rate are given in Section 3.1. In Section 3.2, the rate-distortion optimization problem for a frame based coding system is presented. Due to the complexity of finding the optimal solution, the hard problem is replaced by a simpler one by using the Lagrangian Multiplier method. Section 3.3 shows the number of solutions for an example of a specific coding scheme, and it demonstrates the combinatorial explosion when increasing the number of selected vectors for every signal block. In Section 3.4 the RDO Matching Pursuit is presented, as it was introduced in [17]. Examples of other work in RDO compression are given in Section 3.5.

3.1 Definitions

We define the bit rate and the distortion for each block to be independent of the other blocks, i.e., the total bit rate, R, and the total distortion, D, is the sum of the rates and the distortions for each block, respectively. The distortion, D_l, for block l is defined by

$$D_l(\tilde{\mathbf{w}}_l) = \left\|\mathbf{x}_l - \tilde{\mathbf{x}}_l\right\|^2 = \left\|\mathbf{x}_l - \mathbf{F}\,\tilde{\mathbf{w}}_l\right\|^2, \tag{3.1}$$

where $\tilde{\mathbf{w}}_l$ is the quantized coefficient vector of block l. Since a large number of the K elements in $\tilde{\mathbf{w}}_l$ is expected to be equal to zero, only the quantized *values* of the nonzero coefficients and their corresponding *indices* are encoded. An End Of Block (EOB) symbol is introduced. After the last nonzero coefficient in each block, the EOB symbol indicates the start of the next block. We use two distinct Variable Length Codeword (VLC) tables of finite length, one with *value* symbols and one with *index* symbols. A VLC table contains the symbols and the bit patterns, or codewords, used to represent the corresponding symbols. The EOB symbol is added as the last term in the *index* codeword table, thus the EOB codeword should not be confused with any of the *index* codewords. The rate, R_l, for block l is defined by

$$R_l(\tilde{\mathbf{w}}_l) = \sum_{k \in \mathbf{i}_l} (R_{l,k}^{val} + R_{l,k}^{ind}) + R^{EOB}, \tag{3.2}$$

where $\mathbf{i}_l$ is a vector of indices of the nonzero coefficients, $R_{l,k}^{val}$ and $R_{l,k}^{ind}$ are the number of bits used to encode the *value* and the *index* for the k^{th} coefficient, respectively, and R^{EOB} is the number of bits needed to represent the EOB symbol. If there are no vectors selected to represent signal block l, only the EOB symbol needs to be transmitted and $R_l(\tilde{\mathbf{w}}_l) = R^{EOB}$.

3.2 The minimization problem

Our goal is, for a given set of signal blocks, $\mathbf{X} = [\, \mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \cdots \;\; \mathbf{x}_L \,]$, to find the appropriate set of coefficient vectors, $\tilde{\mathbf{W}} = [\, \tilde{\mathbf{w}}_1 \;\; \tilde{\mathbf{w}}_2 \;\; \cdots \;\; \tilde{\mathbf{w}}_L \,]$, so that the distortion of the reconstruction is minimized subject to a given bit budget, R_{budget}. We can formulate our initial optimization problem as

$$\begin{aligned} \min_{\tilde{\mathbf{w}}_1,\ldots,\tilde{\mathbf{w}}_L} \quad & \sum_{l=1}^{L} D_l(\tilde{\mathbf{w}}_l) \\ \text{s.t.} \quad & \sum_{l=1}^{L} R_l(\tilde{\mathbf{w}}_l) \leq R_{budget}. \end{aligned} \tag{3.3}$$

This problem looks similar to the one in (2.3), but it is quite different. Both problems are nonlinear, due to the norm sign in the definition of the distortion in the objective function. But, while (2.3) is constrained by a sparsity budget, (3.3) is constrained by a bit rate budget. (3.3) is harder to solve than (2.3).

Due to the quantized valued $\tilde{\mathbf{w}}_l$, (3.3) is an integer problem [59]. In order to make the problem solvable, we relax it by using the *Lagrangian Multiplier Method*: The constraint is added to the objective function, weighted by the Lagrangian Multiplier, λ. The following unconstrained optimization problem is formulated as

$$\min_{\tilde{\mathbf{w}}_1,\ldots,\tilde{\mathbf{w}}_L} \Big(\sum_{l=1}^{L} D_l(\tilde{\mathbf{w}}_l) + \lambda \Big(\sum_{l=1}^{L} \big(R_l(\tilde{\mathbf{w}}_l) \big) - R_{budget} \Big) \Big), \tag{3.4}$$

for $\lambda \in \mathbb{R}^+$. After (3.4) is solved to optimality for a chosen λ-value, the appropriate λ needs to be iteratively adjusted down to a value where further reduction of λ would cause an overrun of the bit budget, R_{budget}. When this λ-value is found, we have the optimal solution of (3.3).

From now on, we focus on finding the rate-distortion convex envelope. The convex envelope will tell us the quality of the coding scheme, where the minimum distortion is a function of a given bit rate. We leave the focus on the bit budget, and R_{budget} can be removed from the problem formulation, since it represents a constant term in the objective function. The problem can now be formulated as

$$\begin{aligned} &\min_{\tilde{\mathbf{w}}_1,\ldots,\tilde{\mathbf{w}}_L} \Big(\sum_{l=1}^{L} D_l(\tilde{\mathbf{w}}_l) + \lambda \sum_{l=1}^{L} R_l(\tilde{\mathbf{w}}_l) \Big) \\ =&\sum_{l=1}^{L} \Big[\min_{\tilde{\mathbf{w}}_l} \Big(D_l(\tilde{\mathbf{w}}_l) + \lambda R_l(\tilde{\mathbf{w}}_l) \Big) \Big]. \end{aligned} \tag{3.5}$$

The signal blocks are independent and hence the minimization can be moved inside the summation operator. By using the formulation in (3.5) with several different λ-values, we can find segments of the rate-distortion convex envelope. λ is the "turning knob" used to change between low and high bit rate coding.

Figure 3.1 illustrates our new focus. Contrary to traditionally frame based coding, illustrated in Figure 1.3, we will find the operational rate-distortion *optimal* representation, Y^{opt}, of an input, X, given the frame $\mathbf{F}$, the Lagrangian multiplier λ, and the bit rate of coefficient *values* and *indices*. Y^{opt} is a bit stream which is put into the channel in a communication system. (See Figure 1.1.) The next section discusses the complexity of the rate-distortion optimal procedure.

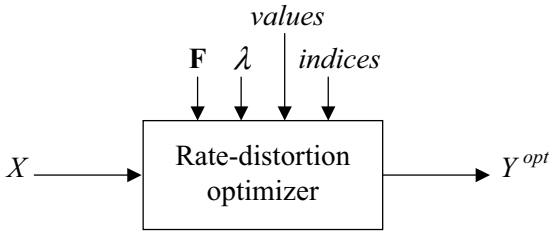


Figure 3.1: An operational rate-distortion optimizer finds the best possible set of representation symbols, Y^{opt}, for an input signal, X, given the frame $\mathbf{F}$, the Lagrangian multiplier λ, and the bit rate of coefficient *values* and *indices*.

3.3 Problem size considerations

In (3.5), the signal blocks are independent. It would be preferable to have a similar situation within each block, that is, for the rate and distortion to be minimized with respect to each of the coefficients of $\tilde{\mathbf{w}}_l$, independently. Unfortunately, the distortion is dependent on all coefficients of $\tilde{\mathbf{w}}_l$, due to the nonorthogonal decomposition, and this dependency can not be decoupled. Hence the minimization can not be carried out coefficient by coefficient, but needs to be carried out for every combination of coefficient values. The complexity of the minimization in (3.5) is high, since for every block we need to search among all possible ways to place M nonzero elements in a coefficient vector of length K. $M = 1, 2, \ldots, M_{max}$, where M_{max} is the largest number of nonzero coefficients we choose to use for a single signal block. The number of combinations with M nonzero coefficients is $\binom{K}{M}$. In addition, each nonzero coefficient can take on a given number of different *value* symbols, J. For each combination, the number of solutions is J^M. Thus, the total number of different solutions is $\sum_{M=0}^{M_{max}} \binom{K}{M} J^M$. In all practical cases, $M_{max} << K$, due to the sparse representation idea. In this work, frames with dimensions 16×32 are used, thus $K = 32$. Assume that all nonzero coefficient can take on $J = 32$ different values. For $M = \{1, 2, 3, 4\}$, $\binom{K}{M} = \{32, 496, 4960, 35960\}$ and $J^M = \{32, 1024, 32768, 1048576\}$. A way to avoid the latter combinatorial explosion is presented in Chapter 4, where we introduce a computationally efficient algorithm that allows us to find the optimal solution. This is achieved by first moving to an orthogonal space, where the distortion becomes the sum of the coefficient distortions, and then reformulating the optimization problem in that space. These two steps allow us to reduce the J^M complexity above to an $JMM!$ complexity. By choosing the right search strategy, the problem size can be further reduced from $JMM!$ to $JM!$ for each combination of vectors.

Together this is a tremendous reduction, keeping in mind that $M << J$ in all practical cases. For example, in the above case with $M = 4$, this reduction in problem size is a factor of $32^4/(32 \cdot 4!) = 1365$.

3.4 RDO Matching Pursuit

Rate-distortion optimized Matching Pursuit was first mentioned by Vetterli and Kalker [57]. But, elaboration and experimental work was first made by Gharavi-Alkhansari [17], thus this section is based on his work. First, it should be pointed out that RDO MP is *not* the RD optimal solution to the problem in (3.5). RDO MP is a greedy algorithm where the focus is to get a minimum *rate-distortion* solution rather than a minimum *distortion* subject to a sparsity constraint. RDO MP finds a suboptimal solution to the problem in (3.5).
For signal block l, the *cost* is defined by $C_l \triangleq D_l + \lambda R_l$, and $C_{l,m}$ is the cost at iteration m in the RDO MP algorithm. When selecting a frame vector, the bit rate associated with representing the frame *index* is known, when the VLC tables are set. By finding a corresponding quantized coefficient among the set of predefined values, we know the total rate of adding the current vector to the selected set. We can find the resulting distortion reduction, due to the distortion definition in (3.1). Both the selection criterion and the stop criterion are different in RDO MP compared to MP. The frame vector selected at iteration m is the one that, together with the quantized coefficient, generates the largest cost reduction. Or in other words, the vector that maximizes $\Delta C = C_{l,m-1} - C_{l,m}$. The algorithm stops when ΔC is negative. The last selected vector, resulting in $\Delta C < 0$, is not added to the set of selected vectors.
Gharavi-Alkhansari presents two algorithms, the RDO Basic Matching Pursuit (RDO BMP) and the RDO Ordered Recursive Matching Pursuit (RDO ORMP[1]). The RDO BMP algorithm is presented in Figure 3.2. Compared to BMP, the sparsity constraint in the input is exchanged by a VLC table of *values*, a VLC table of frame vector *indices*, and a Lagrangian multiplier, λ. The tables of *values* and *indices* has J and K codewords, respectively. The output is now the *quantized* coefficient vector, $\tilde{\mathbf{v}}_l$, and the corresponding vector of the selected set of indices, $\mathbf{i}_l$. The RDO ORMP algorithm is shown in Figure 3.3. It is similar to RDO BMP, except from the orthogonalization of the frame vectors in line 6. This follows the principles of ORMP. The output is now the quantized coefficients connected to the *orthogonalized* set of selected frame vectors, $\tilde{\mathbf{v}}_l^o$, together with $\mathbf{i}_l$. In [17] the numbers of selected vectors for each signal block are encoded and sent to the decoder. In this work, the EOB symbol is used to separate the signal blocks from each other.

[1] ORMP is denoted as Fully-Orthogonal Matching Pursuit in [17].

RDO Basic Matching Pursuit

input : $\mathbf{x}_l$, $\mathbf{F}$, $values$, $indices$, λ
output: $\tilde{\mathbf{v}}_l$, $\mathbf{i}_l$

$C_{l,0} \leftarrow \|\mathbf{x}_l\|^2 + \lambda R^{EOB}$
$\mathbf{r}_0 \leftarrow \mathbf{x}_l$
$\delta_0 \leftarrow R^{EOB}$
$m \leftarrow 1$
do
  $[k_m, j_m] \leftarrow \arg\min_{k\in\{1,\ldots,K\},j\in\{1,\ldots,J\}} \left\{ \|\mathbf{r}_{m-1} - values(j)\mathbf{f}_k\|^2 + \lambda(\delta_{m-1} + R_{values(j)} + R_{indices(k)}) \right\}$
  $C_{l,m} \leftarrow \|\mathbf{r}_{m-1} - values(j_m)\mathbf{f}_{k_m}\|^2 + \lambda(\delta_{m-1} + R_{values(j_m)} + R_{indices(k_m)})$
  if $C_{l,m} \leq C_{l,m-1}$ **then**
    $i_{l,m} \leftarrow k_m$
    $\tilde{v}_{l,m} \leftarrow values(j_m)$
    $\mathbf{r}_m \leftarrow \mathbf{r}_{m-1} - \tilde{v}_{l,m}\mathbf{f}_{i_{l,m}}$
    $R^{val}_{l,m} \leftarrow R_{values(j_m)}$
    $R^{ind}_{l,m} \leftarrow R_{indices(k_m)}$
    $\delta_m \leftarrow \delta_{m-1} + R^{val}_{l,m} + R^{ind}_{l,m}$
    $m \leftarrow m + 1$
  end
while $C_{l,m} \leq C_{l,m-1}$

Figure 3.2: A pseudo code of the Rate-Distortion Optimized Basic Matching Pursuit (RDO BMP) algorithm.

RDO Ordered Recursive Matching Pursuit

input : $\mathbf{x}_l$, $\mathbf{F}$, *values*, *indices*, λ
output: $\tilde{\mathbf{v}}_l^o$, $\mathbf{i}_l$

$C_{l,0} \leftarrow \|\mathbf{x}_l\|^2 + \lambda R^{EOB}$
$\mathbf{r}_0 \leftarrow \mathbf{x}_l$
$\delta_0 \leftarrow R^{EOB}$
$m \leftarrow 1$
do
  $[k_m, j_m] \leftarrow \arg\min_{k\in\{1,\ldots,K\}\setminus k\in\mathbf{i}_l, j\in\{1,\ldots,J\}} \Big\{ \|\mathbf{r}_{m-1} - values(j)\hat{\mathbf{f}}_k\|^2 + \lambda(\delta_{m-1} + R_{values(j)} + R_{indices(k)}) \Big\}$

  where $\hat{\mathbf{f}}_k = \frac{\mathbf{f}_k - \sum_{\xi=1}^{m-1}\langle \mathbf{f}_k, \hat{\mathbf{f}}_{k_\xi}\rangle \hat{\mathbf{f}}_{k_\xi}}{\|\mathbf{f}_k - \sum_{\xi=1}^{m-1}\langle \mathbf{f}_k, \hat{\mathbf{f}}_{k_\xi}\rangle \hat{\mathbf{f}}_{k_\xi}\|}$

  $C_{l,m} \leftarrow \|\mathbf{r}_{m-1} - values(j_m)\hat{\mathbf{f}}_{k_m}\|^2 + \lambda(\delta_{m-1} + R_{values(j_m)} + R_{indices(k_m)})$
  if $C_{l,m} \leq C_{l,m-1}$ **then**
    $i_{l,m} \leftarrow k_m$
    $\tilde{v}_{l,m}^o \leftarrow values(j_m)$
    $\mathbf{r}_m \leftarrow \mathbf{r}_{m-1} - \tilde{v}_{l,m}^o \hat{\mathbf{f}}_{i_{l,m}}$
    $R_{l,m}^{val} \leftarrow R_{values(j_m)}$
    $R_{l,m}^{ind} \leftarrow R_{indices(k_m)}$
    $\delta_m \leftarrow \delta_{m-1} + R_{l,m}^{val} + R_{l,m}^{ind}$
    $m \leftarrow m + 1$
  end
while $C_{l,m} \leq C_{l,m-1}$

Figure 3.3: A pseudo code of the Rate-Distortion Optimized Ordered Recursive Matching Pursuit (RDO ORMP) algorithm.

The RDO MP algorithms follow the principle of best RD performance. How the quantizer is defined, plays an important role for the results. In [17], a uniform quantizer is proposed. Under mild conditions it is proven that uniform quantizers are optimal in a rate-distortion sense, when the quantized values are entropy encoded [16]. The entropy encoding used in [17] is arithmetic coding, where only *estimates* of the bit rate is used to find the RDO MP solutions. In our work, Huffman coding [24, 16] is used. The reason for this is that we focus on *operational* RDO coding, i.e., practical optimal coding where all parameters are known before coding.

3.5 Other work in RDO compression

In all compression schemes the rate-distortion behavior is a central entity. In image and video coding the volume of information is large, and the use of lossy compression of digital images and video is very important. A nice overview of rate-distortion methods for image and video compression is given by Ortega and Ramchandran [38]. They start by looking at the fundamentals of RD theory, and illustrate the concept of designing operational RD coders. The rate-distortion techniques used in the image compression standard JPEG [40] and the video standard MPEG [33] are presented. Both JPEG and MPEG are based on the DCT transform, where blocks of 8×8 pixels are transformed. The transformed coefficients are quantized, and a zigzag scan is performed to get the two-dimensional set of coefficients into a one-dimensional vector. Most of the coefficients are quantized to zero, and the nonzero coefficients are mostly present in the beginning of this vector. This is the low pass coefficients, stating the fact that most digital images mainly contain low pass information. But, there may be "outliers" of nonzero coefficients representing higher frequencies. Entropy encoding of these "outliers" is more expensive than of the low pass coefficients, from a bit rate point of view. In a rate-distortion setting, one may get better results by quantizing these coefficients to zero. Ramchandran and Vetterli [42] presents an RDO Fast Thresholding algorithm that thresholds (or put to zero) all "outliers" that generate a worse RD performance without thresholding. The algorithm is completely compatible to standard JPEG and MPEG decoders. An in-depth study of RD based video compression is made by Schuster and Katsaggelos [50], which includes exhaustive research in optimal video frame compression and object boundary encoding [49].

In our work, we compress electrocardiograms (ECG) using an operational rate-distortion optimal method. This is also done by Nygaard [36, 37] by a time domain algorithm. She uses shortest path dynamic programming in order to

find the optimal solution. Except from the work described in Section 3.4, the rate-distortion optimization focus in *frame based coding* has received more attention in recent works. Neff and Zachor present operational RD models for MP video coding [35]. They propose two models for *predicting* the rate and distortion. Thus, it is not RDO MP, but faster methods finding solutions near to RDO MP. An operational RDO scheme is presented by Vleeschouwer and Zachor [58]. Here, *in-loop* quantization is used, or in other words, Quantized MP (QMP) in an RDO setting.

Chapter 4

The operational rate-distortion encoder (ORDE)

In this chapter we introduce the core contribution of this work, which is the formulation and solution of our minimization problem. The reader should keep in mind that in this chapter, the frame $\mathbf{F}$, the coefficient *value* and *index* VLC tables, and the Lagrangian Multiplier λ are fixed and known. In Section 4.1, the set of selected frame vectors is orthonormalized. Due to this, the time consumption of finding the rate-distortion optimal solution is radically reduced. A new problem formulation is presented in Section 4.2. This is the same problem as presented in Section 3.2, but with respect to orthonormalized coefficients. Still the complexity of this problem is high. Graph theory techniques presented in Section 4.3 reduce the time consumption radically without changing the problem formulation. In Section 4.4, the problem formulation is changed one more time by adding a constraint. Only ordered selection of frame vectors is allowed. Now, instead of encoding coefficient indices, the *runs* between the indices are encoded. We can use run-length coding (RLC) as a part of the entropy coder, resulting in a reduction in the bit rate. The solution set is smaller, resulting in the chance of higher distortion, but RLC generates a lower bit rate. A summary of the proposed algorithms' complexity is given in Section 4.5.

4.1 Orthonormalization of selected frame vectors

In this section, our starting point is the sum of minimization problems in (3.5), which is formulated as

$$\sum_{l=1}^{L} \Big[\min_{\tilde{\mathbf{w}}_l} \Big(D_l(\tilde{\mathbf{w}}_l) + \lambda R_l(\tilde{\mathbf{w}}_l) \Big) \Big]. \tag{4.1}$$

Let us look at one of these problems, the one including signal block l, $\mathbf{x}_l$. First, we first choose a particular set of M frame vectors and attempt to find an optimal solution for this particular selection. If the M vectors were orthonormal, the total distortion would be the sum of the coordinate distortions. However, this is not necessarily the case. The optimization problem still involves dependent quantizers, and is therefore very complex. We now force this orthonormal condition on the problem by using a *QR-decomposition* [3], which results in a new space where the total distortion is simply the sum of the coordinate distortions. Hence in this space, there are no dependencies among the coefficients and therefore the problem is now an independent quantizer allocation problem, which is much faster to solve as the set of optimal quantizers for each coefficient is also the optimal solution to the overall problem.

For a given combination of M nonzero coefficients, the rate for the *index* symbols is known, since we know the Variable Length Codeword (VLC) table in the entropy coder. Yet, we still do not know the distortion nor the rate for the *value* symbols. It is always the case that $M < N$, due to the sparse representation idea. Let us define a new matrix, $\mathbf{\Phi}_l$, which is the set of selected column vectors of $\mathbf{F}$ (corresponding to the nonzero coefficients). The matrix $\mathbf{\Phi}_l$ will have dimensions $N \times M$ and represent an *undercomplete* set of vectors. Let $\mathbf{v}_l = [\ v_{l,1} \quad v_{l,2} \quad \cdots \quad v_{l,M}\ ]^T$ and $\tilde{\mathbf{v}}_l = [\ \tilde{v}_{l,1} \quad \tilde{v}_{l,2} \quad \cdots \quad \tilde{v}_{l,M}\ ]^T$ be the nonzero coefficients of $\mathbf{w}_l$ and its quantized version $\tilde{\mathbf{w}}_l$, respectively, like it was defined in Section 2.2.1. Since $M < N$, the best reconstruction of signal vector $\mathbf{x}_l$ when unquantized coefficients are used is given by

$$\hat{\mathbf{x}}_l = \mathbf{\Phi}_l \mathbf{v}_l = \mathbf{\Phi}_l (\mathbf{\Phi}_l^T \mathbf{\Phi}_l)^{-1} \mathbf{\Phi}_l^T \mathbf{x}_l, \tag{4.2}$$

due to the Best Approximation Theorem [3]. More details about projection and this theorem are given in Appendix A.2. The error is orthogonal to any vector spanned by the column vectors in $\mathbf{\Phi}_l$. By using the Pythagorean theorem, the distortion in block l can be written as

$$D_l = \|\mathbf{x}_l - \tilde{\mathbf{x}}_l\|^2 = \|\mathbf{x}_l - \hat{\mathbf{x}}_l\|^2 + \|\hat{\mathbf{x}}_l - \tilde{\mathbf{x}}_l\|^2. \tag{4.3}$$

This is illustrated in Figure 4.1, where the dots represent possible values the reconstructed vector, $\tilde{\mathbf{x}}_l$, can take. The first term in (4.3) is the distortion caused by the vector selection. This term is a constant due to (4.2), since both $\mathbf{x}_l$ and $\mathbf{\Phi}_l$ are known when the set of selected vectors is known. Now, we can focus on the second term of the equation, which is the quantization distortion. The column vectors in $\mathbf{\Phi}_l$ are not necessarily orthogonal, and therefore we still have dependencies between the coefficients. By using QR-decomposition, we can get an orthogonal version of $\mathbf{\Phi}_l$. We can write $\mathbf{\Phi}_l = \mathbf{Q}_l\mathbf{R}_l$, where $\mathbf{Q}_l$ is an $N \times M$ matrix with all vectors orthonormal, and $\mathbf{R}_l$ is an $M \times M$ upper triangular matrix. QR-decomposition is explained in detail in Appendix A.1. The nonzero coefficient vector related to the orthonormal basis, $\mathbf{v}_l^o$, and its quantized version, $\tilde{\mathbf{v}}_l^o$, were introduced in Section 2.2.3. Since $\mathbf{R}_l$ is known, the relation between the orthogonal and the nonorthogonal coefficient vector can easily be written as $\mathbf{v}_l^o = \mathbf{R}_l\mathbf{v}_l$. The last term of (4.3) can be written as

$$\begin{aligned} \|\hat{\mathbf{x}}_l - \tilde{\mathbf{x}}_l\|^2 &= \|\mathbf{Q}_l\mathbf{v}_l^o - \mathbf{Q}_l\tilde{\mathbf{v}}_l^o\|^2 \\ &= (\mathbf{v}_l^o - \tilde{\mathbf{v}}_l^o)^T\mathbf{Q}_l^T\mathbf{Q}_l(\mathbf{v}_l^o - \tilde{\mathbf{v}}_l^o) \\ &= \|\mathbf{v}_l^o - \tilde{\mathbf{v}}_l^o\|^2 = \sum_{m=1}^{M}(v_{l,m}^o - \tilde{v}_{l,m}^o)^2. \end{aligned} \tag{4.4}$$

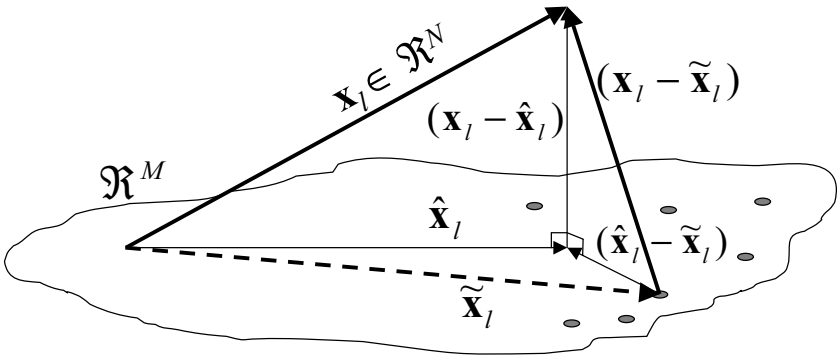


Figure 4.1: A visualization of the orthogonality between the minimum error vector and the subspace spanned by the column vectors in $\mathbf{\Phi}_l$. The dots represent possible values of $\tilde{\mathbf{x}}_l$.

Due to the QR-decomposition of $\mathbf{\Phi}_l$ the computation of (4.2) is simplified.

That is, the projection matrix, $\mathbf{\Phi}_l(\mathbf{\Phi}_l^T\mathbf{\Phi}_l)^{-1}\mathbf{\Phi}_l^T$, can be replaced by $\mathbf{Q}_l\mathbf{Q}_l^T$ in the following way:

$$\begin{aligned} & \mathbf{\Phi}_l(\mathbf{\Phi}_l^T\mathbf{\Phi}_l)^{-1}\mathbf{\Phi}_l^T \\ = \; & \mathbf{Q}_l\mathbf{R}_l(\mathbf{R}_l^T\mathbf{Q}_l^T\mathbf{Q}_l\mathbf{R}_l)^{-1}\mathbf{R}_l^T\mathbf{Q}_l^T \\ = \; & \mathbf{Q}_l\mathbf{R}_l(\mathbf{R}_l)^{-1}(\mathbf{R}_l^T)^{-1}\mathbf{R}_l^T\mathbf{Q}_l^T \\ = \; & \mathbf{Q}_l\mathbf{Q}_l^T. \end{aligned} \tag{4.5}$$

It should be noted that we expect the upper triangular matrix, $\mathbf{R}_l$, to be invertible, i.e., all its diagonal elements are nonzero. This is true when all column vectors of $\mathbf{\Phi}_l$ are linearly independent. If the column vectors are linearly dependent, there are redundancy in the representation and one of the dependent vectors should be removed. Thus, $\mathbf{R}_l$ will always be invertible. It should also be noted that if the order of the selected frame vectors is changed, $\mathbf{Q}_l$ and $\mathbf{R}_l$ will be different. Having M vectors, the number of ways to order them is $M!$, and consequently the number of different QR-decompositions is $M!$.

The importance of (4.4) is that in the orthonormal space, the quantization distortion is simply the sum of the coefficient distortions. More specifically, D_l in (4.3), can be written as a constant plus the sum of *independent* coefficient distortions.

4.2 Reformulation of the optimization problem

The original optimization problem in (3.3) is hard to solve since the distortion cannot be written as the sum of coefficient distortions. In the previous section we have shown an orthogonalization procedure, which results in the quantization distortion being the sum of the coefficient distortions in an orthogonal space. The original optimization problem, however, asks for quantization and encoding of the original decision vectors, which are not in this orthogonal space. In this section we reformulate the original optimization problem, in that we quantize and encode the orthogonal coefficients, which has the effect that the distortion and the rate become additive with respect to its orthogonal coefficient rates and distortions. I.e., the dependencies between the orthogonal coefficients are decoupled. We will from now on use the coefficient vector $\tilde{\mathbf{v}}_l^o = [\ \tilde{v}_{l,1}^o \ \ \tilde{v}_{l,2}^o \ \ \cdots \ \ \tilde{v}_{l,M}^o\ ]^T$ as the decision variable. For a given combination of M nonzero coefficients and a given order of selected frame vectors, we know

$\mathbf{\Phi}_l$. We find $\mathbf{Q}_l$ and $\mathbf{R}_l$, the QR-decomposition of $\mathbf{\Phi}_l$. For signal block l the new problem is

$$\begin{aligned} & \|\mathbf{x}_l - \mathbf{Q}_l\mathbf{Q}_l^T\mathbf{x}_l\|^2 \\ + \quad & \lambda\big(\sum_{m=1}^{M}(R_{l,m}^{ind}) + R^{EOB}\big) \\ + \quad & \sum_{m=1}^{M}\min_{\tilde{v}_{l,m}^o}\big((v_{l,m}^o - \tilde{v}_{l,m}^o)^2 + \lambda R_{l,m}^{val}\big), \end{aligned} \tag{4.6}$$

minimized with respect to $\tilde{\mathbf{v}}_l^o$. The first term of (4.6) is the first term of (4.3), found by (4.2) and (4.5). Note that the rates, $R_{l,m}^{ind}$ and $R_{l,m}^{val}$, now depend on the coefficients of the new decision vector, since these are the coefficients encoded and transmitted. The coefficient values are restricted to be one of the J symbols in the *values* codeword table. Due to this fact, the last term of (4.6) can be written as

$$\sum_{m=1}^{M}\min_{j\in\{1,\ldots,J\}}\big((v_{l,m}^o - values(j))^2 + \lambda R_{values(j)}\big), \tag{4.7}$$

where $R_{values(j)}$ is the number of bits of codeword no. j in *values*. From (4.7) it becomes clear that the problem is now much faster to solve, since only JM comparisons are necessary in order to find the minimum solution, compared to the J^M comparisons needed in our original problem. The unquantized coefficient vector, $\mathbf{v}_l^o$, is found by

$$\mathbf{v}_l^o = \mathbf{R}_l\mathbf{v}_l = \mathbf{R}_l(\mathbf{\Phi}_l^T\mathbf{\Phi}_l)^{-1}\mathbf{\Phi}_l^T\mathbf{x}_l = \mathbf{Q}_l^T\mathbf{x}_l. \tag{4.8}$$

To find the optimal rate-distortion tradeoff, we must solve (4.6) for all $\binom{K}{M}M!$ combinations for $M = \{1,\ldots,M_{max}\}$, and store the minimum of all solutions. When working with low bit rate compression, the maximum number of nonzero coefficients per block, M_{max}, is low, resulting in a computationally manageable problem.

It is the orthogonal coefficient vector, $\tilde{\mathbf{v}}_l^o$, and the corresponding indices that are entropy encoded and transmitted through the channel. Therefore, a QR-decomposition is required in the decoder in addition to knowledge of the frame and the VLC tables used in the encoder. For every signal block, l, the set of

selected frame vectors, $\mathbf{\Phi}_l$, is found by decoding the *index* codewords. The matrix $\mathbf{Q}_l$ is found by QR-decomposing $\mathbf{\Phi}_l$, and the reconstructed signal vector, $\tilde{\mathbf{x}}_l$, is found by

$$\tilde{\mathbf{x}}_l = \mathbf{Q}_l \tilde{\mathbf{v}}_l^o. \tag{4.9}$$

A full search through all combinations of up to M_{max} frame vectors (including all vector ordering permutations) is necessary in order to find the optimal solution. Yet, there is a considerable amount of time to save by choosing the right search strategy. This is the topic of Section 4.3.

ORMP, described in Section 2.2.3, follows similar steps to the ones we propose in this chapter. The orthogonalization of frame vectors is essential in both approaches. But, ORMP is clearly based on a heuristic and therefore it does not provide an RD optimal solution. In this work, we choose the orthogonalized vector that is giving us the best RD tradeoff. This is done for *all* combinations of up to M_{max} vectors. If the value of M_{max} is at least as large as the largest number of coefficients needed per signal block, our new approach finds the *rate-distortion optimal solution of the problem that the RDO ORMP algorithm addresses*, and it does so in an efficient way. While the problem in Section 3.2 leads to the RD optimal solution of the problem that the Basic MP algorithm addresses, the solution of the problem in Section 4.2 is the RD optimal solution of the problem that ORMP addresses.

While the *encoding* of the signal is different, the *decoding* is the same, all of the ORMP, RDO ORMP, and ORDE cases. The reconstruction of signal block l is described in the algorithm in Figure 2.4. This is done for the entire set of L signal blocks.

4.3 Time complexity reduction using a solution tree

In this section we organize all $\sum_{M=0}^{M_{max}} \binom{K}{M} M!$ possible ways of combining up to M_{max} frame vectors. We build a solution tree for each signal block, whose purpose is to make it possible to use techniques that give us additional complexity reduction of coding each signal vector optimally.

Consider a tree where each node represents a unique set of selected vectors from the frame $\mathbf{F}$, i.e., a unique $\mathbf{\Phi}_l$. The root node represents the selection of *zero* vectors ($M = 0$). It has K child nodes. Each of these represents the selection of exactly *one* frame vector ($M = 1$). The edges that connect the root node to its child nodes are named “1”, “2”, ... ,“K”, to indicate the index

of the frame vector that is selected. All nodes in level 1 have $K-1$ child nodes each, all nodes in level 2 have $K-2$ child nodes, and so on, down to level M_{max} where the tree is bounded and no child nodes exist. For any node in the tree (except from the nodes at level M_{max}), the branches to the child nodes are named "1", "2", ... ,"K", except from the branch names that lead from the root node down to the current node. This is why the number of children for each node at level M is equal to $K-M$. In other words, a frame vector can never be selected twice. In Figure 4.2 an illustration of this tree is shown. It will have $M_{max}+1$ generations, or levels, included level 0. Level M will have

$$\prod_{m=0}^{M-1}(K-m) = \frac{K!}{(K-M)!} = \binom{K}{M} M! \tag{4.10}$$

number of nodes. The entire tree represents all possible combinations of $0, 1, \ldots, M_{max}$ vectors selected, including all vector ordering permutations. For each node, we find the minimum rate-distortion solution by solving (4.6) for the given combination. To find the globally optimal solution for the signal block, we need to evaluate all nodes in the tree.

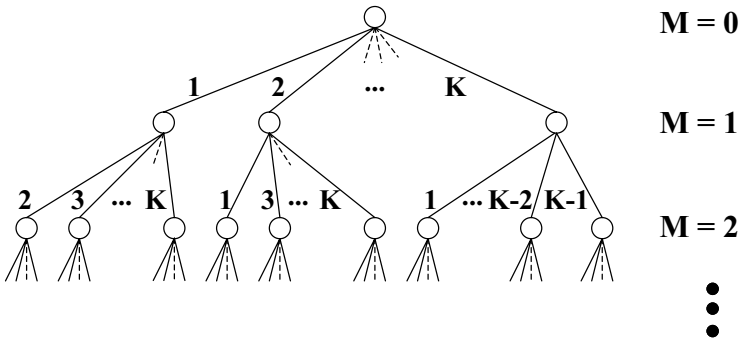


Figure 4.2: Graphical representation of the solution tree. Each node represents the minimum rate-distortion solution given a unique set of selected vectors. The root node has *zero* vectors selected, each node in the 1st generation have *one* vector selected, each node in the 2nd generation have *two* vectors selected, and so on. The entire tree represents all possible combinations of selecting up to M_{max} of K frame vectors.

4.3.1 Time complexity reduction by depth-first-search

By choosing *depth-first-search* (DFS) as the search strategy in this tree, we can build the QR-decomposition for each node recursively. There are two benefits by using DFS: There are fewer computations in order to find the QR-decomposition, and only one coefficient has to be found in order to find the best set of coefficients for the respective node.

DFS starts in the root node, moving on to the first child, then to the first child of the first child, and so on, until a node in level M_{max} is reached. It then backtrack to level $M_{max} - 1$. For any node in the tree, the next step in the DFS strategy is as follows: If there are unvisited children, go to one of them. If not, backtrack to parent node. DFS stops when all nodes in the tree are visited.

Suppose that we know the minimum cost, $C_{l,M-1}$, for a parent node at level $M - 1$,

$$C_{l,M-1} = \|\mathbf{e}_{l,M-1}\|^2 + \alpha_{l,M-1} + \beta_{l,M-1}, \tag{4.11}$$

where

$$\begin{aligned}
\mathbf{e}_{l,M-1} &= \mathbf{x}_l - \mathbf{Q}_{l,M-1}\mathbf{Q}_{l,M-1}^T\mathbf{x}_l, \\
\alpha_{l,M-1} &= \lambda\big(\sum_{m=1}^{M-1}(R_{l,m}^{ind}) + R^{EOB}\big), \\
\beta_{l,M-1} &= \sum_{m=1}^{M-1}\min_{\tilde{v}_{l,m}^o}\big((v_{l,m}^o - \tilde{v}_{l,m}^o)^2 + \lambda R_{l,m}^{val}\big).
\end{aligned}$$

Here $\mathbf{Q}_{l,M-1}$ $(N \times (M-1))$ is the orthonormal basis in the QR-decomposition of $\mathbf{\Phi}_{l,M-1}$, the matrix of selected frame vectors. When going from a parent node to a child node, we use the same set of selected frame vectors, but only adding a new frame vector, $\boldsymbol{\phi}_{l,M}$, as the last column in the set of selected frame vectors, $\mathbf{\Phi}_{l,M}$ $(N \times M)$, that is

$$\mathbf{\Phi}_{l,M} = \begin{bmatrix} \mathbf{\Phi}_{l,M-1} & \boldsymbol{\phi}_{l,M} \end{bmatrix}. \tag{4.12}$$

The new frame vector will always be added to the right end of the matrix. Let $\mathbf{Q}_{l,M}$ denote the orthonormal basis of the QR-decomposition of $\mathbf{\Phi}_{l,M}$.

Since an iteration of the *Gram-Schmidt process* [3] leaves orthonormal vectors computed in previous steps unchanged, $\mathbf{Q}_{l,M}$ can be written

$$\mathbf{Q}_{l,M} = \left[\begin{array}{cc} \mathbf{Q}_{l,M-1} & \mathbf{q}_{l,M} \end{array} \right], \tag{4.13}$$

where

$$\mathbf{q}_{l,M} = \frac{\boldsymbol{\phi}_{l,M} - \sum_{m=1}^{M-1} \langle \boldsymbol{\phi}_{l,M}, \mathbf{q}_{l,m} \rangle \mathbf{q}_{l,m}}{\left\| \boldsymbol{\phi}_{l,M} - \sum_{m=1}^{M-1} \langle \boldsymbol{\phi}_{l,M}, \mathbf{q}_{l,m} \rangle \mathbf{q}_{l,m} \right\|}. \tag{4.14}$$

The Gram-Schmidt process is explained more thoroughly in Appendix A.1. The minimum solution for a particular child node at level M is given by

$$C_{l,M} = \|\mathbf{e}_{l,M}\|^2 + \alpha_{l,M} + \beta_{l,M}, \tag{4.15}$$

According to (4.13), the difference between $\mathbf{x}_l$ and its projection on $span(\boldsymbol{\Phi}_{l,M})$ can be written as

$$\begin{aligned} \mathbf{e}_{l,M} &= \mathbf{x}_l - \mathbf{Q}_{l,M}\mathbf{Q}_{l,M}^T\mathbf{x}_l \\ &= \mathbf{x}_l - \mathbf{Q}_{l,M-1}\mathbf{Q}_{l,M-1}^T\mathbf{x}_l - \mathbf{q}_{l,M}\mathbf{q}_{l,M}^T\mathbf{x}_l \\ &= \mathbf{e}_{l,M-1} - \mathbf{q}_{l,M}\mathbf{q}_{l,M}^T\mathbf{x}_l. \end{aligned} \tag{4.16}$$

Going from a parent node in level $M-1$ to the child node in level M, the first $M-1$ vectors are not changed, thus the first $M-1$ *index* codewords are the same. A new *index* codeword is added, and $\alpha_{l,M}$ is

$$\begin{aligned} \alpha_{l,M} &= \lambda\Big(\sum_{m=1}^{M} (R_{l,m}^{ind}) + R^{EOB} \Big) \\ &= \lambda\Big(\sum_{m=1}^{M-1} (R_{l,m}^{ind}) + R^{EOB} \Big) + \lambda R_{l,M}^{ind} \\ &= \alpha_{l,M-1} + \lambda R_{l,M}^{ind}. \end{aligned} \tag{4.17}$$

Each element in the coefficient vector $\tilde{\mathbf{v}}_l^o$ is found individually according to (4.7). Due to (4.8) the first $M-1$ unquantized coefficients, $[\, v_{l,1}^o \;\; \cdots \;\; v_{l,M-1}^o \,]^T$, will be the same because the first $M-1$ vectors in $\mathbf{Q}_{l,M}$ are not changing.

The quantized coefficients, $[\ \tilde{v}^o_{l,1}\ \ \cdots\ \ \tilde{v}^o_{l,M-1}\ ]^T$, will be the same, and only the last added coefficient, $\tilde{v}^o_{l,M}$, needs to be found. Thus, $\beta_{l,M}$ can be written as

$$\begin{aligned}\beta_{l,M} &= \sum_{m=1}^{M} \min_{\tilde{v}^o_{l,m}} \left((v^o_{l,m} - \tilde{v}^o_{l,m})^2 + \lambda R^{val}_{l,m}\right) \\ &= \sum_{m=1}^{M-1} \min_{\tilde{v}^o_{l,m}} \left((v^o_{l,m} - \tilde{v}^o_{l,m})^2 + \lambda R^{val}_{l,m}\right) \\ &\quad + \min_{\tilde{v}^o_{l,M}} \left((v^o_{l,M} - \tilde{v}^o_{l,M})^2 + \lambda R^{val}_{l,M}\right) \\ &= \beta_{l,M-1} + \min_{\tilde{v}^o_{l,M}} \left((v^o_{l,M} - \tilde{v}^o_{l,M})^2 + \lambda R^{val}_{l,M}\right),\end{aligned} \tag{4.18}$$

or, by using (4.7) for $m = M$:

$$\beta_{l,M} = \beta_{l,M-1} + \min_{j\in\{1,\ldots,J\}} \left((v^o_{l,M} - values(j))^2 + \lambda R_{values(j)}\right). \tag{4.19}$$

To find $\mathbf{q}_{l,M}$ in (4.14) we need $2NM+1$ multiplications and $NM-1$ additions when using Gram-Schmidt. If we would need to use Gram-Schmidt to find the entire $\mathbf{Q}_{l,M}$, the number of multiplications and additions would be equal to $\sum_{m=1}^{M} 2Nm + 1$ and $\sum_{m=1}^{M} Nm - 1$, respectively. In all practical cases, $N \gg 1$, thus the number of multiplications and additions is approximately equal to $2NM(M+1)/2$ and $NM(M+1)/2$, respectively. Computing only $\mathbf{q}_{l,M}$ instead of the entire $\mathbf{Q}_{l,M}$ will result in a reduction in the number of multiplications and additions by a factor equal to $(M+1)/2$. The complexity reduction factor is larger for larger M, i.e., the benefits of using the DFS strategy are greater for larger values of M_{max}.

Due to the knowledge of the parent node's optimal solution, the number of comparisons needed to find the optimal solution of the child node at level M is reduced. The first $M-1$ elements in the coefficient vector is the same as the parent's coefficients. Only the last coefficient must be found. In (4.18) the number of comparisons is reduced from JM to J.

4.3.2 Further time reduction by pruning and lower bounds

So far, full search has been claimed to be the only way to find the optimal solution of the presented problem in this chapter. In fact, there is a situation where pruning can be done and the problem still being solved optimally. Consider the case where the algorithm is working on a node at level M in the tree, where $M < M_{max}$. Going to a child node at level $M+1$ would increase the number of nonzero coefficients by one, as one more vector is added to the set of selected frame vectors. The bit rate will increase with minimum $(R_{min}^{val} + R_{min}^{ind})$ bits, where R_{min}^{val} and R_{min}^{ind} are the minimum number of bits for a *value* and an *index* codeword from the VLC tables, respectively. The projection error of the child node, $\|\mathbf{e}_{l,M+1}\|^2$, will always be less than or equal to the projection error of the parent node, $\|\mathbf{e}_{l,M}\|^2$. In (4.15) this is the only term that can cause a decrease in the cost function when adding a vector, but the projection error will obviously never be negative. Thus, if

$$\|\mathbf{e}_{l,M}\|^2 < \lambda(R_{min}^{val} + R_{min}^{ind}), \tag{4.20}$$

this node is not representing the optimal solution, since the minimum *increase* by adding a vector in the set of selected vectors is larger than the potential *decrease* in the cost function. This branch of the tree can be *pruned*, since this node's children never will hold the optimal solution. Equation (4.20) is of great value in the order of time saving: The full search solution can be found without a full search, which is called implicit enumeration [59] in the integer programming terminology.

Further time savings can be achieved if particular nodes can be eliminated as an optimal candidate, without a full calculation of this node. Let C_l^* be the *so far* minimum cost of block l. As an initial value, C_l^* could be set to the root node cost, $C_{l,0} = \|\mathbf{x}_l\|^2 + \lambda R^{EOB}$. An alternative is to use a fast heuristic to find a suboptimal solution. C_l^* is updated every time a node's solution is better than C_l^*. For a particular node at level M, we define the *lower bound* (LB) for the minimum node cost as

$$LB = \|\mathbf{e}_{l,M}\|^2 + \alpha_{l,M} + \lambda M R_{min}^{val}. \tag{4.21}$$

LB represents the theoretical minimum cost, when the set of selected vectors is known. If $C_l^* < LB$ for a particular node, we know that the global optimal solution is not represented by the particular set of frame vectors, and further computation in order to find $\beta_{l,M}$ is not necessary. The DFS algorithm proceeds to the next node, e.g., a child node. In the worst case $C_l^* \geq LB$ in

all nodes, and a full computation is necessary. But, in an average case many nodes do not need full computation, and the time saved by the introduction of lower bounds is significant.

The pseudo code of the operational rate-distortion encoder (ORDE) algorithm is presented in Figure 4.3. It can be divided into three major parts. The initialization in line 3 to 8, the graph traversal in line 9 to 32, and the particular node cost calculation in line 14. For the sake of readability, this part is fully shown in Figure 4.4. As input to the ORDE algorithm, the signal, $\mathbf{x}$, is reshaped to be a matrix, $\mathbf{X}$, with dimensions $N \times L$, where column vector l is signal block l, $\mathbf{x}_l$. In addition, the frame $\mathbf{F}$, the VLC tables, λ, and the maximum number of selected vectors per block, M_{max}, are given. The outputs of the algorithm are the set of optimal quantized coefficient values, $(\tilde{\mathbf{V}}^o)^*$, and their corresponding set of indices, $(\mathbf{I})^*$. Both sets are matrices with dimensions $M_{max} \times L$, where column vector l is the set of optimal coefficient values, $(\tilde{\mathbf{v}}_l^o)^*$, and their corresponding set of indices, $(\mathbf{i}_l)^*$, respectively.[1]

The graph traversal method in this algorithm is depth-first-search (DFS). Very often DFS is presented recursively. Here, it is presented iteratively, or nonrecursively. The reason for this is to make the implementation of this algorithm more effective. In any program, a function's local data are stored on the program stack when another function is called. A recursive function may call itself many times, resulting in some time consumption when maintaining the stack [30]. With a nonrecursive representation these function calls can be avoided and unnecessary time consumption is saved.

In line 15 the conditional statement make the algorithmic traversal to a child node, as long as the current node is not at level M_{max}, *and* the pruning criterion in (4.20) is not met. Else, the algorithm traverses to the sibling node "next right". By "next right", we mean the node representing the set of selected vectors where the last added vector is replaced by the not selected vector having the lowest index value *higher* than the current one. If there are no sibling nodes next right, the algorithm backtracks (in line 25 to 31).

In Figure 4.4, the node cost is calculated. The so far best set of coefficient values and indices, $(\tilde{\mathbf{v}}_l^o)^*$ and $(\mathbf{i}_l)^*$, are both inputs and outputs. The outputs will be different from the inputs only if the minimum cost of the current node, $C_{l,M}$, is smaller than the so far minimum cost of signal block l, C_l^*. See line 14 to 18. A full calculation is only performed if the lower bound (LB) of this node represents a smaller value than the so far minimum cost for this signal block, C_l^*. In other words, $\beta_{l,M}$ is not calculated for all nodes. This implies

[1] If the rate-distortion optimal solution of block l has less than M_{max} coefficients, the last elements of $(\tilde{\mathbf{v}}_l^o)^*$ and $(\mathbf{i}_l)^*$ are put to 0, respectively.

Algorithm Operational Rate-Distortion Encoder (ORDE)

input : $\mathbf{X} = [\ \mathbf{x}_1\ \ \mathbf{x}_2\ \ \cdots\ \ \mathbf{x}_L\ ]$, $\mathbf{F}$, *values*, *indices*, λ, M_{max}
output: $(\tilde{\mathbf{V}}^o)^* = [(\tilde{\mathbf{v}}_1^o)^*\ (\tilde{\mathbf{v}}_2^o)^* \cdots\ (\tilde{\mathbf{v}}_L^o)^*]$, $(\mathbf{I})^* = [(\mathbf{i}_1)^*\ (\mathbf{i}_2)^* \cdots\ (\mathbf{i}_L)^*]$

```
1  // For all signal blocks:
2  for l = 1, ..., L do
3      C_l^* ← C_{l,0} ← ||x_l||^2 + λR^{EOB} // Initialize the so far min. cost.
4      (ṽ_l^o)^* ← ∅ // The so far best set of nonzero coefficient values.
5      (i_l)^* ← ∅ // The so far best set of nonzero coefficient indices.
6      e_{l,0} ← x_l, α_{l,0} ← 0, β_{l,0} ← 0
7      M ← 1 // Number of selected vectors initially put to 1.
8      i_l ← [ 1 ]^T
9      // Depth-first-search in full tree:
10     while i_{l,1} ≤ K do
11         // Still unmarked nodes in the tree.
12         if i_{l,M} ≤ K then
13             // Find the M^{th} quantized coefficient, ṽ^o_{l,M}, if LB ≤ C_l^*:
14             Algorithm Node cost calculation in Figure 4.4.
15             if M < M_{max} and ||e_{l,M}||^2 ≥ λ(R^{val}_{min} + R^{ind}_{min}) then
16                 // Go to first child node:
17                 i_{new} ← lowest index of not selected frame vectors.
18                 i_l ← [ i_l^T  i_{new} ]^T
19                 M ← M + 1
20             else
21                 β_{l,M} ← ∅
22                 // Go to the sibling node "next right":
23                 i_{l,M} ← lowest index value higher than i_{l,M}, from the
                   set of not selected frame vectors.
24             end
25         else
26             // i_{l,M} > K. Backtrack.
27             M ← M − 1 // Go to parent node.
28             β_{l,M} ← ∅
29             // Go to the sibling node "next right":
30             i_{l,M} ← lowest index value higher than i_{l,M}, from the set
               of not selected frame vectors.
31         end
32     end
33 end
```

Figure 4.3: Operational Rate-Distortion Encoder (ORDE) algorithm.

that there can be situations where a child node needs full calculation, but where the parent node is not fully calculated. $\beta_{l,M-1}$ is unknown, but has to be found before $\beta_{l,M}$ can be found, according to line 11. Obviously, β_l for all ancestors have to be known in order to find $\beta_{l,M}$ for the current node.

Algorithm Node cost calculation

input : $\mathbf{x}_l$, $\mathbf{F}$, *values*, λ, C_l^*, $(\tilde{\mathbf{v}}_l^o)^*$, $(\mathbf{i}_l)^*$, $\mathbf{i}_l$
output: $(\tilde{\mathbf{v}}_l^o)^*$, $(\mathbf{i}_l)^*$

1 *// Find the M^{th} quantized coefficient, $\tilde{v}_{l,M}^o$, if $LB \leq C_l^*$:*
2 $\boldsymbol{\phi}_{l,M} \leftarrow \mathbf{f}_{i_{l,M}}$
3 $\mathbf{q}_{l,M}$ found by Gram-Schmidt *// Equation (4.14).*
4 $\mathbf{e}_{l,M} \leftarrow \mathbf{e}_{l,M-1} - \mathbf{q}_{l,M}\mathbf{q}_{l,M}^T\mathbf{x}_l$ *// Equation (4.16).*
5 $\alpha_{l,M} \leftarrow \alpha_{l,M-1} + \lambda R_{l,M}^{ind}$ *// Equation (4.17).*
6 $LB \leftarrow \|\mathbf{e}_{l,M}\|^2 + \alpha_{l,M} + \lambda M R_{min}^{val}$ *// Equation (4.21).*
7 **if** $LB \leq C_l^*$ **then**
8 **if** β_l *for all ancestors not found* **then**
9 Find those, such that $\beta_{l,M-1}$ is known
10 **end**
11 $\beta_{l,M} \leftarrow \beta_{l,M-1} + \min_{j\in\{1,\ldots,J\}} \left((v_{l,M}^o - values(j))^2 + \lambda R_{values(j)}\right)$ *// Equation (4.19).*
12 $\tilde{v}_{l,M}^o \leftarrow values(j_M)$ *// j_M is the index of the optimal element of the values table.*
13 $C_{l,M} \leftarrow \|\mathbf{e}_{l,M}\|^2 + \alpha_{l,M} + \beta_{l,M}$ *// Equation (4.15).*
14 **if** $C_{l,M} \leq C_l^*$ **then**
15 $C_l^* \leftarrow C_{l,M}$
16 $(\tilde{\mathbf{v}}_l^o)^* \leftarrow \tilde{\mathbf{v}}_l^o$
17 $(\mathbf{i}_l)^* \leftarrow \mathbf{i}_l$
18 **end**
19 **end**

Figure 4.4: Calculation of minimum cost of a current node.

Let us show an example that illustrates the time savings of using lower bounds and pruning. We use the proposed algorithm to code a Gaussian AR(1) process with $\rho = 0.95$. We use a 16×32 frame trained on another AR(1) signal. The training scheme used is presented in next chapter. The number of different coefficient values is $J = 32$. In Table 4.1 we show the time consumption

M_{max}	2	3	4
Original problem	1	320	74411
Orthogonalized problem	0.127	5.7	223
Orthogonalized problem using DFS	0.064	1.9	56
Orthogonalized problem using DFS, LB and pruning	0.028	1.26	38.2

Table 4.1: Time units relative to encoding one signal block using the original formulation in (4.1) when $M_{max} = 2$. $K = 32$ and $J = 32$ for all cases.

when using 1) the original problem in (4.1), 2) the orthogonalized version in (4.6) using an arbitrary traversing algorithm, 3) the orthogonalized version by DFS strategy, and 4) the orthogonalized version by DFS strategy, lower bounds (LB) and pruning techniques. The experiment is done for $M_{max} = \{2, 3, 4\}$. The time units in the table are relative to coding one signal block using the original formulation when $M_{max} = 2$. The largest time reduction is achieved by the orthogonalization procedure described in Section 4.1 and 4.2, particulary in the case when $M_{max} = 4$. When using the DFS strategy, the time is reduced by an additional factor equal to M_{max}. What is interesting to see is that the time is further reduced by 32 - 56 %, when lower bounds in each node in the tree are used. This is a significant contribution for speeding up the algorithm without loosing optimality.

4.4 Ordered vector selection and run-length coding

In order to get an even more efficient algorithm, we add a constraint to our problem: Frame vectors are selected and QR-decomposed only in the order as they appear in the frame $\mathbf{F}$. By doing this we reduce the size of the search in the problem, since all but one of the $M!$ permutations of a given set of M vectors are eliminated. In addition, it opens up for the use of *run-length coding* (RLC) [27] as a part of the entropy coder. Instead of encoding the coefficient *indices*, the number of zeros between each nonzero coefficient, *runs*, is encoded in RLC. After the last nonzero coefficient in each block, an End Of Block (EOB) symbol is transmitted, as it was done when the *indices* got encoded directly. The *run* for the k^{th} coefficient is the number of zeros *in front* of this coefficient. Consider an example where the solution set of selected frame vector *indices* is $\mathbf{i}_l = [\ 7\ \ 8\ \ 23\ ]^T$. The corresponding set of *runs* would be $[\ 6\ \ 0\ \ 14\ ]^T$; There are 6 zeros in front of the 7^{th} coefficient, 0 zeros between the 7^{th} and the 8^{th} coefficient, and 14 zeros between the 8^{th} and the 23^{rd} coefficient.

In the situations where the number of nonzero coefficients is greater than one, there will be more low valued *runs* than high valued *runs*. If we look at the entire set of signal blocks, the entropy of the run-lengths will be less than the entropy of coefficient indices. This is the same as saying that run-length coding is more bit efficient than coding the coefficient indices directly. The introduction of ordered vector selection results in a reduction of the number of solutions, and we can draw a new *reduced* solution tree, like the one presented in Figure 4.5. This tree is not balanced like the full solution tree in Figure 4.2. The reduced tree is equal to the full tree at level 0 and 1, and the edges that connect the root node to its child nodes are named "1", "2", ... ,"K", indicating the vector selected. The node that represents the selection of frame vector no. 1, $\mathbf{f}_1$, has $K-1$ child nodes. These children represent the selection of *two* vectors ($M=2$), where the first vector of $\mathbf{\Phi}_l$ is $\mathbf{f}_1$ and the second is $\mathbf{f}_i$, where $i = 2, 3, \ldots, K$, respectively. But, the node representing $\mathbf{\Phi}_l = [\mathbf{f}_2]$ has $K-2$ children, the node with $\mathbf{\Phi}_l = [\mathbf{f}_3]$ has $K-3$ children, and so on. Thus, the node with $\mathbf{\Phi}_l = [\mathbf{f}_{K-1}]$ has only one child, and the node representing $\mathbf{\Phi}_l = [\mathbf{f}_K]$ has none. The reduced tree will have M_{max} levels, as the full tree, but the number of nodes is reduced from $\sum_{M=0}^{M_{max}} \binom{K}{M} M!$ to $\sum_{M=0}^{M_{max}} \binom{K}{M}$. There is no longer $M!$, but *one* way the set of M frame vectors can be ordered. This is a considerable reduction in problem size. We do not have the same optimization problem as when coding the *indices*, since the vector selection now is ordered. But, the introduction of RLC can give us a more bit efficient representation and thus a better rate-distortion optimal solution. Experimental results from comparisons between ordered and not ordered vector selection are presented in Chapter 6.2.4 on AR(1) signals, and in Chapter 7.3.3 on ECG signals.

The ORDE algorithm for the reduced tree is a bit different than the ORDE algorithm for the full tree. Line 9 to 32, in Figure 4.3 are replaced by the algorithm described in Figure 4.6. It is still nonrecursive depth-first-search, but the number of nodes is reduced, and the traversing through the tree is simpler. It is easier to find the first child node, $i_{l,M+1} = i_{l,M} + 1$, and to find the sibling node "next right", $i_{l,M} = i_{l,M} + 1$. Another difference is that the *indices* table is replaced by a *runs* VLC table as input to the ORDE algorithm. In line 5 of Figure 4.4, $R_{l,M}^{ind}$ is replaced by $R_{l,M}^{run}$, the number of bits of the added *run* codeword.

4.5 ORDE and complexity

An overview of the number of solutions of the cases described so far is given in Table 4.2. We can see the tremendous reduction in complexity between

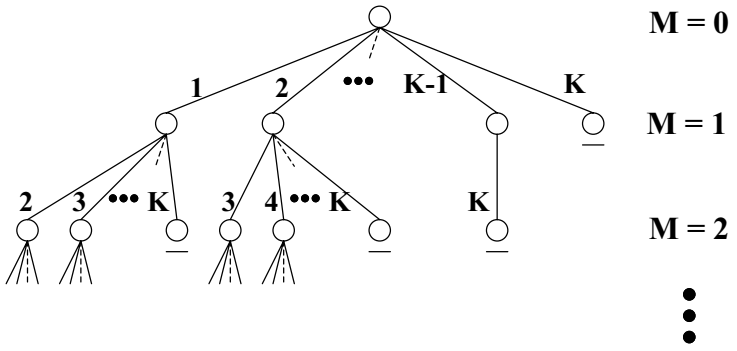


Figure 4.5: The reduced solution tree. Each node represents the minimum rate-distortion solution for a set of selected vectors. The entire tree represents all possible combinations of selecting up to M_{max} of K frame vectors in the same order as they appear in $\mathbf{F}$.

		Complexity
a)	Original nonorthogonalized problem	$1 + \sum_{M=1}^{M_{max}} \binom{K}{M} J^M$
b)	Orthogonalized problem	$1 + \sum_{M=1}^{M_{max}} \binom{K}{M} JMM!$
c)	Orthogonalized problem using DFS	$1 + \sum_{M=1}^{M_{max}} \binom{K}{M} JM!$
d)	Orthogonalized problem using DFS and ordered vector selection	$1 + \sum_{M=1}^{M_{max}} \binom{K}{M} J$

Table 4.2: Number of solutions when encoding one signal block optimally.

the original nonorthogonalized problem and the orthogonalized problem using depth-first-search and ordered vector selection. In addition, we have a complexity reduction caused by the lower bound technique presented in Section 4.3.2. For fixed values of K and J, the complexity reduction is becoming very significant for increasing M_{max}.

So far we have presented a formulation of the optimization problem and an algorithm that finds the optimal solution in an computational efficient way. Now, we will look at the parameters that this optimal solution highly depends on, the frame and the VLC tables. Next chapter is about how to train these parameters.

Algorithm nonrecursive depth-first-search in reduced tree

while $i_{l,1} \leq K$ **do**
  // Still unmarked nodes in the tree.
  if $i_{l,M} \leq K$ **then**
    // Find the M^{th} *quantized coefficient,* $\tilde{v}^o_{l,M}$*, if* $LB \leq C^*_l$*:*
    Algorithm Node cost calculation in Figure 4.4.
    if $M < M_{max}$ **and** $\|\mathbf{e}_{l,M}\|^2 \geq \lambda(R^{val}_{min} + R^{run}_{min})$ **then**
      // Go to first child node:
      $\mathbf{i}_l \leftarrow [\ \mathbf{i}_l^T \ \ (i_{l,M} + 1)\ ]^T$
      $M \leftarrow M + 1$
    else
      $\beta_{l,M} \leftarrow \emptyset$
      // Go to the sibling node "next right":
      $i_{l,M} \leftarrow i_{l,M} + 1$
    end
  else
    // $i_{l,M} > K$*. Backtrack.*
    $M \leftarrow M - 1$ *// Go to parent node.*
    $\beta_{l,M} \leftarrow \emptyset$
    // Go to the sibling node "next right":
    $i_{l,M} \leftarrow i_{l,M} + 1$
  end
end

Figure 4.6: The nonrecursive depth-first-search in reduced tree algorithm.

Chapter 5

Frame design and variable length coder (VLC) optimization

The rate-distortion optimal solution highly depends on the frame and the VLC tables used in the frame based coding scheme. In this chapter, we describe how we train the VLC tables and the frame for the particular class of signals that we are going to compress, in order to get an improved coding result. The training scheme is presented in Section 5.2, and it consists of two loops. In the first loop, a frame design algorithm from previous work, called Method of Optimal Directions (MOD) is used. The MOD algorithm is presented in Section 5.1. The first loop is fully described in Section 5.2.1. In this loop a coarse update of the frame is made. The second loop is presented in Section 5.2.2. A fine update of the frame is made by a suboptimal, but convergent algorithm, in parallel to an optimal update of the VLC tables. The trained frame and the updated VLC tables are ready to be used in an ORDE scheme for the same class of signals.

5.1 The MOD algorithm

When using frames in a compression scheme, it is important that the frame is well designed in order to get a sparse representation with a good quality of the reconstructed signal. The design flexibility is one of the benefits of using frames instead of orthogonal transforms. In [14] the *method of optimal directions* (MOD) is introduced. This is an iterative frame training algorithm

for use with vector selection algorithms, like Matching Pursuit (MP). MOD is inspired by the General Lloyd Algorithm (GLA), which is used for optimizing Vector Quantization (VQ) codebooks [16]. GLA will always find a better codebook for each iteration and converge to a local minimum. This is not the case for MOD, if the vector selection algorithm is MP, or another suboptimal method.

Let the signal vectors be organized as the $N \times L$ matrix $\mathbf{X} = [\ \mathbf{x}_1\ \ \mathbf{x}_2\ \ \cdots\ \ \mathbf{x}_L\ ]$, and let the corresponding coefficient vectors be the $K \times L$ matrix $\mathbf{W} = [\ \mathbf{w}_1\ \ \mathbf{w}_2\ \ \cdots\ \ \mathbf{w}_L\ ]$. Each iteration of MOD can be divided into two steps:

1. $\mathbf{F}$ and $\mathbf{X}$ are known. Find $\mathbf{W}$ by using a vector selection algorithm.

2. $\mathbf{W}$ and $\mathbf{X}$ are known. Find a better $\mathbf{F}$.

The vector selection algorithm used in step 1 is Matching Pursuit or another greedy algorithm. In step 2 the frame is optimally updated. The minimization problem is formulated as

$$\min_{\mathbf{F}} \|\mathbf{X} - \mathbf{F}\mathbf{W}\|^2. \tag{5.1}$$

This is a Least Squares problem [3], and a unique solution exists if $\mathbf{W}$ has full rank. The new frame is found by

$$\mathbf{F} = \mathbf{X}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}. \tag{5.2}$$

In addition, the new frame vectors are normalized. The two steps will continue iterating until a stop criterion is met. Several criteria can be used; for example maximum number of of iterations, or almost constant distortion from one iteration to another. If both step 1 and step 2 in the algorithm were optimally solved, the algorithm would be guaranteed to converge. This is not the situation here, since step 1 is not optimal. Note that the coefficient matrix with unquantized elements, $\mathbf{W}$, is used. Quantization is not a part of MOD, nor any QR-decomposition.

5.2 The training scheme

The training scheme of this work consists of two iterative loops. An illustration is given in Figure 5.1. A central part of both loops is the operational rate-distortion encoder (ORDE) algorithm. Whether the algorithm of Figure 4.3 with *index* coding, or the algorithm of Figure 4.6 with run-length coding is used, the training scheme works exactly the same. Figure 5.1 shows the run-length coding case. For *index* coding, the *runs* VLC table is replaced by the *indices* VLC table. The purpose of the first loop, Loop 1, is to get a coarse update of the frame. In this loop, the frame vectors can change order for each iteration, thus an update of the *runs* VLC table would be meaningless. This is why the update of VLC tables is placed in the second loop, Loop 2.

5.2.1 Loop 1: Coarse frame design in an RDO coding scheme

In Loop 1, we use the same technique as MOD to find the new frame, but on the set of quantized coefficients, $\tilde{\mathbf{W}} = [\ \tilde{\mathbf{w}}_1\ \ \tilde{\mathbf{w}}_2\ \ \cdots\ \ \tilde{\mathbf{w}}_L\ ]$, instead of $\mathbf{W}$, thus

$$\mathbf{F} = \mathbf{X}_{tr}\tilde{\mathbf{W}}^T(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T)^{-1}. \tag{5.3}$$

$\mathbf{X}_{tr}$ is the training signal organized as an $N \times L$ matrix. The output from ORDE is the set of *orthogonalized* and quantized coefficient values, $(\tilde{\mathbf{V}}^o)^*$, and indices, $(\mathbf{I})^*$. To generate $\tilde{\mathbf{W}}$ from $(\tilde{\mathbf{V}}^o)^*$ and $(\mathbf{I})^*$, the QR-decomposition of each set of selected vectors from $\mathbf{F}$, $\mathbf{\Phi}_l$, has to be found. This is a straight forward operation when using ordered vector selection, since we know the combination of the selected vectors from $(\mathbf{I})^*$:

$$\mathbf{\Phi}_l = [\ \mathbf{f}_{(i_{l,1})^*}\ \ \mathbf{f}_{(i_{l,2})^*}\ \ \cdots\ ]. \tag{5.4}$$

By Gram-Schmidt, the QR-decomposition of $\mathbf{\Phi}_l$, $\mathbf{Q}_l$ and $\mathbf{R}_l$, are found. Column vector l in $\tilde{\mathbf{W}}$, $\tilde{\mathbf{w}}_l$, is found by

$$\tilde{\mathbf{w}}_l = \mathbf{R}_l^{-1}\tilde{\mathbf{w}}_l^o. \tag{5.5}$$

$\tilde{\mathbf{w}}_l^o$ is a vector of K elements, where the only nonzero elements are the set with index values as indicated in $(\mathbf{i}_l)^*$. These elements' values are listed in $(\tilde{\mathbf{v}}_l^o)^*$. In (5.3), the matrix $(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T)$ has to be full rank in order to be invertible. In situations where a particular frame vector, $\mathbf{f}_k$, has not been selected for any of the L signal blocks, $(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T)$ will not be full rank. This problem is solved

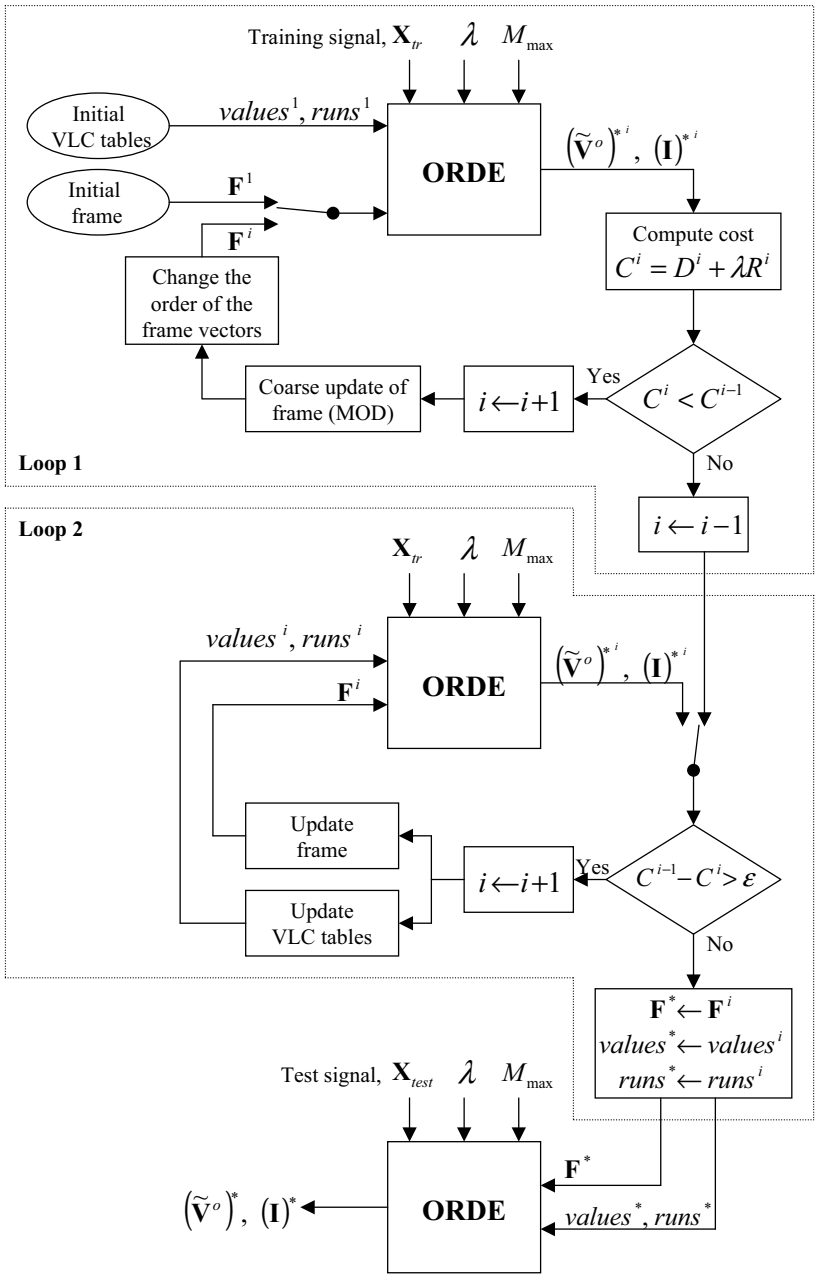


Figure 5.1: Operational Rate-Distortion Encoder (ORDE) and training loops for the run-length coding case. The scheme is the same for *index* coding, by replacing "*runs*" with "*indices*".

by removing frame vector $\mathbf{f}_k$ from $\mathbf{F}$. Let the remaining frame be named $\mathbf{F}'$. Correspondingly, row k in $\tilde{\mathbf{W}}$ is removed and the remaining $(K-1)\times L$ matrix is called $\tilde{\mathbf{W}}'$. Then we solve

$$\mathbf{F}' = \mathbf{X}_{tr}\tilde{\mathbf{W}}'^{T}(\tilde{\mathbf{W}}'\tilde{\mathbf{W}}'^{T})^{-1}. \tag{5.6}$$

When $\mathbf{F}'$ is found, a new frame vector is added in order to get a frame with dimensions $N \times K$. Instead of choosing a random vector, the signal vector with corresponding coefficient vector with the highest number of nonzero coefficients is chosen. We call this vector $\mathbf{x}_s$, and the corresponding coefficient vector $\tilde{\mathbf{w}}_s$. If there are more than one coefficient vector in this category, $\tilde{\mathbf{w}}_s$ is the vector that in addition has the highest bit rate. By adding $\mathbf{x}_s$ to the frame, we will reduce the bit rate in the next iteration for signal block s, since only one frame vector will be chosen to represent $\mathbf{x}_s$. All the column vectors in the updated frame are normalized, in order to get frame vectors of unit length.

The last part of the frame updating in Loop 1 is a reorganization of the frame vectors. The most frequently chosen vector from ORDE is the first vector in the new frame. The rest of the vectors is placed in descending frequency order. This is done in order to get a higher density of low valued *runs*, and thereby an overall lower bit rate, after the update of the VLC tables in Loop 2. Due to the reorganization, the *runs* between nonzero coefficients will change dramatically from one iteration to another. Thus, an update of the VLC tables is not taking place in Loop 1. This reorganization has no meaning when coding *indices* instead of *runs*. Even though (5.2) is optimal in the distortion sense [14], (5.3) is not optimal in this work. This is because $\tilde{\mathbf{W}}$ is a function of $\mathbf{F}$: In (5.5), for each column of $\tilde{\mathbf{W}}$, the QR-decomposition of a subset of $\mathbf{F}$ has to be found. Or in other words, $\mathbf{F}$ is on both sides of the equality sign of (5.3). A change in $\mathbf{F}$ would change column vectors in $\tilde{\mathbf{W}}$, resulting in another design of $\mathbf{F}$.

The algorithm will iterate as long as the computed cost for iteration no. i, $C^i = D^i + \lambda R^i$, is less than C^{i-1}, the cost in the previous iteration. Loop 1 is not guaranteed to converge, nor in distortion or rate-distortion sense. However, experimental results in the next chapter show that this scheme works well and generates frames that are well suited for the given class of input data. The loop terminates at iteration i' where $C^{i'} \geq C^{i'-1}$. The cost, $C^{i'}$, the frame, $\mathbf{F}^{i'}$, and the set of coefficient values, $(\tilde{\mathbf{V}}^o)^{*^{i'}}$, and indices, $(\mathbf{I})^{*^{i'}}$, are set equal to $C^{i'-1}$, $\mathbf{F}^{i'-1}$, $(\tilde{\mathbf{V}}^o)^{*^{i'-1}}$, and $(\mathbf{I})^{*^{i'-1}}$, respectively. i' is set to $i'-1$. The algorithm proceeds to Loop 2. Loop 1 is summarized in the pseudo code in Figure 5.2.

Algorithm Training Loop 1

input : $\mathbf{X}_{tr}$, $\mathbf{F}^1$, $values^1$, $runs^1$, λ, M_{max}

output: $C^{i'}$, $\mathbf{F}^{i'}$, $(\tilde{\mathbf{V}}^o)^{*^{i'}}$, $(\mathbf{I})^{*^{i'}}$

$[(\tilde{\mathbf{V}}^o)^{*^1}, (\mathbf{I})^{*^1}] \leftarrow \text{ORDE}(\mathbf{X}_{tr}, \mathbf{F}^1, values^1, runs^1, \lambda, M_{max})$
$C^1 \leftarrow D^1 + \lambda R^1$
$C^0 \leftarrow 2C^1$ *// Just to make the while loop to start.*
$i \leftarrow 1$
while $C^i < C^{i-1}$ **do**
    $i \leftarrow i + 1$
    for $l = 1, \ldots, L$ **do**
        $\tilde{\mathbf{w}}_l^o \leftarrow \mathbf{0}$
        $M \leftarrow \min\{M_{max},$ index of last nonzero element of $(\tilde{\mathbf{v}}_l^o)^{*^i}\}$
        for $m = 1$ **to** M **do**
            $\tilde{w}^o_{l,(i_{l,m})^{*^i}} \leftarrow (\tilde{v}^o_{l,m})^{*^i}$
        end
        $\tilde{\mathbf{w}}_l \leftarrow \mathbf{R}_l^{-1}\tilde{\mathbf{w}}_l^o$ *// Equation (5.5).*
    end
    $\mathbf{F}^i \leftarrow \mathbf{X}_{tr}\tilde{\mathbf{W}}^T(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T)^{-1}$ *// Equation (5.3).*
    for $k = 1, \ldots, K$ **do**
        $\mathbf{f}_k^i \leftarrow \frac{\mathbf{f}_k^i}{\|\mathbf{f}_k^i\|}$ *// Normalization of frame vector k.*
    end
    Sort the frame vectors in decreasing selected frequency order.
    $[(\tilde{\mathbf{V}}^o)^{*^i}, (\mathbf{I})^{*^i}] \leftarrow \text{ORDE}(\mathbf{X}_{tr}, \mathbf{F}^i, values^1, runs^1, \lambda, M_{max})$
    $C^i \leftarrow D^i + \lambda R^i$
end
$C^{i'} \leftarrow C^{i-1}$
$\mathbf{F}^{i'} \leftarrow \mathbf{F}^{i-1}$
$(\tilde{\mathbf{V}}^o)^{*^{i'}} \leftarrow (\tilde{\mathbf{V}}^o)^{*^{i-1}}$
$(\mathbf{I})^{*^{i'}} \leftarrow (\mathbf{I})^{*^{i-1}}$
$i' \leftarrow i - 1$

Figure 5.2: Training Loop 1.

5.2.2 Loop 2: Convergent frame design and VLC optimization

Loop 2 is guaranteed to converge to a lower or equal cost, $C^i \leq C^{i-1}$, for each iteration. As in Loop 1, the vector selection is done by ORDE, which always will give us the minimum cost. The updates of the VLC tables and the frame are done in parallel, since they are separable. That is, a change in the VLC tables would only affect the bit rate, and the update of $\mathbf{F}$ will, in this loop, only affect the distortion. Updating of the VLC tables is based on the symbol frequency distributions for the encoded versions of $(\tilde{\mathbf{V}}^o)^*$ and $(\mathbf{I})^*$. If run-length coding is used, it is necessary to find the set of *runs* between the index values in the columns of $(\mathbf{I})^*$. Based on the symbol frequencies in this set, the *runs* VLC table is updated. Huffman coding [16] is used to find the new set of symbol codewords, both for the *runs* and the *values* VLC table. The reason for using Huffman coding is that we focus on *operational* RDO coding, i.e., practical optimal coding where all parameters are known prior to the encoding. In arithmetic coding, for example, the bit stream will be made after all codewords are known, based on the knowledge of each codewords entropy. After the VLC tables updates, the bit rate will always be less or equal than with the previous VLC tables.

The update of the frame, $\mathbf{F}$, in Loop 2 will not affect the rate, but only the distortion. Thus, the bit rate will after several iterations converge to a local optimum [32]. The update of $\mathbf{F}$ is a simple heuristic, where a change in it is made only if it results in a smaller distortion. The overall distortion, D, is given by

$$D = \sum_{l=1}^{L} \|\mathbf{x}_l - \mathbf{Q}_l \tilde{\mathbf{v}}_l^o\|^2, \tag{5.7}$$

where $\mathbf{x}_l$ and $\tilde{\mathbf{v}}_l^o$ are fixed parameters in this part of the training loop. For a single frame element, f_{nk}, we add a small real valued constant, δ, and normalize the entire frame vector, and call it $\mathbf{f}_k^+$. Let us call the new frame $\mathbf{F}^+$, the QR-decomposition of its subset in block l $\mathbf{Q}_l^+$, and the new overall distortion D^+. D^+ is found by

$$D^+ = \sum_{l=1}^{L} \|\mathbf{x}_l - \mathbf{Q}_l^+ (\tilde{\mathbf{v}}_l^o)^*\|^2. \tag{5.8}$$

If $D^+ > D$, we turn δ negative and repeat the procedure described above. If $D^+ < D$, the frame $\mathbf{F}$ is set equal to $\mathbf{F}^+$, and the algorithm continues adding

δ to f_{nk}, normalizing $\mathbf{f}_k$, and so on. The iterations stop when $D^+ \geq D$. This is done for every single frame element, f_{nk}, $n = 1, \ldots, N$ and $k = 1, \ldots, K$ in sequential order.

The new frame and VLC tables are used as inputs in the next iteration of Loop 2. The iterations terminate when the cost reduction from one iteration to next is less than a predefined small number, ε. The training procedure is complete. The last updated frame and VLC tables are used as parameters in the encoding of the test signal, $\mathbf{X}_{test}$, in addition to the same λ and M_{max} values that are used in the training scheme. Loop 2 is summarized in the pseudo code in Figure 5.3 and Figure 5.4.

Algorithm Training Loop 2

input : $\mathbf{X}_{tr}$, $\mathbf{F}^{i'}$, $values^1$, $runs^1$, λ, M_{max}, $C^{i'}$, $(\tilde{\mathbf{V}}^o)^{*^{i'}}$, $(\mathbf{I})^{*^{i'}}$
output: $\mathbf{F}^*$, $values^*$, $runs^*$

$\varepsilon \leftarrow$ small positive number
$i \leftarrow i'$
while $C^{i-1} - C^i > \varepsilon$ **or** $i = i'$ **do**
    $i \leftarrow i + 1$
    // Update the VLC tables:
    Find the frequencies of *value* symbols in $(\tilde{\mathbf{V}}^o)^{*^{i-1}}$
    Find the *runs* between the index values in the columns of $(\mathbf{I})^{*^{i-1}}$
    Find the symbol frequencies of this set of *run* symbols.
    Based on the symbol frequencies, find the new Huffman table for the *value* and *run* symbols. New VLC tables: $values^i$ and $runs^i$.
    // Update the frame:
    Algorithm Convergent frame update in Figure 5.4.
    $[(\tilde{\mathbf{V}}^o)^{*^i}, (\mathbf{I})^{*^i}] \leftarrow \mathrm{ORDE}(\mathbf{X}_{tr}, \mathbf{F}^i, values^i, runs^i, \lambda, M_{max})$
    $C^i \leftarrow D^i + \lambda R^i$
end
$\mathbf{F}^* \leftarrow \mathbf{F}^i$, $values^* \leftarrow values^i$, $runs^* \leftarrow runs^i$

Figure 5.3: Training Loop 2.

Algorithm Convergent frame update

input : $\mathbf{X}_{tr}$, $\mathbf{F}^{i-1}$, D^{i-1}, $(\tilde{\mathbf{V}}^o)^{*^{i-1}}$, $(\mathbf{I})^{*^{i-1}}$
output: $\mathbf{F}^i$

1 $\mathbf{F}^i \leftarrow \mathbf{F}^{i-1}$, $D^i \leftarrow D^{i-1}$
2 **for** $k = 1, \ldots, K$ **do**
3 **for** $n = 1, \ldots, N$ **do**
4 $\delta \leftarrow$ small positive number
5 **do**
6 $\mathbf{f}_k^+ \leftarrow [\ f_{1k}^i \ \ f_{2k}^i \ \ \cdots \ \ (f_{nk}^i + \delta) \ \ \cdots \ \ f_{Nk}^i\ ]^T$
7 $\mathbf{f}_k^+ \leftarrow \frac{\mathbf{f}_k^+}{\|\mathbf{f}_k^+\|}$ *// Normalization of frame vector k.*
8 $\mathbf{F}^+ \leftarrow [\ \mathbf{f}_1^i \ \ \mathbf{f}_2^i \ \ \cdots \ \ \mathbf{f}_k^+ \ \ \cdots \ \ \mathbf{f}_K^i\ ]$
9 Find $\mathbf{Q}_l^+$ for every signal block, where $\mathbf{F}^+$ is frame.
10 $D^+ \leftarrow \sum_l \|\mathbf{x}_l - \mathbf{Q}_l^+ (\tilde{\mathbf{v}}_l^o)^{*^{i-1}}\|^2$
11 **if** $D^+ > D^i$ **then**
12 $\delta \leftarrow -\delta$
13 **end**
14 **while** ($D^+ > D^i$ **and** $\delta < 0$) **end**
15 **while** $D^+ < D^i$ **do**
16 $\mathbf{F}^i \leftarrow \mathbf{F}^+$, $D^i \leftarrow D^+$
17 Repeat line 6 to 10.
18 **end**
19 **end**
20 **end**

Figure 5.4: Convergent frame design algorithm.

Chapter 6

Compression of AR(1) signals

In this chapter we show some experiments done to illustrate the capability of the Operational Rate-Distortion Encoder (ORDE) on AR(1) processes. In Section 6.1 the theoretical rate-distortion function for an AR(1) process is presented. One of the experiments in Section 6.2 compares the obtained results to this lower bound, in order to illustrate how close the practical experiments using the ORDE scheme are to the theoretical bound. Section 6.3 summarizes the results from the 5 experiments presented in Section 6.2.

6.1 The theoretical Rate-Distortion Function

The average information content, or *entropy*, of a continuous-amplitude source is infinite. Each sample from such a source can have an infinite number of possible values. Furthermore, the bit rate needed for an exact reproduction, R, is infinitive. No practical coding procedure can reproduce a continuous waveform with perfect fidelity. But, given an upper bound of R, it is possible to find the theoretical lower bound of the distortion, D, for some known classes of signals. Gaussian sources are in this group of signals. The rate-distortion function, $D(R)$, is given by the following parametric equations [27]:

$$D(\phi) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \min\{\phi, S_{xx}(e^{j\omega})\} d\omega \tag{6.1}$$

and

$$R(\phi) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \max\left\{0, \frac{1}{2} \log_2 \frac{S_{xx}(e^{j\omega})}{\phi}\right\} d\omega, \tag{6.2}$$

where $S_{xx}(e^{j\omega})$ is the power density spectrum of the Gaussian signal, and $\phi \in \mathbb{R}^+$ is the parametric variable. $D(\phi)$ and $R(\phi)$ only depend on $S_{xx}(e^{j\omega})$. If the Gaussian source is memoryless, the signal is white noise and $S_{xx}(e^{j\omega})$ is a constant. To be more precise, $S_{xx}(e^{j\omega}) = \sigma_x^2$, where σ_x^2 is the variance of the source signal. If the Gaussian source has memory, signal samples are correlated, and the source entropy will be less than the entropy for a memoryless source. In other words, the distortion will be less at the same bit rate, resulting in a better rate-distortion performance.

An AR(1) signal is a first-order Markov process [41] with a Gaussian source, or in other words, a signal from a Gaussian source with memory. The power density spectrum is no more a constant. For an AR(1) process,

$$S_{xx}(e^{j\omega}) = \frac{1-\rho^2}{1+\rho^2-2\rho\cos(\omega)}\sigma_x^2, \tag{6.3}$$

where $\rho \in [0, 1\rangle$ is the adjacent-sample correlation coefficient. If $\rho = 0$, there is no correlation between adjacent samples.

The *small distortion region* [27] is defined by

$$D \leq \frac{1-\rho}{1+\rho}\sigma_x^2. \tag{6.4}$$

For this region the rate-distortion function can be found in a simpler way than by using the parametric equations in (6.1) and (6.2). $D(R)$ is given by

$$D(R) = (1-\rho)^2 2^{-2R}\sigma_x^2. \tag{6.5}$$

In this work $\rho = 0.95$, i.e., the small distortion region is defined by $D \leq 0.0256\sigma_x^2$, due to (6.4). In Figure 6.9 in Experiment no. 2, the dotted line represents the theoretical RD function for an AR(1) process with $\rho = 0.95$. The part of the curve where $D/\sigma_x^2 > 0.0256$ is found by solving (6.1) and (6.2) with discrete values of ϕ in the range 0.0256 and up. It is not possible to reach the theoretical RD function in a practical coding scheme. Yet, a comparison between an encoder's operational RD function and the theoretical RD function is useful in order to get a measure of the encoder's RD performance.

6.2 Experiments on AR(1) signals

In all experiments in this section, two different AR(1) processes with $\rho = 0.95$ are used as input *training* and input *test* signals. For an AR(1) process with $\rho = 0.95$, the DCT transform is almost equal to the KLT transform [27], due to the fact that the eigenvectors of this process are cosines at this ρ-value. Thus, the DCT transform is optimal in the energy packing sense with this process as input signal. This is one of the main reasons for emphasizing experiments with this class of signals.
The main focus of the experiments is to get a view of the performance of the Operational Rate-Distortion Encoder (ORDE), and on how different parameters and settings affect it. The object of each experiment is as follows:

Experiment no. 1:	Training of frame and VLC tables. ORDE rate-distortion performance for different initial frames.
Experiment no. 2:	ORDE rate-distortion performance compared to the theoretical RD function.
Experiment no. 3:	Bit rate versus signal to noise ratio (SNR) for different values of λ, number of frame vectors (K), number of value codewords (J), and maximum number of selected vectors per signal block (M_{max}).
Experiment no. 4:	Comparisons of ORDE and RD optimized Matching Pursuit in RD performance and in time consumption.
Experiment no. 5:	The reduction in calculations when introducing lower bounds in ORDE. (See Section 4.3.2.)

The training and testing signals are the same for all experiments, as well as the signal block length, $N = 16$. In experiments 1, 2, 3, and 5, *ordered* vector selection and Run-Length Coding is used in the ORDE, while a comparison between four different vector selection algorithms is made in experiment 4. In experiments 1, 2, 3, and 4, the number of signal blocks is $L = 2048$, while in experiment 5, $L = 64$. In experiments 1, 2, 4, and 5, $K = 32$, $J = 32$, and $M_{max} = 4$. In experiment 3, $K = \{16, 32, 48\}$, $J = \{16, 32, 64\}$, and $M_{max} = \{2, 3, 4, 5\}$.

6.2.1 Experiment no. 1

This experiment focuses on how the frame $\mathbf{F}$ and the VLC tables *values* and *runs* change as a result of the training described in Section 5.2. Three different frames are used as initial frame, all with dimensions 16×32 and with normalized column vectors:

Frame 1: Ad hoc frame. The column vectors are random vectors from an AR(1) process.

Frame 2: Ad hoc frame. The first 16 column vectors are the 16×16 Inverse Discrete Cosine Transform (IDCT), and the last 16 vectors are random vectors from an AR(1) process.

Frame 3: Designed frame from previous work [12]: Design for using Orthogonal Matching Pursuit on AR(1) signals.

An illustration of the frame column vectors of Frame 1, Frame 2, and Frame 3, is given in Figure 6.4.

In each iteration of the training scheme, the object is to find the minimum cost for all signal blocks, $C = D + \lambda R$, where R and D are the overall rate and distortion, respectively. A representative learning curve, where each iteration's cost, C, is shown in Figure 6.1. In this case, where $\lambda = 0.0005$, $M_{max} = 4$, and Frame 3 is the initial frame, the training terminates after 39 iterations. The number of iterations will vary from case to case. In the figure we can see a significant drop in the cost when entering training Loop 2. This drop is noticeable in the learning curve of all experiments in this work. The drop is taking place where the updating of the VLC tables starts. In Loop 1, the object is to reduce the distortion, by frame training. In Loop 2, a bit rate reduction is added, by VLC optimization, resulting in a considerable cost reduction.

If there were no frame training, but only VLC optimization, the cost function results would highly depend on how the initial frame is designed. This is demonstrated in Figure 6.3(a). The operational rate-distortion function (ORDF) is presented for three cases, where Frame 1, 2 and 3 are used as initial frames, respectively. To be accurate, it is discrete points on the *convex envelope* of the ORDF that are found. In this experiment eight points on the convex envelope are found by coding the *test* signal for eight different values of λ: $\{\frac{1}{2}, \frac{2}{2}, \ldots, \frac{8}{2}\}0.0005$. With no frame training, there is a big difference in the RD performance. In particular, Frame 1 is much worse than the other two, which is no surprise, since the frame is found randomly. Even though Frame 2 and Frame 3 have a better RD performance, frame training would improve the results. In Figure 6.2 the full training learning curves are shown. The initial cost values are quite different for the three frames, but the cost at last iteration of each curve are approximately the same. The number of iterations vary, particularly in Loop 1. While the Frame 3 case has 18 Loop 1 iterations, the Frame 2 case has 84. Loop 1 terminates when the cost, C, is no longer decreasing from one iteration to the next. It is not possible to tell, in advance, at what iteration Loop 1 terminates, but we can see it happens

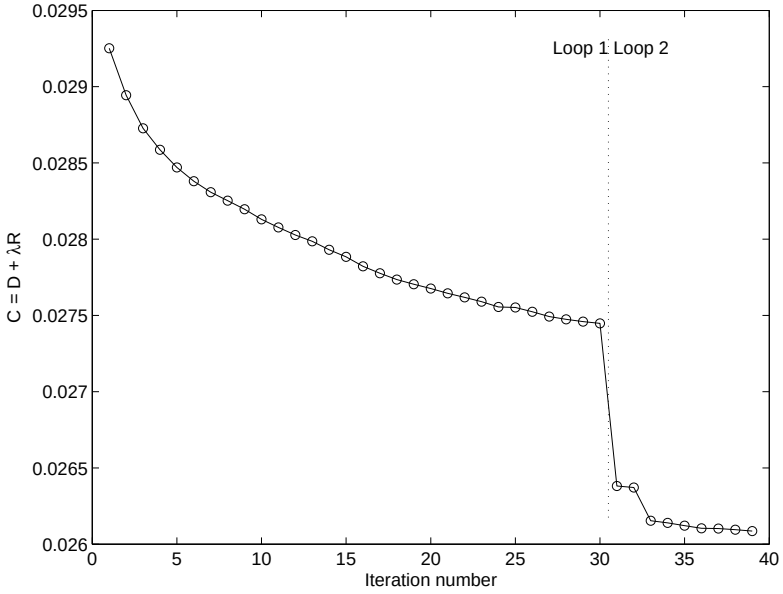


Figure 6.1: The learning curve when training the frame and optimizing the VLC tables on an AR(1) signal. The first 30 iterations are executed in Loop 1, and the last 9 in Loop 2. $L = 2048$, $\lambda = 0.0005$, $M_{max} = 4$. Initial frame is Frame 3.

when the learning curves has flattened out. The results of using the *trained* frames are presented in Figure 6.3(b). Compared to Figure 6.3(a) the results are better for all three cases, and the RD performance are, in practice, the same. The striking resemblance between the results makes it interesting to look closer to the three initial frames and their trained versions. The initial frames are shown in Figure 6.4, while their correspondingly trained versions are shown in Figure 6.5. The sample sets represent the frame column vectors in the order as they appear in **F**. We can see that the trained frames are quite different from the initial frames, in particular Frame 2. Frame 3 has the least changes. These observations correlate to the learning curves in Figure 6.2: It is the Frame 2 case that has the largest cost reduction. The trained frames in Figure 6.5 are more similar to each other than the initial frames. It seems like all three training experiments have converged to almost the same design. The similarity between the frames explain the approximately equal characteristics shown in Figure 6.3(b).

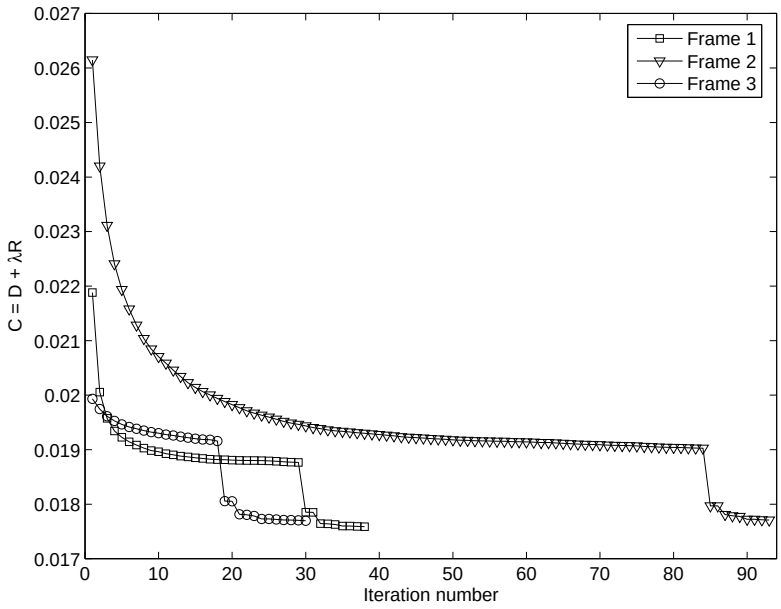
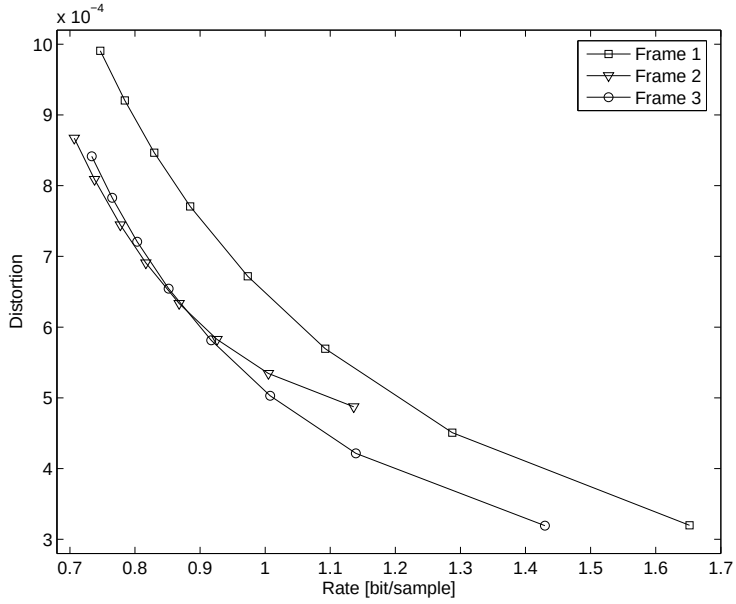
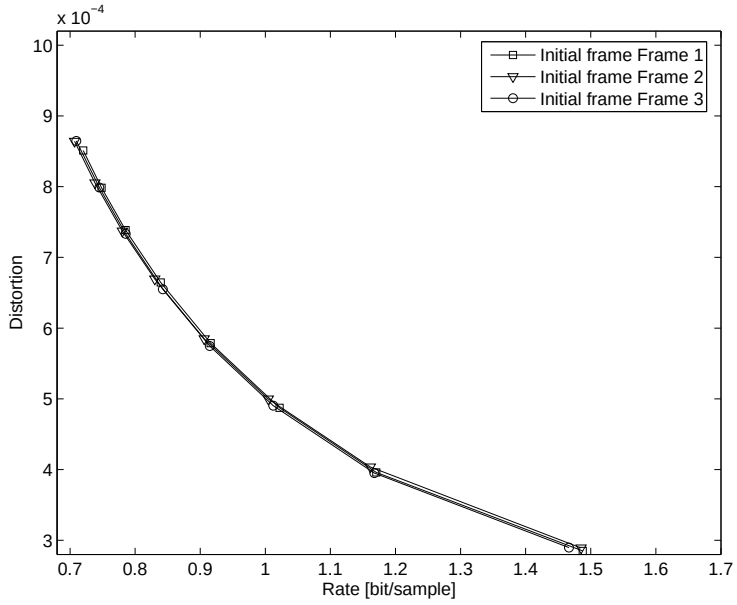


Figure 6.2: The learning curve for three different initial frames Frame 1, Frame 2, and Frame 3. $L = 2048$, $\lambda = 0.0005$, and $M_{max} = 4$.

In training Loop 1, the updated frame vectors are reorganized in an order where the most frequently used frame vector is the first one in the new frame. The other vectors are placed in order of descending usage frequency. (See



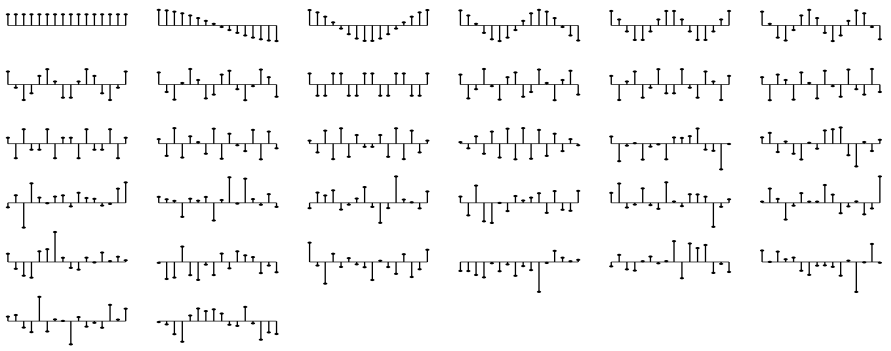
(a)



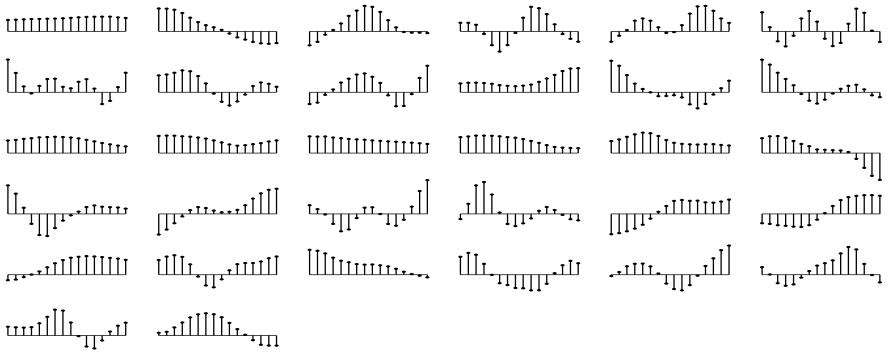
(b)

Figure 6.3: Bit rate versus distortion when encoding the test signal at eight different λ-values. Frame 1, 2 and 3 are used as initial frame. The results (a) with no frame training, and (b) with training included, are presented. Both (a) and (b) include VLC optimization.

(a)

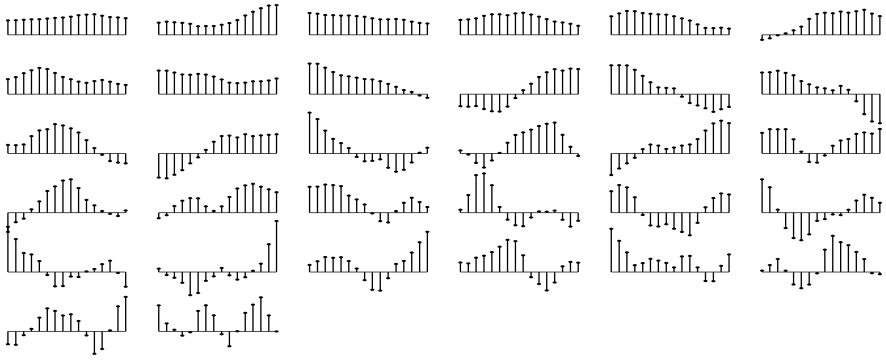


(b)

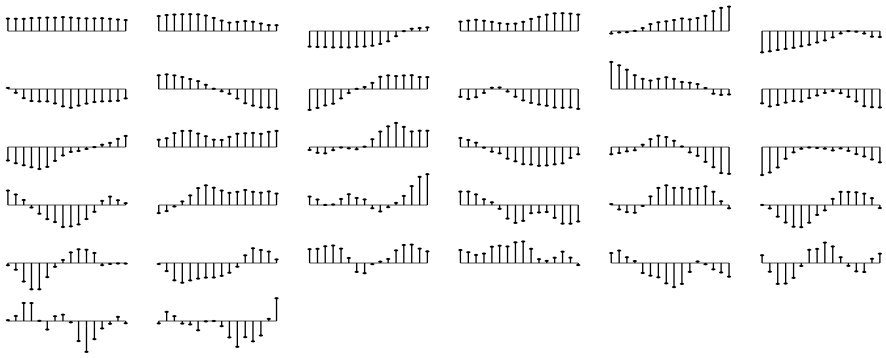


(c)

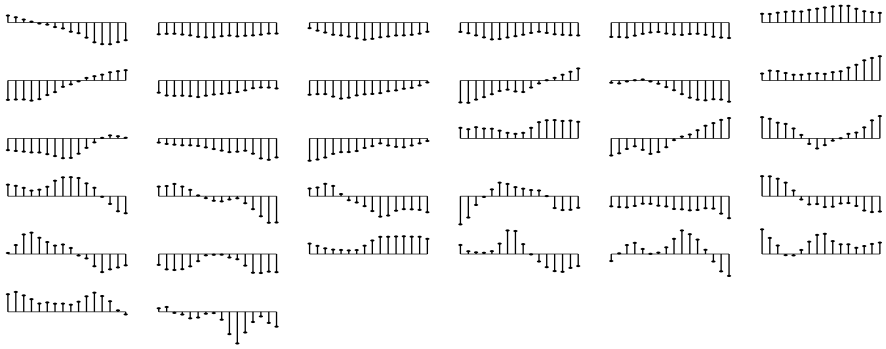
Figure 6.4: The 32 frame vectors in the initial frames (a) Frame 1, (b) Frame 2, and (c) Frame 3. A column in the frame is presented as one sample set in the figure.



(a)



(b)

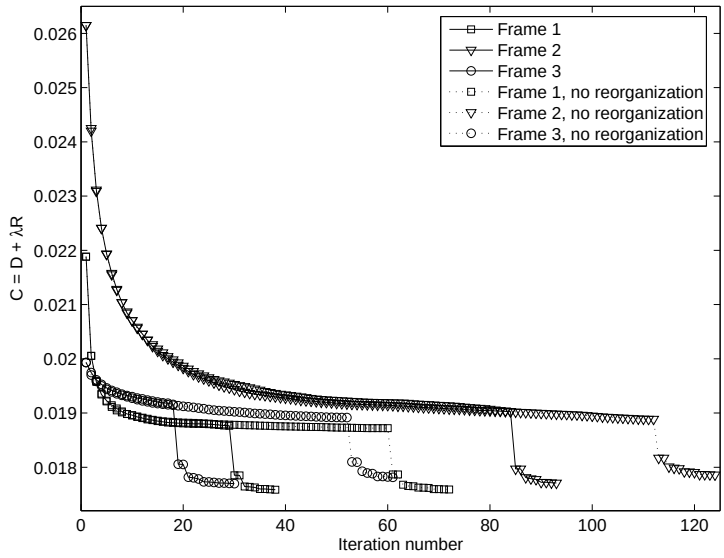


(c)

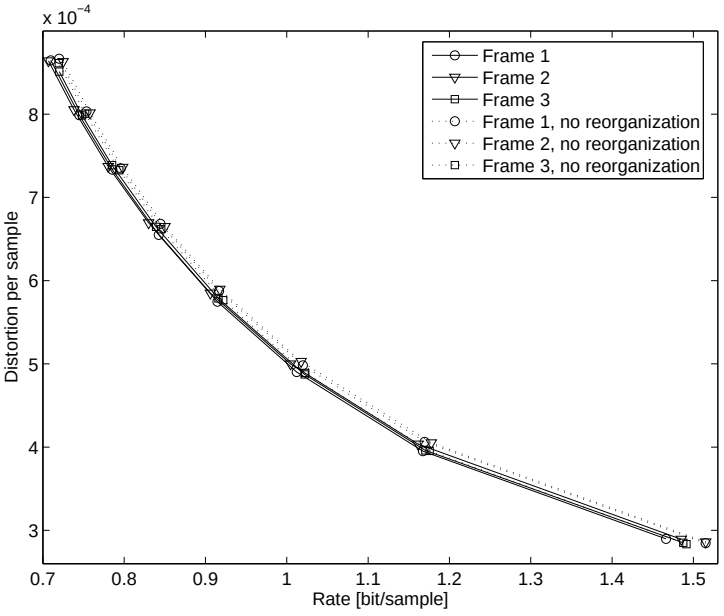
Figure 6.5: The frames after training, where the initial frames were (a) Frame 1, (b) Frame 2, and (c) Frame 3. A column in the frame is presented as one sample set in the figure.

Section 5.2.1). There will be fever long runs and more short runs between the nonzero coefficients, that after VLC optimization in Loop 2 will give us a lower bit rate for coding the *run* codewords. In Figure 6.6, the results from using this reorganization is shown as solid lines, while the dotted lines are the results from *not* having this reorganization in Loop 1. The plots show results from using Frame 1, Frame 2, and Frame 3 as initial frame, respectively. In all three cases, the training period expands with more than 30 iterations. In addition, the cost at the final training iteration is marginally worse. This is also the case for the test signal, where the rate-distortion plot in Figure 6.6(b) illustrates a small shift to the right in all curves when not including the reorganization. The bit rate is marginal higher for all situations, while the distortion is approximately at the same place. This demonstrates the value of using the reorganization of frame vectors in training Loop 1.

Not only the frame, but also the VLC tables for *run* and *value* codewords are updated in the training scheme. To set the initial codeword lengths for *values*, a Gaussian probability density function (pdf) with zero mean is used. In other words, we expect low values to be more frequently used than high values. The initial *run* codeword lengths are set based on a uniform pdf, saying that long and short runs between the nonzero coefficient have an equal probability of occurrence. From the last experiment where the effect of reorganization of frame vectors were demonstrated, this is not true. But, the difference is not large, and will in any case be updated in the first iteration of training Loop 2. The initial codeword lengths for *values* and *runs* are shown in Figure 6.7 (a) and (b), respectively. There are 32 *values* and 32 *runs* in this case. The initial *value* codeword lengths vary from 4 to 6 bits, where the small absolute values have 4 bits, and the large absolute values have 6 bits. The distribution of bits is chosen to be like this, because there is a higher likelihood that small values are used than large values. The 33rd bar in the *run* bit length plot represents the bit length of the End Of Block (EOB) symbol, which is initially set to be 1 bit. The reason for this is the high probability of this symbol. In this work Huffman coding is used to find the representation for all symbols. Due to this, all bit lengths are integers. An example of how the *value* and *run* codeword bit lengths have changed after training is shown in Figure 6.7 (c) and (d), respectively. The bit lengths of *values* indicates that the frequency of small, but not too small values, are large. Short *runs* are more frequent than longer *runs*, and the frequency of the EOB symbol is large, due to its bit length of 2 after Huffman coding.



(a)



(b)

Figure 6.6: (a) The learning curve from the training and (b) the rate-distortion curves from the testing where Frame 1, Frame 2, and Frame 3 are initial frames. The solid lines represent the case where replacement of frame vectors find place in training Loop 1, while the dotted lines are the cases without this replacement.

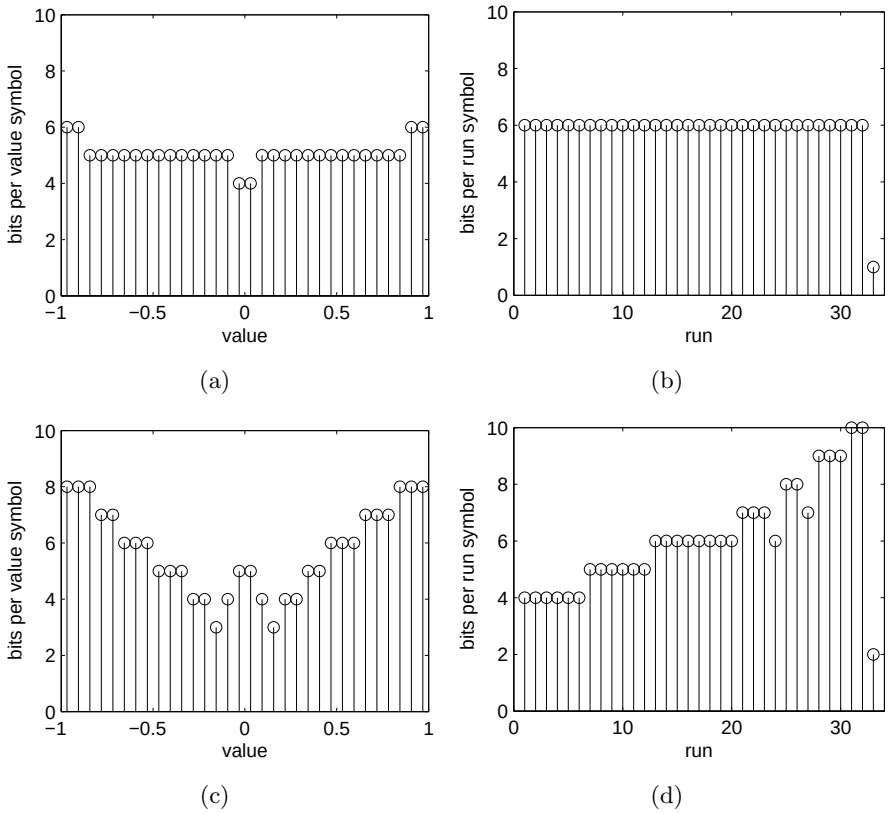


Figure 6.7: The initial bit lengths for the (a) *value* and (b) *run* codewords, and the bit lengths after last iteration of the training Loop 2 for the (c) *value* and (d) *run* codewords.

6.2.2 Experiment no. 2

In this experiment we want to show the RD performance of ORDE compared to the theoretical RD bound for an AR(1) process. We also want to show the difference of using a 16×16 Discrete Cosine Transform (DCT) and a 16×32 trained frame.

In the training scheme we use Frame 3 from the previous experiment as the initial frame. The learning curve is shown in Figure 6.8. In the same figure the learning curve of the DCT case is plotted. For both cases, $\lambda = 0.001$. Remember that the DCT is not trained, but kept intact through the entire training and testing procedure. It is only the VLC tables that are updated. This is the reason why it has just a few training iterations. Except from the initial encoding step, all of them take place in Loop 2.

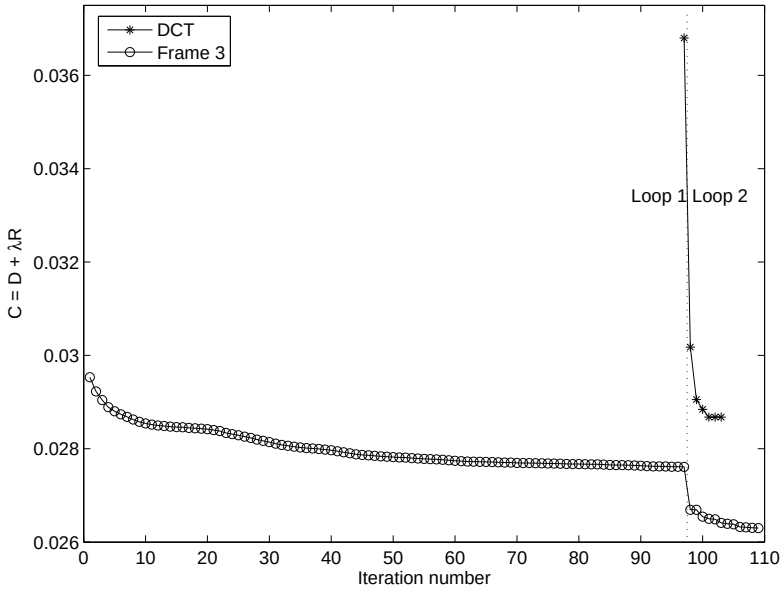


Figure 6.8: The learning curves for a case where a 16×16 DCT is used as frame, and a case where Frame 3 is trained. The DCT case has just a few number of iteration, since the Loop 1 frame training is omitted and only VLC optimization in Loop 2 is included.

The rate-distortion diagram in Figure 6.9 illustrates the RD performance of the two cases where the AR(1) test signal is input, and λ vary from 0.0005 to 0.004. Note that the y-axis is distortion per sample divided by σ_x^2, where σ_x^2 is the variance of the input signal. In this experiment, $\sigma_x^2 = 0.0105$, since the

input signal is scaled in order to fit the quantization range. The theoretical Rate-Distortion Function (RDF) is plotted as a dotted line, and found by using the parametrical equations in (6.1) and (6.2) for $D > 0.0256\sigma_x^2$. In the small distortion region, where $D \leq 0.0256\sigma_x^2$, Equation (6.5) is used to find RDF. The experimental results from using ORDE and a trained frame is represented with the solid line with circles. The line follows the curvature of RDF very well for the presented set of convex envelope points, where the bit rate is between 0.3 and 0.5 bit/sample larger than RDF at the same values of SNR. For the DCT case, the bit rates are 0.03 to 0.05 bit/sample larger than for the trained frame case.

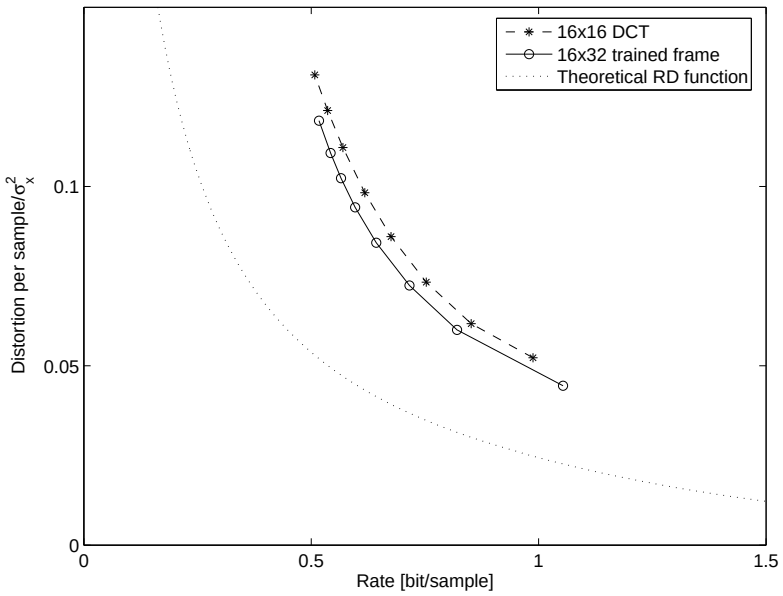


Figure 6.9: The rate-distortion results for the DCT and the trained frame case compared to the theoretical rate-distortion function. The y-axis is distortion per signal block divided by the input signal variance, σ_x^2.

6.2.3 Experiment no. 3

There are some parameter settings that will impact the RD performance of the coding algorithm of this work. In this experiment we will examine the influences of: The dimensions of $\mathbf{F}$ (N and K), the number of quantization steps (J), the Lagrangian multiplier (λ), and the maximum number of selected vectors per signal block (M_{max}). These entities do not only affect the RD

performance, but will also have a significant influence on the encoding time. The number of rows in $\mathbf{F}$, N, is set to 16, and is not adjusted to a higher value in this experiment, due to encoding time considerations. In most of the experiments in this work, $K = 2N$. If we double N, we would double K, which would increase the number of nodes in the tree from $\sum_{M=0}^{M_{max}} \binom{K}{M}$ to $\sum_{M=0}^{M_{max}} \binom{2K}{M}$ in the reduced tree case. If $K = 32$ and $M_{max} = 4$, we have 41449 nodes in the tree. If $K = 64$ and $M_{max} = 4$, the number of nodes is 679121. The doubling of K results in an increase in the number of nodes by a factor of approximately 16. If N is doubled as well, the time needed for calculations of each node would increase, and the total time consumption would cross the limit of reasonable time consumptions of encoding.

First, we present the RD results when using frames with different numbers of columns. $N = 16$ and $K = \{16, 32, 48\}$. The initial frames are all put together like Frame 1, which is a set of normalized random AR(1) column vectors. All three frames are trained with $\lambda = 0.0005$, prior to the testing on another AR(1) signal with eight λ-values in the range $[0.00025, 0.002]$. The RD results are presented in Figure 6.10, where the Signal-to-Noise Ratio (SNR) is used on the y-axis. SNR is defined by

$$SNR = 10 \log_{10} \frac{\sum_{l=1}^{L} \|\mathbf{x}_l\|^2}{\sum_{l=1}^{L} D_l}. \tag{6.6}$$

The y-axis has an inverted role compared to previous RD plots: The best RD curve is the one which is nearest to the upper left corner of the diagram. The three curves are very close to each other, but the case where $K = 16$ has a lower bit rate at coarse coding, while the cases where $K = 32$ and $K = 48$ has better rates when coding with smaller loss of information. The reason is that the number of bits needed to represent a *run* codeword is reduced at $K = 16$. The set of frame vectors to choose from is smaller, but this disadvantage does not seem to manifest itself before λ is small, and a higher bit rate is allowed.

A change in the number of quantization steps, J, will affect the RD performance of the ORDE. Note that an increase in J will increase the number of bits per symbol, but reduce the granular noise in the quantizer. The quantization overload noise is the same, since the quantization range is unchanged. Results from an experiment with $J = \{16, 32, 64\}$ is shown in Figure 6.11. Having just a few steps, $J = 16$, will reduce the number of bits of *value* codewords. This will result, as demonstrated in the figure, in a lower bit rate at coarse coding, but it has a disadvantage at fine coding, since the SNR will decrease as a result of the increase in quantization noise. The reasonable number of quantization levels has an upper limit. In this experiment it seems like 64

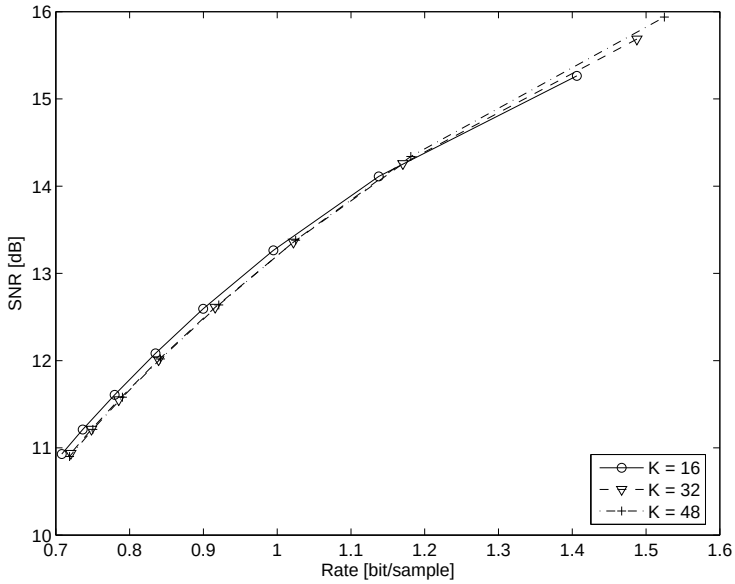


Figure 6.10: Rate versus SNR results of AR(1) test signal for different dimensions of $\mathbf{F}$: 16×16, 16×32, and 16×48.

quantization levels is too much. For the given set of λ-values, the case with 32 levels is all over 0.05 bit per sample better at the same SNR values.

The Lagrangian multiplier, λ, is the turning knob used to change between low and high bit rate coding. If $\lambda = 0$, the bit rate is not taken into consideration, and the reconstructed signal of the representation code will be of good quality, i.e., the SNR is large. When increasing λ, the bit rate will be reduced, and the same will the SNR value. This is illustrated in Figure 6.12. The curves show the results of coding the same AR(1) test signal, but with different trained frames. The same initial frame, Frame 3, is used, but the training has been done for seven different values of λ. For each of the seven training cases, testing is done with eight different Lagrangian multiplier values, $\boldsymbol{\lambda} = \{\frac{1}{2}, \frac{2}{2}, \ldots, \frac{8}{2}\}\lambda$, where λ is the Lagrangian multiplier in the training procedure. The theoretical RDF is presented in the plot as a dotted line, where RDF's SNR,

$$SNR_{RDF}(R) = 10 \log_{10} \frac{\sigma_x^2}{D_{RDF}(R)}. \tag{6.7}$$

$D_{RDF}(R)$ is found by (6.5) for the small distortion region, and by the parametrical equations, (6.1) and (6.2), for the rest of the curve.

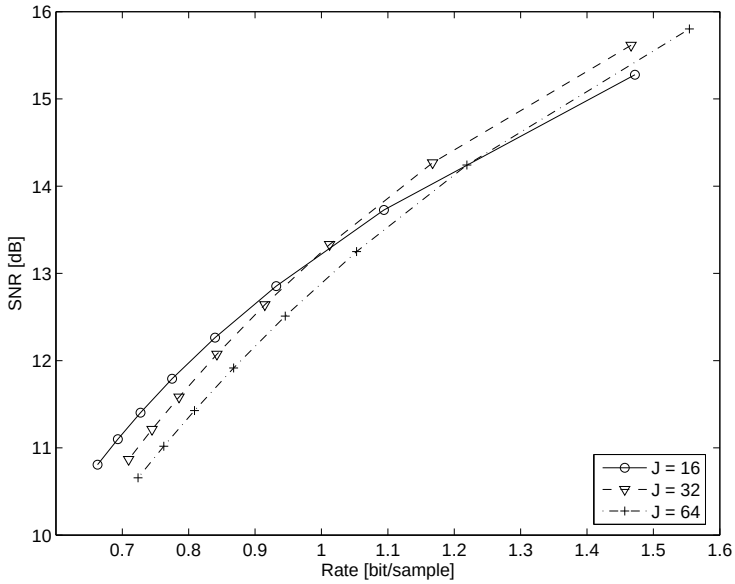


Figure 6.11: Rate versus SNR results of AR(1) test signal for different number of *value* symbols: 16, 32, and 64.

It is necessary to set an upper limit for the number of selected vectors of each signal block, in order to keep the algorithm's time consumption to a reasonable level. The value of this limit, M_{max}, will affect the RD performance of the encoder if there are signal blocks where the cost, $C_l = D_l + \lambda R_l$, would be smaller when having more than M_{max} selected vectors. In Figure 6.13 we can see the bit rate versus SNR results at different values of M_{max}. In Figure 6.13(a) the frame and VLC tables are trained at $\lambda = 0.0005$, while in Figure 6.13(b) $\lambda = 0.001$ in the training procedure. Thus, the eight points of the ORDF's convex envelope in the test signal case are at $\lambda = \{0.00025, 0.0005, \ldots, 0.002\}$, and $\lambda = \{0.0005, 0.001, \ldots, 0.004\}$, respectively. We can see in both diagrams that at rates less than 1 bit per sample there are approximately no differences in the results, even though M_{max} is as small as 2. At higher rates, we can see that the results are worse for the case where $M_{max} = 2$, than for the other cases. The differences between the other three cases, where M_{max} is 3,4, and 5, are very small at this bit rate range. But, the impact of M_{max} to the RD performance of the ORDE highly depends on λ. If λ is chosen to be smaller, the total number of bits can be larger, and the number of signal blocks where M_{max} prevents a better solution to be found, will increase.

In Figure 6.14, histograms show the number of signal blocks with 0, 1, 2, 3, 4, or 5 selected vectors, at the case where $M_{max} = 5$. In Figure 6.14(a),

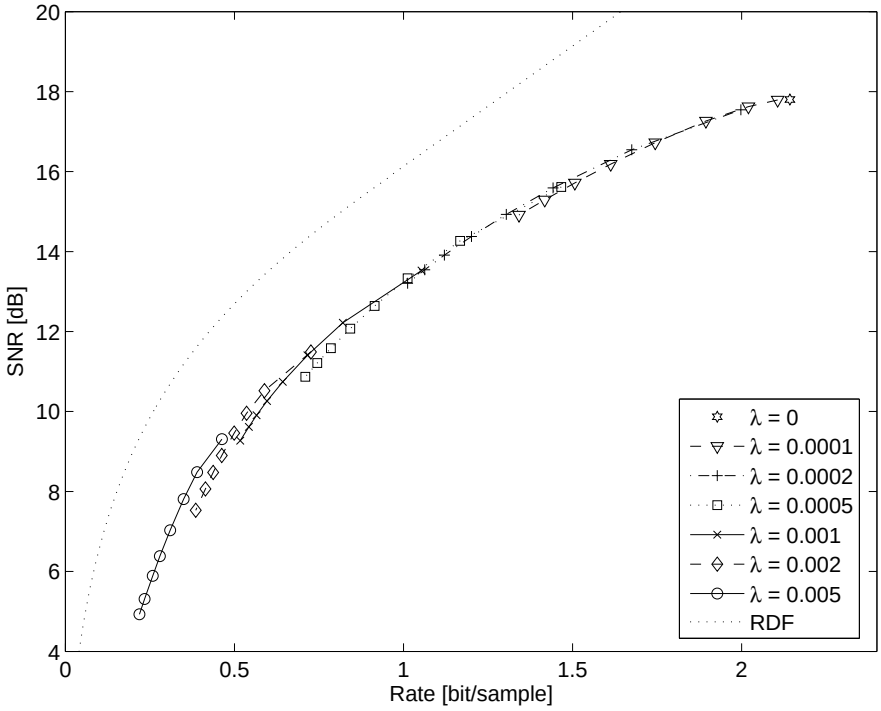
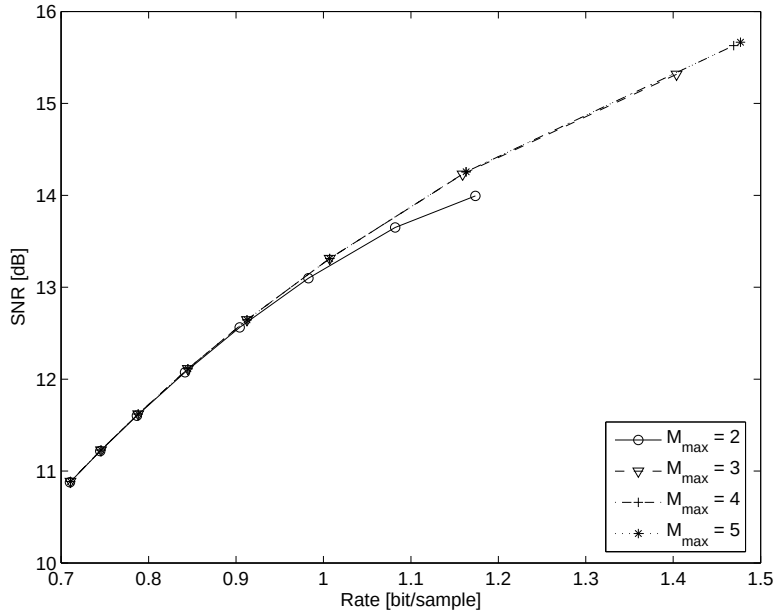
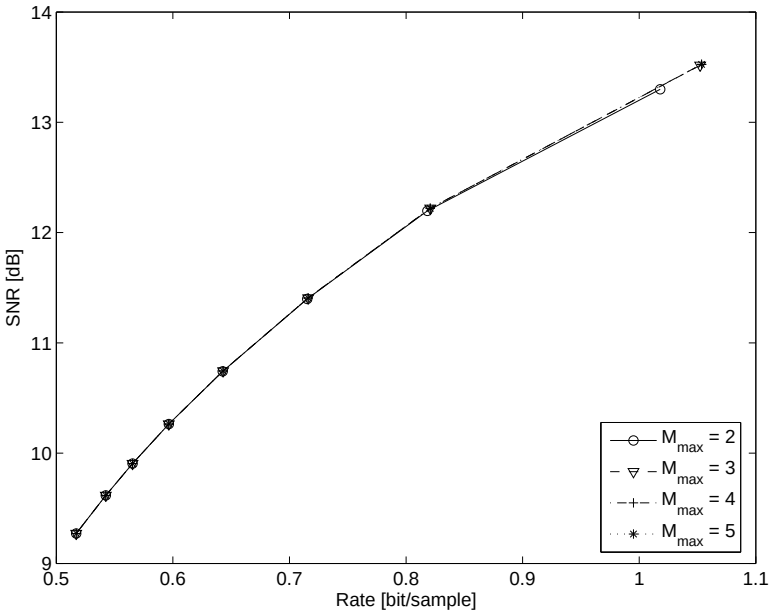


Figure 6.12: Rate versus SNR results for 7 different values of λ in the training scheme. The dotted line is the theoretical Rate-Distortion Function (RDF).



(a)



(b)

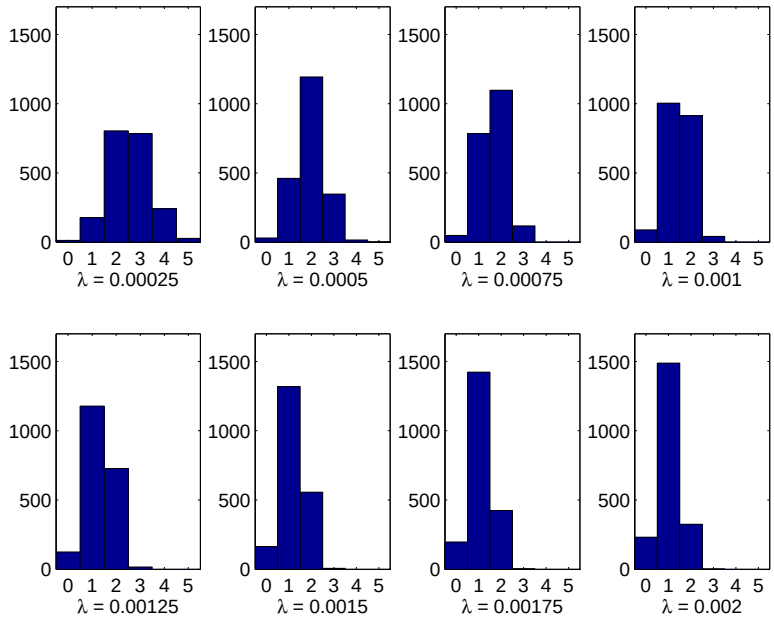
Figure 6.13: Rate versus SNR results of the test signal for different values of M_{max}. The Lagrangian multiplier in the training scheme is (a) $\lambda = 0.0005$ and (b) $\lambda = 0.001$.

$\lambda = 0.0005$ in the training procedure, while in Figure 6.14(b) the frame and VLC tables are trained at $\lambda = 0.001$. There are only two of the histograms that includes any signal blocks with as much as 5 selected vectors. In the case where $\lambda = 0.00025$ and $\lambda = 0.0005$ in Figure 6.14(a), there are 27 and 1 blocks with 5 selected vectors, respectively. This shows that if $M_{max} \geq 5$, the RD performance will not be deteriorated at $\lambda > 0.0005$ for this class of signals. In Figure 6.14 (a) and (b), there are several histograms with the same λ-value in the test case. But, these histograms are not equal. The only difference in the setups is that λ-values in the training procedure are 0.0005 and 0.001, respectively. For the latter situation, all corresponding histograms show a small increase in the number of signal blocks with a small number of selected vectors. This observation verify that the λ-value should be chosen carefully, in order to find the best designed frame and VLC tables according to the desired rate and distortion features for the given class of input signals.

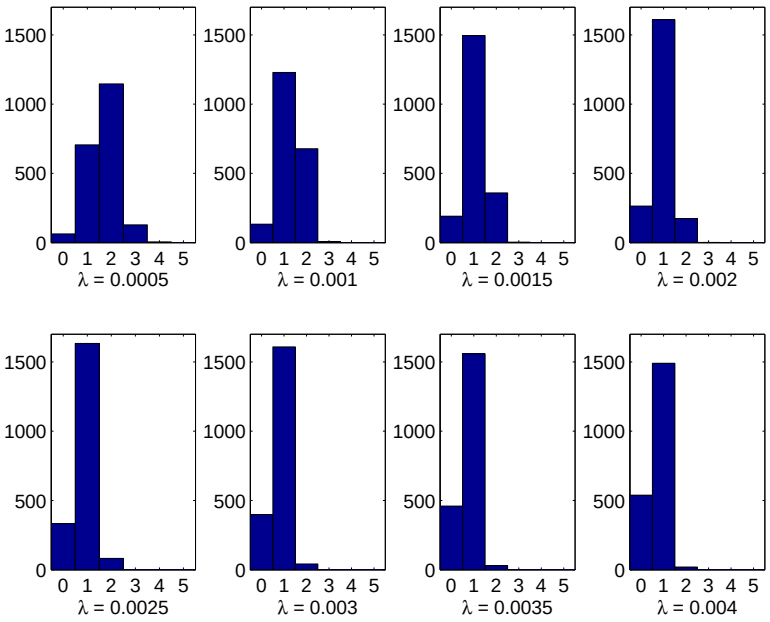
6.2.4 Experiment no. 4

We will now compare the ORDE algorithm with Rate-Distortion Optimized (RDO) Matching Pursuit, proposed in [17], and described in Section 3.4. Both RDO Basic (or standard) Matching Pursuit (RDO BMP) and RDO Ordered Recursive Matching Pursuit (RDO ORMP) will be tested. RDO BMP and RDO ORMP are both greedy vector selection algorithms. They are much faster than the ORDE algorithm, but they are suboptimal. In Table 4.2 d) in Section 4.5, the algorithm complexity of ORDE is equal to $1 + \sum_{M=1}^{M_{max}} \binom{K}{M} J$. The corresponding worst case complexity for RDO BMP and RDO ORMP is M_{max}. In order to get a fair comparison, we have to use the ORDE algorithm with *index* coding (ORDE *ind*) instead of Run-Length Coding (RLC). This is the full tree search algorithm described in Section 4.3.2. The reason for using this algorithm is that the RDO matching pursuit techniques can not use RLC. The frame vectors are selected one by one and in an arbitrary order. This allows a frame vector with a lower index number than the previous selected one to be chosen. If RLC had been used, the new coefficient would destroy the previous selected coefficient's *run* data. Thus, index coding is used in RDO BMP and RDO ORMP. Even though we cannot compare the matching pursuit techniques directly to the ORDE with Run-length Coding (ORDE *run*) presented in Section 4.4, we show the latter algorithm's performance in the same diagrams.

The learning curves are all shown in Figure 6.15. In Figure 6.15(a), $\lambda = 0.0005$, and in Figure 6.15(b), $\lambda = 0.001$. $M_{max} = 4$ in both cases. In all four curves the big drop in the cost, which is the intersection between training Loop 1 and



(a)



(b)

Figure 6.14: The number of signal blocks where the optimal solution has $0, 1, \ldots, 5$ selected frame vectors. The Lagrangian multiplier in the training scheme is (a) $\lambda = 0.0005$ and (b) $\lambda = 0.001$.

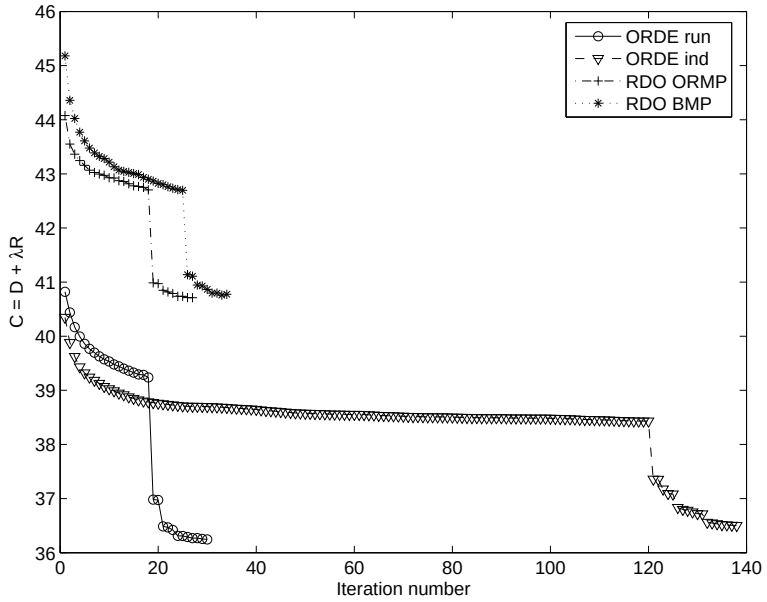
2, occurs after a random number of iterations. The total number of iterations is varying, as well. As is clearly seen, the ORDE algorithms outperforms the RDO matching pursuit algorithms in both figures. ORDE *run* has the best cost, and its training procedure terminates after fewer iterations. When the trained frames and VLC tables are used as inputs in the four respective test cases, we can see the results in the rate versus SNR plots in Figure 6.16. As in the previous figure, the Lagrangian multiplier in the training scheme is (a) $\lambda = 0.0005$ and (b) $\lambda = 0.001$. At low bit rates the four algorithms have approximately the same SNR. But, as the rate increases the ORDE algorithms are much better than the MP algorithms. At 1.2 bits per sample, the ORDE algorithms are approximately 1 dB better. In both (a) and (b), ORDE *run* is slightly better than ORDE *ind*, even though ORDE *ind* has a larger set of solutions. The reason why ORDE *run* is doing better is that Run-Length Coding is used. Encoding the *runs* instead of the *indices* results in a more bit efficient representation.

As mentioned, there is a difference in the time consumptions of the four algorithms presented. With $M_{max} = 4$, the difference is considerable. Time results are shown in Table 6.1. The table shows the average time used to generate one RD-point in the curves of Figure 6.16 (a) and (b). As the MP algorithms use 9 and 13 seconds in average to encode 2048 signal blocks, the ORDE *run* algorithm uses 271 seconds in average for the same blocks. ORDE *ind* uses 5569 seconds, or one hour and 33 minutes, in average. A full search is very time consuming. ORDE *ind* is more than 20 times more time consuming than ORDE *run*. At the same time, the ORDE *run* often performs better in the rate-distortion sense, like it did in this experiment. See Figure 6.16.

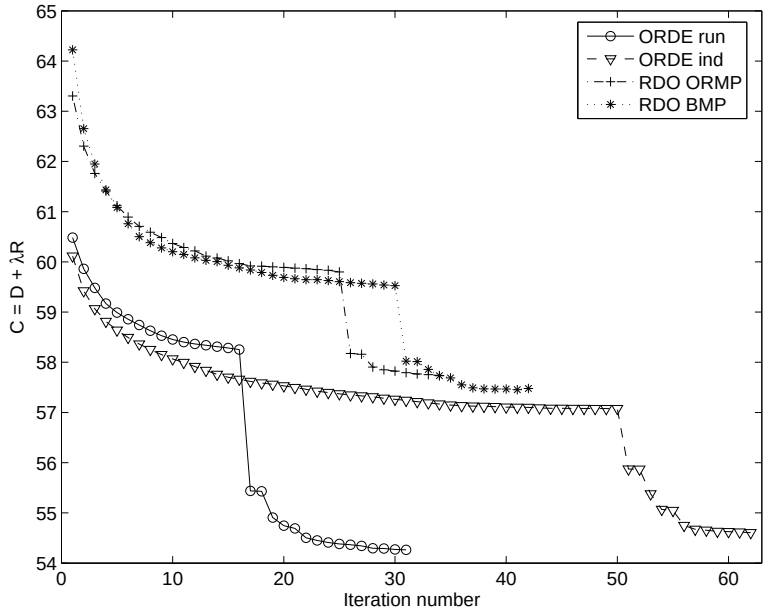
The algorithms are implemented in Matlab® [26]. Parts of the algorithms are implemented in C++, in order to speed up the encoding of each signal block. The computer used in all experiments is a PC with a 2.4 GHz Intel Pentium 4 processor.

6.2.5 Experiment no. 5

The last experiment on AR(1) signals focus on the time consumption reduction by the introduction of *lower bounds*, presented in Section 4.3.2. It is not necessary to do a full calculation of each and every node in the solution tree, if we can eliminate a particular node from being optimum based on some knowledge of the node cost and the so far best solution, C_l^*. For a particular node, we know the lower bound (LB) of the cost, given by (4.21). If $C_l^* < LB$, it is not necessary to evaluate this node any further. We can not prune the

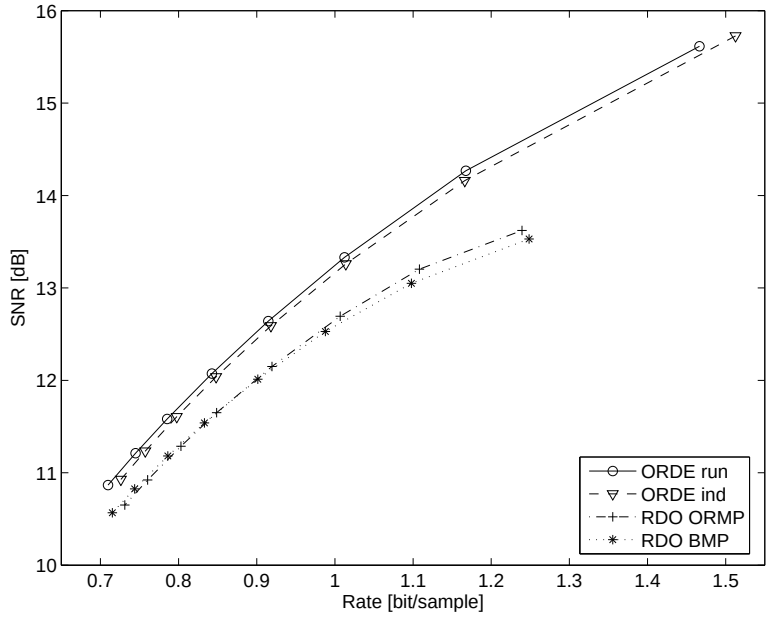


(a)

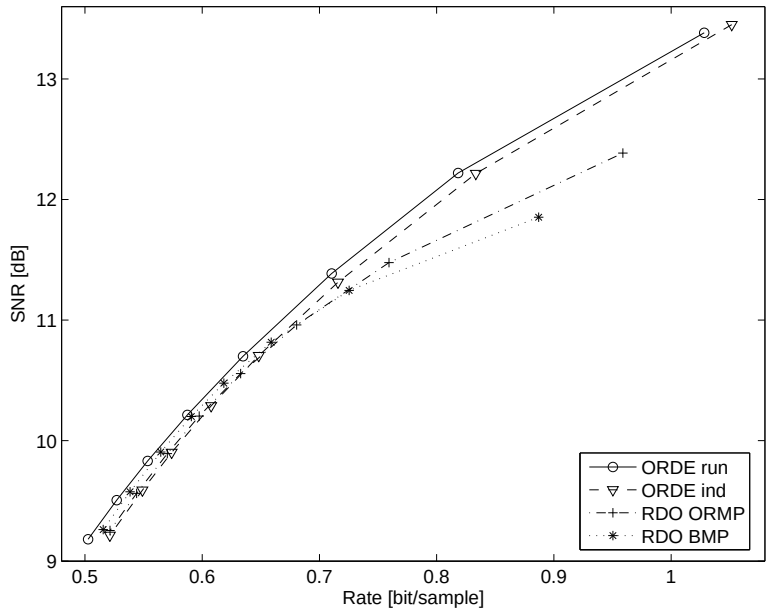


(b)

Figure 6.15: Learning curves for four different algorithms: ORDE *run*, ORDE *ind*, RDO ORMP, and RDO BMP. The Lagrangian multiplier in the training scheme is (a) $\lambda = 0.0005$ and (b) $\lambda = 0.001$.



(a)



(b)

Figure 6.16: Rate versus SNR test signal results for four different algorithms: ORDE *run*, ORDE *ind*, RDO ORMP, and RDO BMP. The Lagrangian multiplier in the training scheme is (a) $\lambda = 0.0005$ and (b) $\lambda = 0.001$.

Algorithm	Average time in seconds
RDO BMP	9
RDO ORMP	13
ORDE *run*	271
ORDE *ind*	5569

Table 6.1: The average time used to encode 2048 signal blocks for the RDO BMP, RDO ORMP, ORDE *run*, and ORDE *ind* algorithms, when $M_{max} = 4$.

tree based on this knowledge. Eventual child nodes have to be evaluated, but the chance of a full calculation is smaller if the parent node has not been fully calculated. In other words, we are in a branch of the tree that seems not to include the optimal node. Figure 6.17 shows the in percentage number of fully calculated nodes in the tree, for different values of M_{max} and λ. The solid lines and the dotted lines represent the test results where $\lambda = 0.0005$ and $\lambda = 0.001$ in the training loops, respectively. Except for small values of λ, $\lambda = 0.0001$, the number of fully calculated nodes decreases as the tree is getting deeper, by increasing M_{max}. In all the 24 different encodings of the signal, it is less than 1.6 % of the nodes that are fully calculated. This results in a considerable reduction in the time used to encode the signal in a rate-distortion optimal way.

6.3 Summary

The experiments on AR(1) signals can be summarized as follows: The results show the importance of having a well designed frame and optimized VLC tables for the particular class of signals. Compared to the suboptimal RDO Matching Pursuit algorithms, the algorithms presented in this work, the Operational Rate-Distortion Encoder (ORDE) are slower but they perform better than RDO BMP and RDO ORMP.

For each of the five experiments presented, we can summarize as follows:

Experiment no. 1:
Training of frames is essential. The main thing is not the shape of the initial frame, but that a complete training is performed, both by a coarse updating in Loop 1 and a finer updating in Loop 2. Yet, if a specific initial frame should been chosen, Frame 3 is preferred. The update of VLC tables in Loop 2 are essential. The experiment shows that small absolute *values* are more

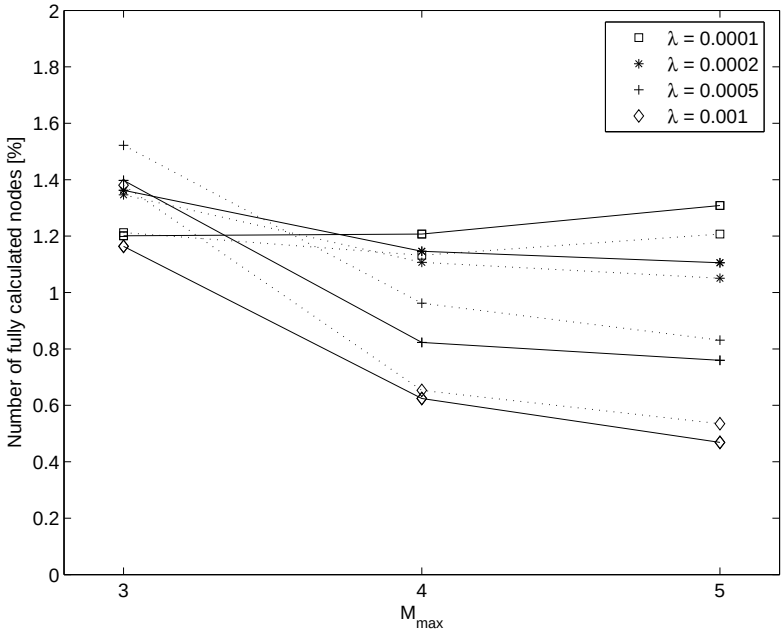


Figure 6.17: The percentage number of fully calculated nodes for different values of M_{max} and λ.

frequently used than large absolute *values*, and consequently are the codewords with the smallest number of bits. In fact, it is the third smallest absolute *values* that are most frequently used. In the *run* codeword table, the smallest *run* codewords are more frequently used, thus they have the smallest number of bits.

Experiment no. 2:
The results from using a 16×32 trained frame is better than from using a 16×16 DCT. For the same range of SNR, the bit rate is 0.03 to 0.05 bit/sample less. Compared to the theoretical RDF, our algorithm has a bit rate which is 0.3 to 0.5 bit/sample larger, for the given set of convex envelope points.

Experiment no. 3:
The size of the frame has some impact on the RD performance. Using a 16×16 trained frame gives better results at low bit rates, while 16×32 and 16×48 trained frames are best at higher rates than 1.1 bit per sample. The preferable number of quantization steps, J, is 32, Particularly for bit rates larger than 1 bit per sample. At lower bit rates, J should be less. In the training scheme, λ should be chosen carefully, in order get the best possible trained frame and VLC tables for the preferred range of bit rates. The value of M_{max} should not be too large, in order to keep the time consumption at an acceptable level. At the same time, it must not be too small, such that the RD performance is reduced. The right value of M_{max} depends on λ. If λ is not too small, M_{max} can be small. In this experiment, when $\lambda = 0.0005$ it is acceptable that $M_{max} = 4$.

Experiment no. 4:
The RDO MP algorithms are much faster than the ORDE algorithms. While RDO BMP and RDO ORMP use 9 and 13 seconds in average to encode 2048 signal blocks, ORDE *run* and ORDE *ind* use 271 and 5569 seconds in average, respectively. Remember that this is when *encoding* the signal. The *decoding* is executed very quickly for all four algorithms. The differences in the RD performance is negligible at rates lower than 0.7 bits per sample, but at higher rates, the ORDE algorithms outperforms RDO MP. At 1.2 bits per sample, the ORDE algorithms are approximately 1 dB better. ORDE *run* are better than ORDE *ind* for all rates in this experiment.

Experiment no. 5:
For AR(1) signals, and with these λ-values, 98.4 % of the nodes do not need full calculation. This demonstrates the benefit of introducing the lower bound technique, in order to reduce the total time consumption of the ORDE algorithm, without sacrifice of optimality.

Even though conclusions are drawn for AR(1) processes as input signals, the encoder settings can be quite different for other classes of input signals. This is shown in next chapter, where electrocardiograms (ECGs) are used as examples of real-world input signals.

Chapter 7

Compression of ECG signals

In this chapter we show experimental results from coding of electrocardiogram (ECG) signals, using the Operational Rate-Distortion Encoder (ORDE). Compression of ECG signals has been performed in previous work, employing many different techniques. Examples are sub-band coding [25], frame based coding [12], dynamic programming in a shortest path formulation [23], and rate-distortion optimal coding in time domain [37]. The focus in this chapter is rate-distortion optimal frame based coding of ECG signals.

For ECG, and other real-world signals, it may be hard to find the right range of *value* symbols, or quantization levels. A quantization step size optimization procedure is proposed in Section 7.1. This is incorporated into the previously presented training scheme. Due to the large dynamic range in an ECG signal, it is preferable to preprocess the signal prior to the encoding by ORDE. Two different preprocessing proposals are presented in Section 7.2. The experiment results are given in Section 7.3. Section 7.4 summarizes the findings of the chapter.

7.1 Quantization step size optimization

So far in this work, the *value* symbols have been fixed, both in the training and the testing scheme. These symbols are representation values from a mid-rise uniform quantizer [16]. In a uniform quantizer the distance from one value to the next in the table is constant. This constant is the quantization step size, Δ. The J *values* can be represented as a vector, **val**, given by

$$\mathbf{val}(\Delta) = \Delta\mathbf{a}, \tag{7.1}$$

where

$$\mathbf{a} = \left[\; -\frac{J+1}{2}+1 \qquad -\frac{J+1}{2}+2 \qquad \cdots \qquad -\frac{J+1}{2}+J \; \right]^T. \tag{7.2}$$

When J is even, there are $J/2$ positive and $J/2$ negative elements in $\mathbf{a}$, in a symmetric distribution around zero. Δ and J are the two parameters that fix the range of **val**. For real-world signals, like ECG signals, it is hard to find the best prefixed range of **val** for a particular signal, without a time consuming "trial-and-error" testing. This is why we will include an optimization procedure for Δ in the training scheme. Our objective is: *Given the set of frame vector coefficients and the input signal, find the minimum rate-distortion solution with respect to* Δ. We still have a uniform quantizer, since Δ is a scalar, but we search for another value of Δ that would give us better rate-distortion results.

First, we note that it is only the distortion that can be changed, as each coefficient is connected to the same *values* codeword, whose bit rate is unchanged. Thus, our objective function can be reduced to

$$\begin{aligned} z &= \min \sum_{l=1}^{L} \|\mathbf{x}_l - \mathbf{Q}_l \tilde{\mathbf{v}}_l^o\|^2 \\ &= \min \sum_{l=1}^{L} (\mathbf{x}_l^T \mathbf{x}_l - 2\mathbf{x}_l^T \mathbf{Q}_l \tilde{\mathbf{v}}_l^o + \tilde{\mathbf{v}}_l^{o^T} \tilde{\mathbf{v}}_l^o). \end{aligned} \tag{7.3}$$

The first term in the sum, $\mathbf{x}_l^T \mathbf{x}_l$, is a constant, thus it can be removed from the objective function. Next, we write the $M \times 1$ vector $\tilde{\mathbf{v}}_l^o$ as

$$\tilde{\mathbf{v}}_l^o = \mathbf{\Theta}_l \mathbf{val}, \tag{7.4}$$

where $\mathbf{\Theta}_l$ is an $M \times J$ binary (0,1)-matrix. We have only one "1" in each row. The position of the "1" determines the value of the coefficient. Now, the objective function can be written as

$$z(\mathbf{val}) = \min_{\mathbf{val}} \sum_{l=1}^{L} (\mathbf{val}^T \mathbf{\Theta}_l^T \mathbf{\Theta}_l - 2\mathbf{x}_l^T \mathbf{Q}_l \mathbf{\Theta}_l)\mathbf{val} \tag{7.5}$$

$$\begin{aligned} z(\Delta) &= \min_{\Delta} \sum_{l=1}^{L} (\Delta \mathbf{a}^T \mathbf{\Theta}_l^T \mathbf{\Theta}_l - 2\mathbf{x}_l^T \mathbf{Q}_l \mathbf{\Theta}_l)\Delta \mathbf{a} \\ &= \min_{\Delta} \sum_{l=1}^{L} (\Delta^2 \mathbf{a}^T \mathbf{\Theta}_l^T \mathbf{\Theta}_l \mathbf{a} - 2\Delta \mathbf{x}_l^T \mathbf{Q}_l \mathbf{\Theta}_l \mathbf{a}). \end{aligned} \tag{7.6}$$

z is differentiable, and we can find local extreme values at $\frac{dz}{d\Delta} = 0$:

$$\frac{dz}{d\Delta} = 2\delta \sum_{l=1}^{L} (\mathbf{a}^T \mathbf{\Theta}_l^T \mathbf{\Theta}_l \mathbf{a}) - 2 \sum_{l=1}^{L} (\mathbf{x}_l^T \mathbf{Q}_l \mathbf{\Theta}_l \mathbf{a}) = 0. \tag{7.7}$$

The function $z(\Delta)$ is convex and it has only one extremum, thus the extremum is a minimum, which is located at

$$\Delta = \frac{\sum_{l=1}^{L} \mathbf{x}_l^T \mathbf{Q}_l \mathbf{\Theta}_l \mathbf{a}}{\sum_{l=1}^{L} \mathbf{a}^T \mathbf{\Theta}_l^T \mathbf{\Theta}_l \mathbf{a}}. \tag{7.8}$$

This is the optimal solution of (7.3) with respect to Δ. (7.8) is implemented in the training scheme as presented in Figure 7.1, just after ORDE in both training loops. This figure is a part of Figure 5.1 in Chapter 5. The shaded boxes represent the optimization of Δ, and the subsequential update of the *value* symbols, by using the new Δ in (7.1). In Figure 7.1, there is added a loop called "Loop 1.5". The contents of this loop is shown in Figure 7.2. It includes only two blocks, the ORDE and the Δ-optimizer, and the algorithm will continue iterating until the cost reduction from one iteration to the next is less or equal to a small positive number, ε. One may think that since both blocks produce an optimal cost solution to their problems, this loop would always terminate after one iteration. This is not true, due to the fact that output values of ORDE are input values of the Δ-optimizer, and vice versa. But, the cost will converge to a local minimum, like it does in Loop 2. The effects of introducing Δ-optimization in the training scheme is presented in Section 7.3.1.

7.2 ECG signal preprocessing

Typically, an ECG signal has a large dynamic range. In Figure 7.3 we can see a section of an ECG signal of a healthy human. The peaks are the heartbeats. Between these peaks the signal amplitudes are small, but not insignificant. From a medical point of view, it is important to take care of all of the heart activity information.

Due to the fact that most of the signal energy is in the heartbeat peaks, the number of coefficients per signal block, M, would be larger in these parts of the signal than in other parts, in order to keep the overall distortion at a low level. The number of coefficients would be more evenly distributed among

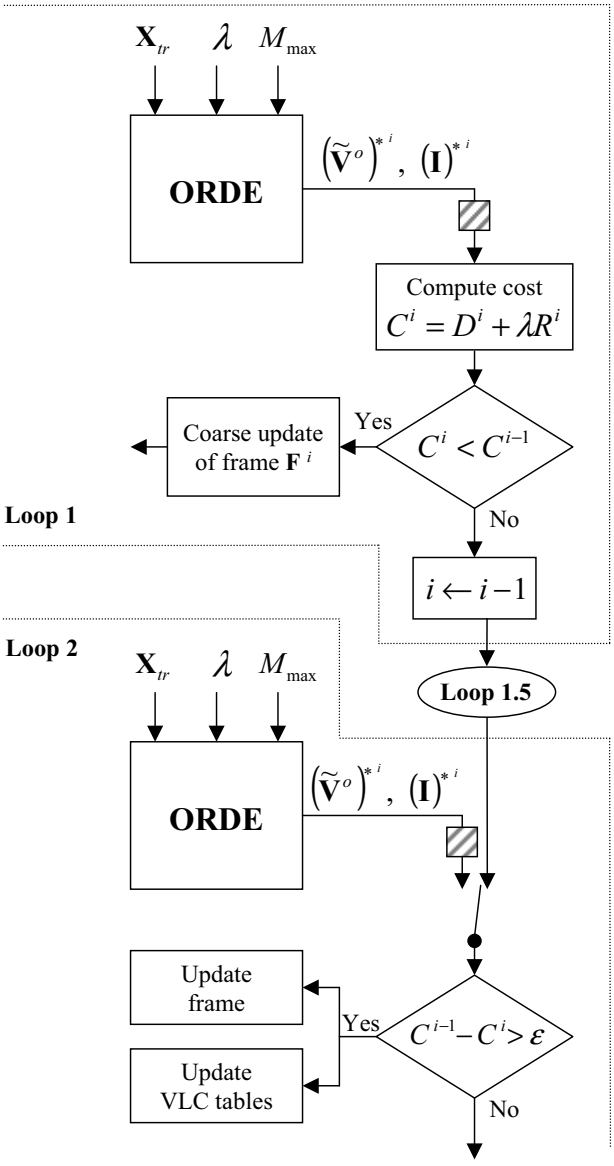


Figure 7.1: A part of the training scheme presented in Figure 5.1, with quantization step size (Δ) optimization added, as shaded boxes. Δ-optimization is added into "Loop 1.5", as well, which is illustrated in Figure 7.2.

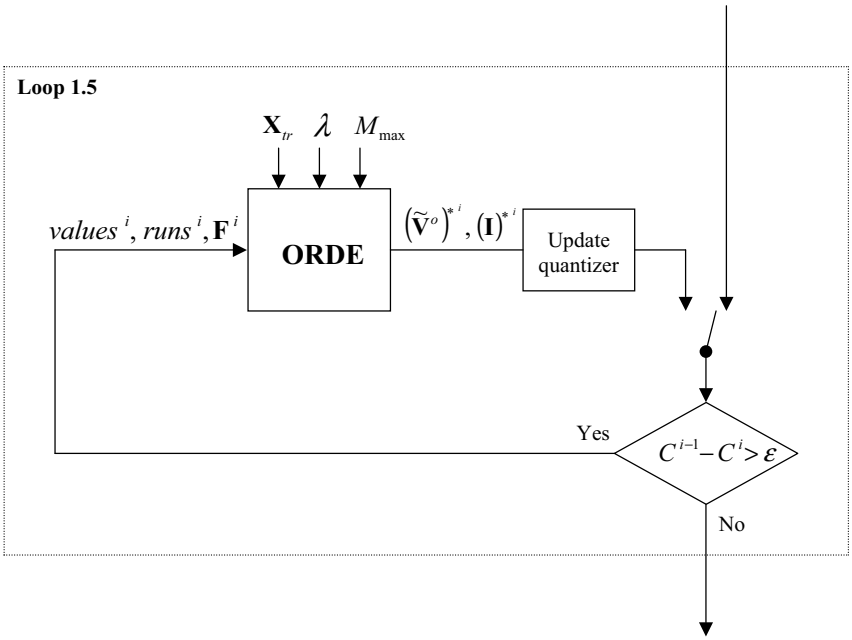


Figure 7.2: "Loop 1.5", where only two blocks are included, the ORDE and the Δ-optimizer.

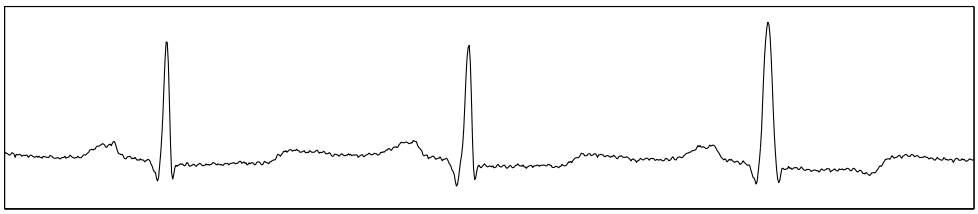


Figure 7.3: A section of an ECG from a healthy human.

signal blocks if the dynamic variations were smaller. This is the reason for introducing a lossless preprocessing of the ECG signal, before it is encoded by the proposed encoder of this work. We will look at two different preprocessors, presented as Preprocessor A and Preprocessor B. In both preprocessors, the idea is to extract some information from the blocks of signal samples. This information is encoded by a lossless coder, generating a string of bits which has to be transmitted, in addition to the bit stream produced by the ORDE. An advantage of this preprocessing is the reduction in both bit rate and distortion of the remaining signal. A disadvantage is the increase in the time of decoding the signal. A lossless decoder has to be added. We justify this time increase, due to the fact that the time needed to decode the signal, in any circumstances, is much lower than the time needed to encode it. A more detailed description of Preprocessor A and Preprocessor B is given in Section 7.2.1 and Section 7.2.2, respectively.

7.2.1 Preprocessor A

In this preprocessor, the *mean values* of each of the L signal blocks are found. The original signal, $\mathbf{X}$, is a matrix with dimensions $N \times L$. If μ_l is the mean value of signal block l, the entire set of mean values is organized as the $L \times 1$ vector $\boldsymbol{\mu}$. Since we have high correlation between adjacent mean values, it makes sense to use Differential Pulse Code Modulation [16] (DPCM) encoding on $\boldsymbol{\mu}$. The DPCM representation symbols are entropy encoded using Huffman coding, resulting in "Bit stream A". In Figure 7.4(a), the entire preprocessor is shown. The bit stream has to be decoded by an entropy decoder and an inverse DPCM, resulting in the reconstructed mean value vector, $\breve{\boldsymbol{\mu}} = [\ \breve{\mu}_1 \ \ \breve{\mu}_2 \ \ \cdots \ \ \breve{\mu}_L\ ]^T$. The input signal of ORDE, $\mathbf{E}$, is the difference between the original signal, $\mathbf{X}$, and the reconstructed preprocessor signal, $\breve{\mathbf{X}}$:

$$\mathbf{E} = \mathbf{X} - \breve{\mathbf{X}} = \mathbf{X} - [\ 1 \ \ 1 \ \ \cdots \ \ 1\ ]^T \breve{\boldsymbol{\mu}}^T, \tag{7.9}$$

where the vector of all ones is of length N. (7.9) says that each element of column l in $\breve{\mathbf{X}}$ is the reconstructed mean value of signal block l. Thus, column l in $\mathbf{E}$ is signal block l with zero mean.

The bit stream of *values* and *runs* from ORDE, and "Bit stream A", are together the compressed representation of the original signal. At the decoder, the reconstruction of $\mathbf{E}$, $\tilde{\mathbf{E}}$, is found by using the algorithm in Figure 2.4, for all L signal blocks. The decoding of "Bit stream A" is presented in Figure 7.4(b), and it is identical to the decoding in the preprocessor, used to generate $\breve{\boldsymbol{\mu}}$ and $\breve{\mathbf{X}}$. The reconstruction of the original signal is found by

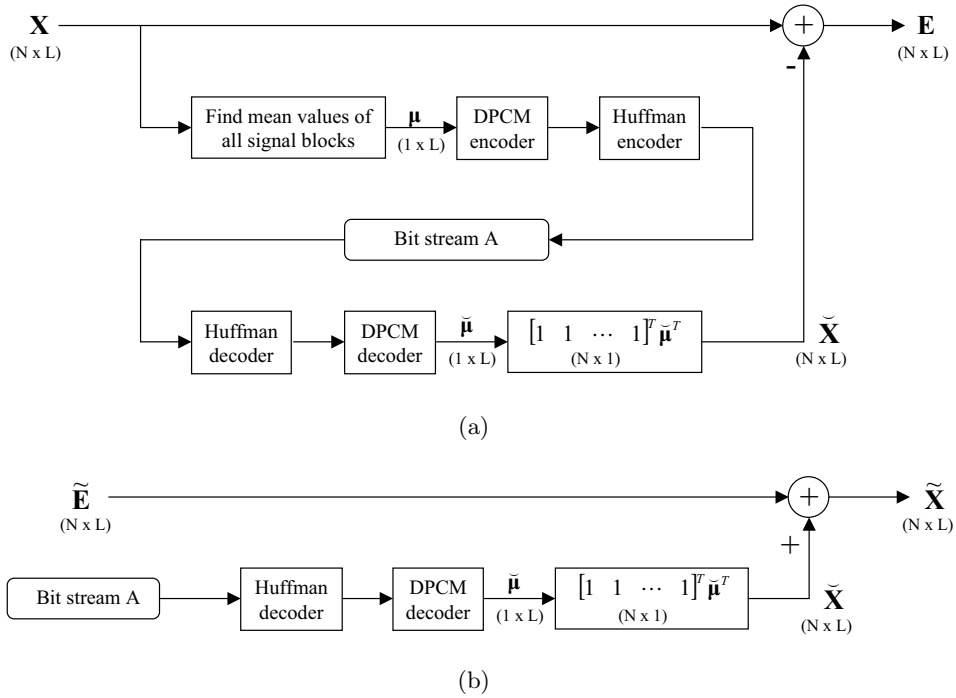


Figure 7.4: (a) Preprocessor A. The preprocessor output, $\breve{\mathbf{X}}$ is subtracted from the original signal, $\mathbf{X}$, and the difference, $\mathbf{E}$, is the input signal to be encoded by ORDE. (b) The reconstruction scheme needed to generate the reconstruction of the original signal, $\tilde{\mathbf{X}}$, based on the ORD decoded signal, $\tilde{\mathbf{E}}$, and the preprocessor bit stream A.

$$\tilde{\mathbf{X}} = \tilde{\mathbf{E}} + \breve{\mathbf{X}}. \tag{7.10}$$

Experimental results from coding ECG signals where preprocessor A is included, are given in Section 7.3.2.

7.2.2 Preprocessor B

Preprocessor B has much of the same properties as preprocessor A. A difference signal, $\mathbf{E} = \mathbf{X} - \breve{\mathbf{X}}$, is generated and used as input signal for the ORDE. The preprocessor generates a separate bit stream, which is called "Bit stream B" when preprocessor B is used. See Figure 7.5(a). Instead of finding the mean value of each signal block, the input signal is down-sampled by a factor P. In this work $P = 8$, half of the signal block lengths. In order to avoid aliasing [41], the signal is low-pass filtered prior to down-sampling. A FIR filter is used. The cut-off frequency is at $1/P$, where the frequency band is defined as $[0, 1.0\rangle$. The filter length is $3P$, and the impulse response is denoted $h(n)$. The down-sampled signal is then DPCM encoded and entropy encoded, resulting in "Bit stream B". This is the representation of the preprocessor signal, thus the last part of the preprocessor in Figure 7.5(a) and the decoder shown in Figure 7.5(b) are identical: The bit stream is entropy decoded and DPCM decoded. Then, the signal is up-sampled by a factor P. Finally, it is filtered by a low-pass filter with impulse response $g(n)$. The filter is symmetric, has a cut-off frequency at 0.5, and a filter length equal to $8P + 1$. Out of this, $\breve{\mathbf{X}}$ is generated, which is a low-passed version of $\mathbf{X}$. As in (7.10), the reconstruction of the original signal is $\breve{\mathbf{X}}$ added to the reconstructed difference signal, $\tilde{\mathbf{E}}$.

There is a striking similarity between preprocessor A and B. In fact, preprocessor B is a generalization of preprocessor A. If $P = N$ and $g(n) = h(n) = \frac{1}{N}[\,1 \;\; 1 \;\; \cdots \;\; 1\,]$, preprocessor B is equal to preprocessor A. We still hold on to the notation "preprocessor A" and "preprocessor B", to distinguish between the extraction of the *mean* of sets of signal samples, and the extraction of a *low pass* signal.

Experimental results from coding ECG signals where preprocessor B is included, are given in Section 7.3.2.

7.3 Experimental results

In this section, the main focus is to get a view of the performance of the Operational Rate-Distortion Encoder (ORDE) on ECG signals. In order to avoid

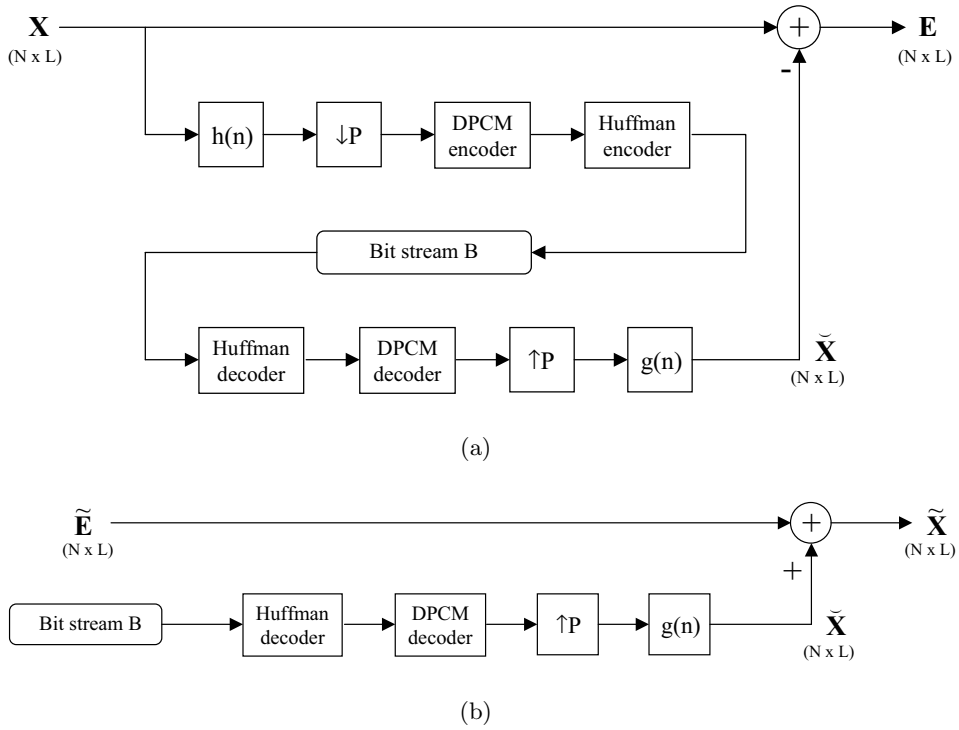


Figure 7.5: (a) Preprocessor B. The preprocessor output, $\breve{\mathbf{X}}$ is subtracted from the original signal, $\mathbf{X}$, and the difference, $\mathbf{E}$, is the input signal to be encoded by ORDE. (b) The reconstruction scheme needed to generate the reconstruction of the original signal, $\tilde{\mathbf{X}}$, based on the ORD decoded signal, $\tilde{\mathbf{E}}$, and the preprocessor bit stream B.

confusion between AR(1) and ECG experiments, we continue the numbering of experiments from the previous chapter. The objectives of the experiments in this chapter are:

Experiment no. 6:	ORDE rate-distortion performance, with and without quantization step size optimization in the training scheme.
Experiment no. 7:	ORDE performance when preprocessing the input signal by preprocessor A and preprocessor B.
Experiment no. 8:	Comparisons of ORDE and RD optimized Matching Pursuit with ECG signals as inputs.

In all experiments of this section, ECG signals are used as input *training* and input *test* signals. The ECG signals used are signals from the MIT arrhythmia database [31]. The records are represented by 12 bits per sample, and the sampling frequency is 360 Hz. Five different signals are used, and short segments of these are presented in Appendix B. MIT100 is an ECG of a healthy human. MIT103, MIT113, MIT207, and MIT217 are ECG signals of hearts with abnormal rhythms. Three different training signals and two different test signals are used:

Training signals:	
MIT100_{train}	4 minutes and 40 seconds of MIT100, from 0:00 to 4:40 minutes. $L = 6300$.
MIT207_{train}	4 minutes and 40 seconds of MIT207, from 6:00 to 10:40 minutes. $L = 6300$.
MITmix_{train}	5 minutes, consisting of five segments of one minute length from MIT100, MIT103, MIT113, MIT207, and MIT217, respectively. The segments are MIT100 0:00 to 1:00 minutes, MIT103 3:25 to 4:25 minutes, MIT113 0:00 to 1:00 minutes, MIT207 6:00 to 7:00 minutes, and MIT217 0:00 to 1:00 minutes. $L = 6750$.

Test signals:	
MIT100_{test}	4 minutes and 40 seconds of MIT100, from 5:30 to 10:10 minutes. $L = 6300$.
MIT207_{test}	4 minutes and 40 seconds of MIT207, from 12:00 to 16:40 minutes. $L = 6300$.

In all experiments, $N = 16$, $K = 32$, $J = 32$, $M_{max} = 4$, and Frame 3 is the initial frame.

7.3.1 Experiment no. 6

In this experiment, the effects of using quantization step size (Δ) optimization is presented. When Δ-optimization is used, it is included in three loops in the training scheme, as presented in Figure 7.1. Due to the optimality in the way we solve the minimization problem in (7.6), the new value of Δ will guarantee a cost that is lower or equal to the current cost, if it is used in ORDE in the next iteration. Figure 7.6, 7.7, and 7.8 show the training results when MIT100_{train}, MIT207_{train} and MITmix_{train} are input signals, respectively. In all three diagrams, the circled line is the learning curve where Δ-optimization is *not* included, while the line with triangles is the learning curve where Δ-optimization *is* included in the training scheme. In Figure 7.6, the learning curves are almost equal. In the case where Δ-optimization is added, the big drop in cost is one iteration prior to the other case. The big drop is, as it was for AR(1) signals, at the iteration when Loop 2 is entered and VLC table optimization is started. It is not shown in the diagram, but Loop 1.5 is iterated only once. It is the last iteration prior to the big drop, iteration no. 25. The cost at the final iteration in the case where Δ-optimization is not used, is close to the cost of the other case. The reason is that the initial value of Δ is close to the optimum value, when MIT100_{train} is input signal. The value of Δ change from 0.0625 to 0.0639.

When MIT207_{train} is the input signal, the situation is different, as shown in Figure 7.7. Initially $\Delta = 0.0625$, but at the last iteration $\Delta = 0.0700$. The cost in the final iteration is 4 % less when Δ-optimization is included. In the same diagram, we can see which loop each iteration in the Δ-optimization curve belongs to. The first five iterations are in Loop 1, and the eight next belong to Loop 1.5. Iteration 14 is the first in Loop 2, and this is where we have the big drop in the cost. Figure 7.8 shows the learning curves when MITmix_{train} is input signal. Here, quantization step size optimization gives larger improvements to the final results. The cost in the final iteration is 13 % less when Δ-optimization is included, compared to the situation where Δ is not adjusted. The value of Δ is changed from 0.0625 to 0.0735.

The results of coding the MIT100_{test} signal is shown in the rate versus SNR plots in Figure 7.9. The frame and VLC tables were trained on (a) MIT100_{train} and (b) MITmix_{train}, respectively. The curves with triangles show the results where the trained Δ is used, and the curves with circles use a fixed Δ-value, the same as the initial value, $\Delta = 0.0625$. In both plots, the case where a trained Δ is used is better than the other. The difference is largest in Figure 7.9(b), where the MITmix_{train} was the training input signal. At bit rates between 0.45 and 0.55 bit/sample, the results when optimizing Δ are between 0.6 dB

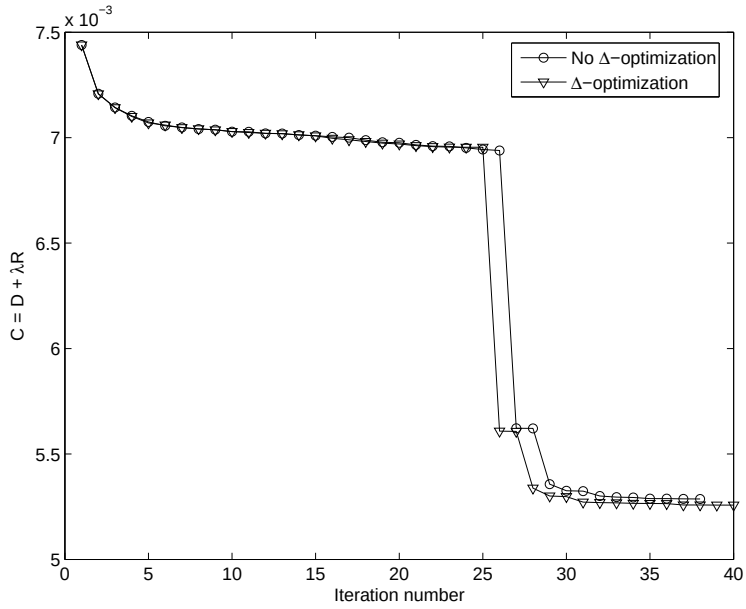


Figure 7.6: The learning curve from the training procedure with MIT100$_{train}$ as input signal.

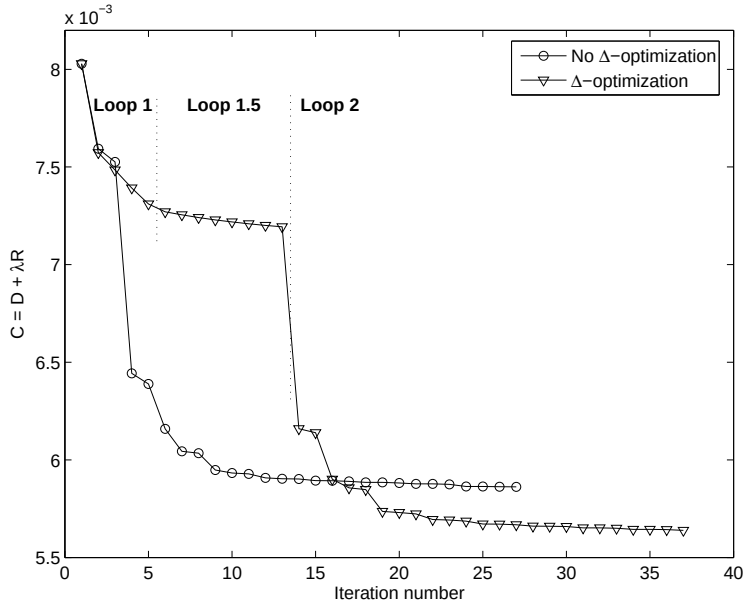


Figure 7.7: The learning curve from the training procedure with MIT207$_{train}$ as input signal.

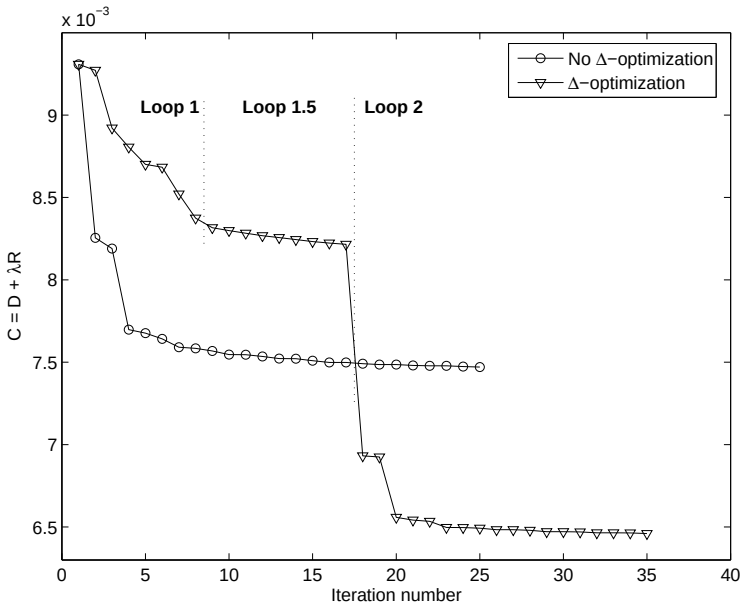


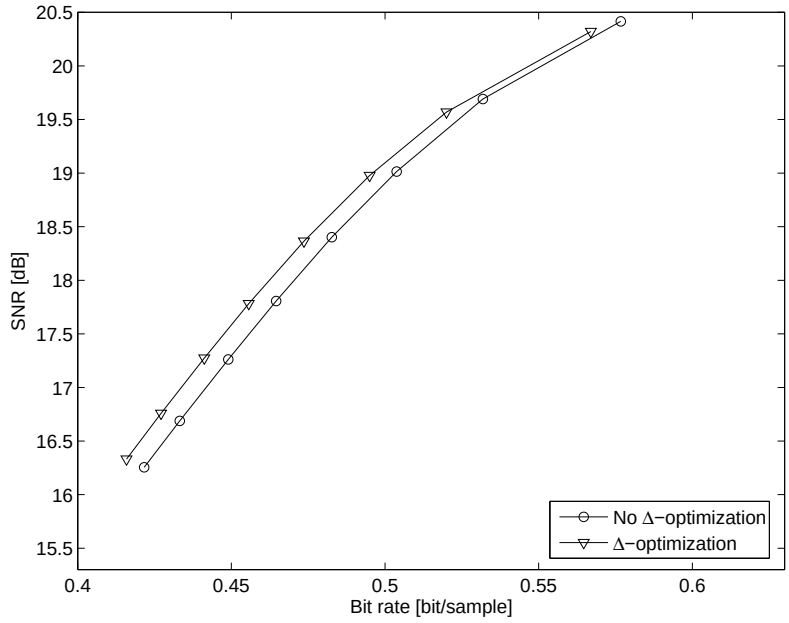
Figure 7.8: The learning curve from the training procedure with MITmix$_{train}$ as input signal.

and 1 dB better, while in the MIT100$_{train}$ case, the improvements at the same bit rates are between 0.1 dB and 0.3 dB. This is no surprise, due to the large differences in the learning curves in Figure 7.8, compared to the small differences in Figure 7.6.

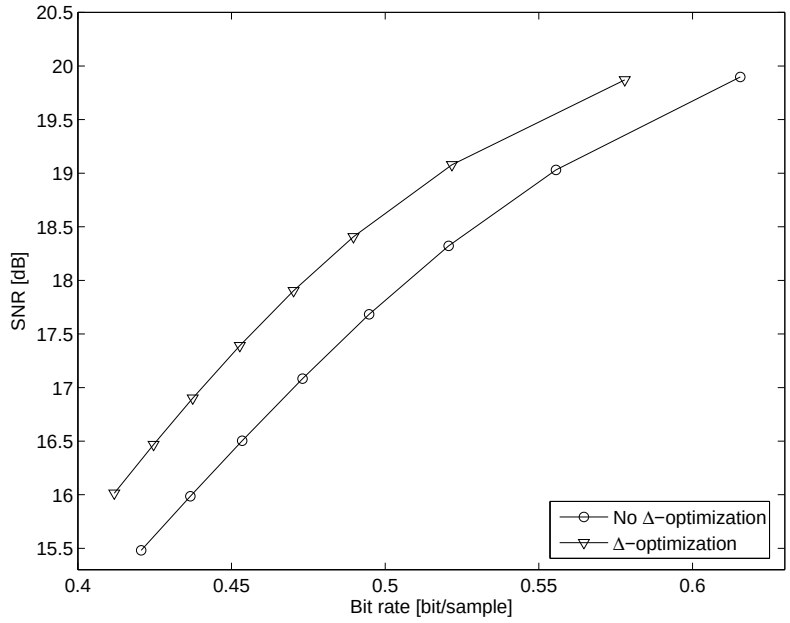
In Figure 7.10 the results of coding the MIT207$_{test}$ signal is shown. The frame and VLC tables were trained on (a) MIT207$_{train}$ and (b) MITmix$_{train}$, respectively. The differences are larger for this signal, compared to the MIT100$_{test}$ signal. For bit rates between 0.55 and 0.65 bit/sample, the result improvements of using Δ-optimization are as large as 1.2 to 1.4 dB in Figure 7.10(a) and 1.9 to 2.3 dB in Figure 7.10(b). This illustrates the importance of including quantization step size optimization in the training loops.

7.3.2 Experiment no. 7

In this experiment, results from preprocessing input signals are presented. Both Preprocessor A and B, introduced in Section 7.2, are tested. In Figure 7.11, a set of 900 samples from MIT100$_{test}$ is shown. The original signal, $\mathbf{X}$, is the dashed line in Figure 7.11(a) and (c). In the same diagrams, the

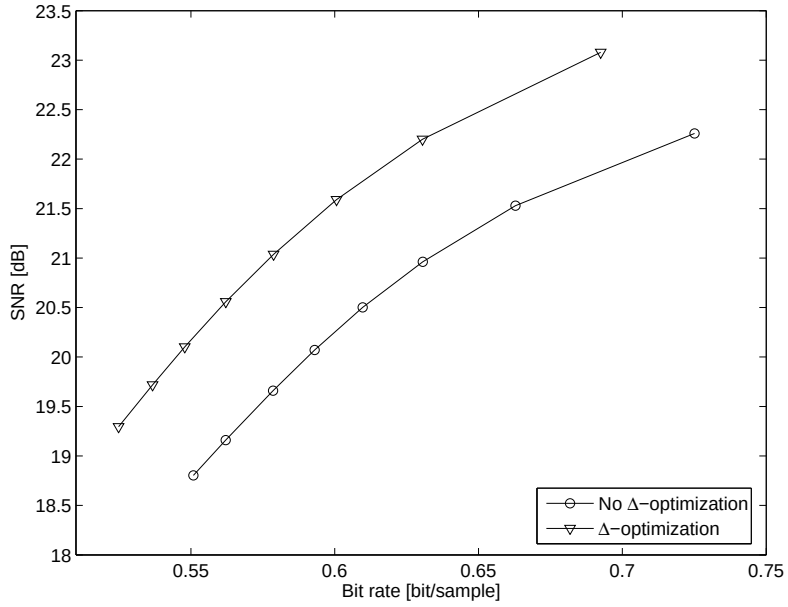


(a)

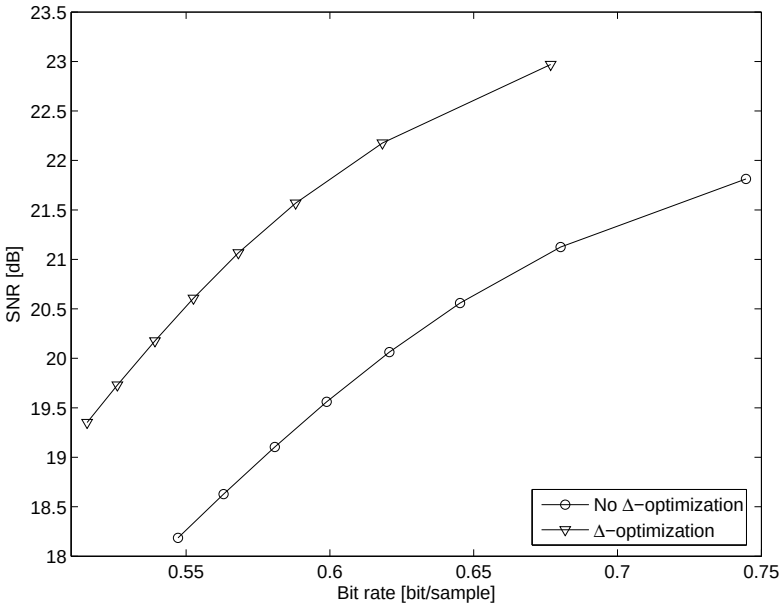


(b)

Figure 7.9: The rate versus SNR results when MIT100$_{test}$ are encoded. The input signal in the training procedure is (a) MIT100$_{train}$ and (b) MITmix$_{train}$.



(a)



(b)

Figure 7.10: The rate versus SNR results when MIT207$_{test}$ are encoded. The input signal in the training procedure is (a) MIT207$_{train}$ and (b) MITmix$_{train}$.

solid line is the preprocessor output signal, $\check{\mathbf{X}}$, from Preprocessor A and Preprocessor B, respectively. In both preprocessors, the DPCM coder has 16 quantization steps. In Figure 7.11(a), we can see the square-shaped output signal from Preprocessor A, where each plateau is the mean value of the corresponding 16 samples from the original signal. The difference signal, $\mathbf{E}$, which is the input signal to the ORDE, is shown in Figure 7.11(b). There are still peaks in this signal, but their absolute magnitudes are smaller. In Figure 7.11(c), we can see the output signal from Preprocessor B follows the original signal better than in the case mentioned above. It is a low pass filtered version of the original signal. This results in a difference signal with less energy, as shown in Figure 7.11(b). In Preprocessor B, the down- and up-sampling factor is 8, the anti-aliasing filter length is 24, and the interpolation filter length is 65.

A sample set of MIT207$_{test}$ is shown in Figure 7.12. As the original signal in Figure 7.12(a) and (c) indicates, this is an ECG of an abnormal heart rhythm. All encoder settings are identical to the MIT100$_{test}$ signal case. The difference signals when using Preprocessor A and B are presented in Figure 7.12(b) and Figure 7.12(d), respectively. For the MIT207$_{test}$ signal case, we can see that the difference signal when using Preprocessor B, is considerably reduced. This should lead to ORDE results with low distortion at low rates.

MIT100$_{test}$ and MIT207$_{test}$ are encoded and decoded at the given settings. For both cases, the training input signal is MITmix$_{train}$, and Δ is optimized. The final test results of the same set of samples shown in Figure 7.11 and Figure 7.12, are presented in Figure 7.13 and Figure 7.14, respectively. For both figures, (a) is the original signal, (b) is the reconstructed signal where no preprocessing is used, (c) is the reconstructed signal where Preprocessor A is used, and (d) is the reconstructed signal where Preprocessor B is used. Going from diagram (b) to (d) in both figures, the curves get more and more smooth. It is no surprise that the distortion is decreasing when going from diagram (b),(c) to (d), but the bit rates are at the same level.

When using a preprocessor, some of the signal information is transmitted, or stored, in the preprocessor bit stream. The bit rates of the ORDE and the preprocessor for the MIT100$_{test}$ signal at SNR values between 20.6 and 21.0 dB is given in Table 7.1. The total bit rate is the bit rate of the ORDE output added to the preprocessor bit rate. The frame and VLC tables used in the test are trained on MITmix$_{train}$. Three situations are presented: When no preprocessing is used, when Preprocessor A is used, and when Preprocessor B is used. For the preprocessor cases, three different numbers of quantization steps, κ, in the DPCM encoder are used: $\kappa = \{4, 8, 16\}$. If κ is small, the number of bits in the preprocessor bit stream will be small, and the preprocessor output signal, $\check{\mathbf{X}}$, will be less accurate. If we want to keep the SNR

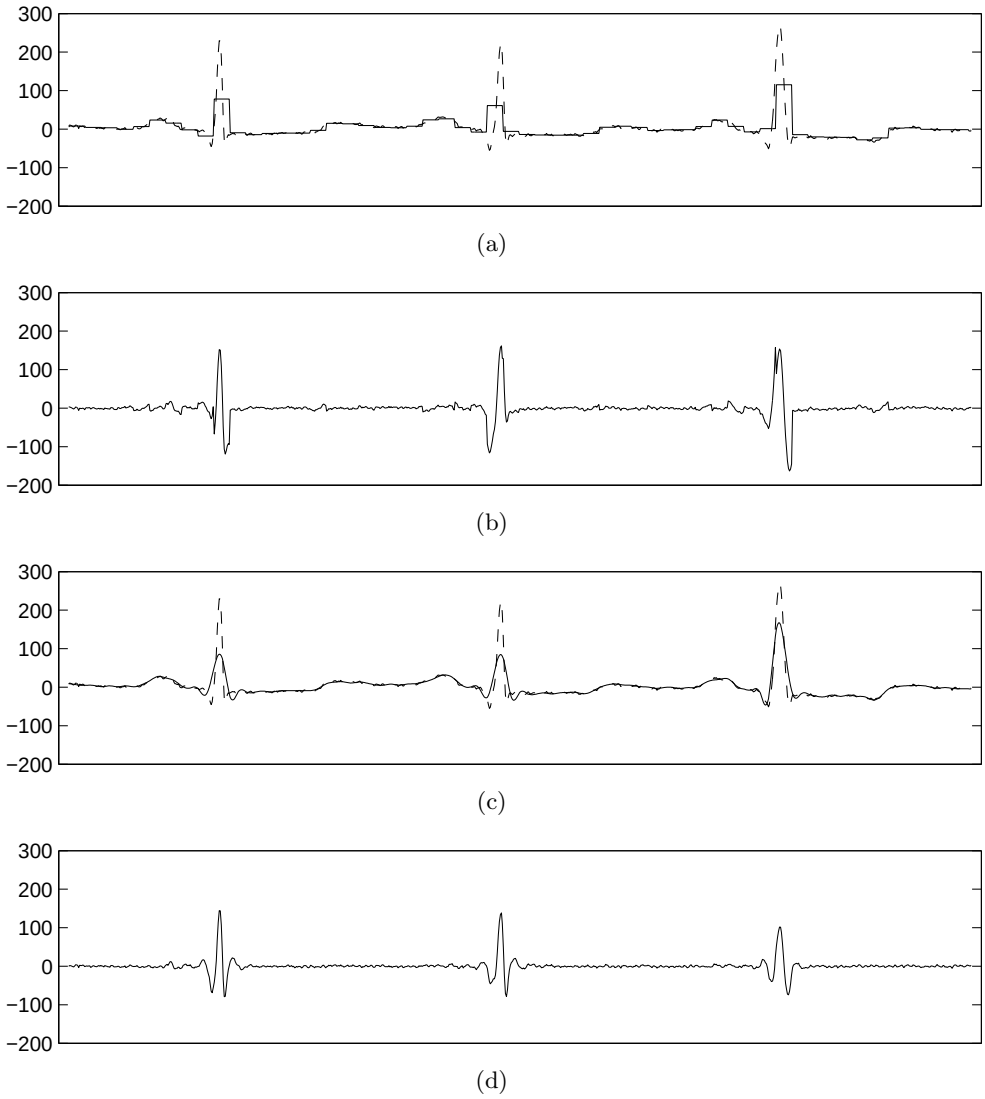


Figure 7.11: MIT100$_{test}$, sample no. 50400 to 51300. Original signal samples as dotted lines in (a) and in (c). The solid lines are the corresponding samples of: Preprocessor A (a) reconstruction signal ($\breve{\mathbf{X}}$), and (b) difference signal ($\mathbf{E}$). Preprocessor B (c) reconstruction signal ($\breve{\mathbf{X}}$), and (d) difference signal ($\mathbf{E}$).

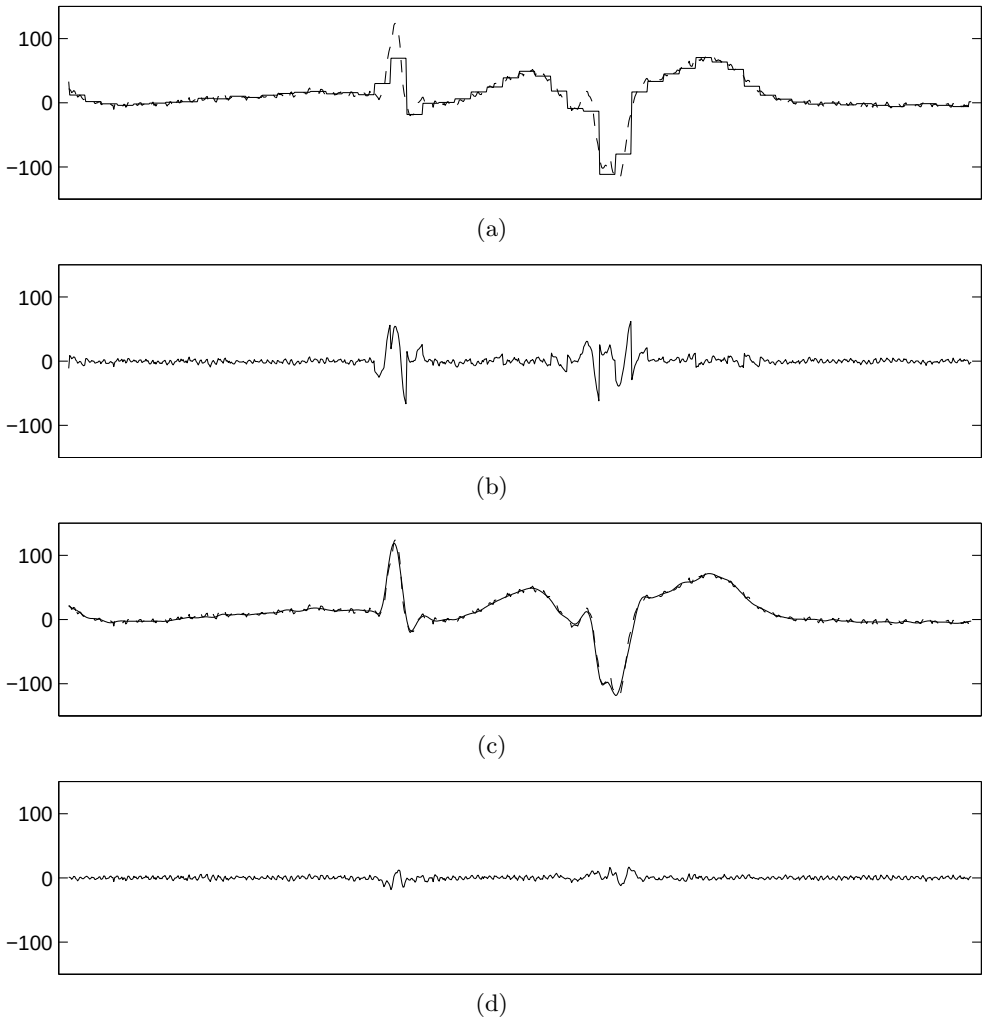
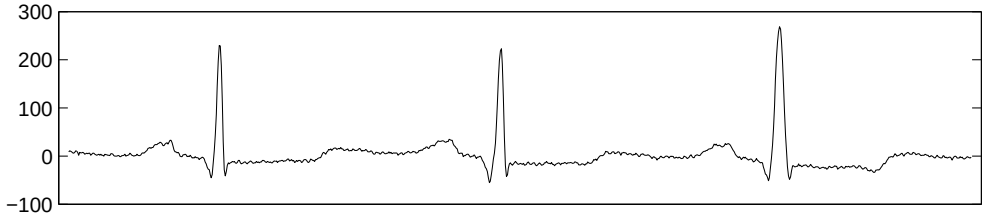
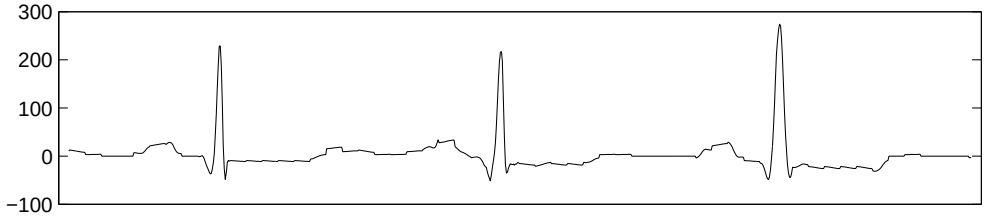


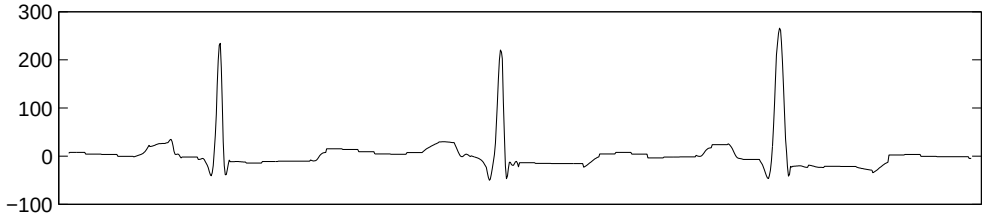
Figure 7.12: MIT207$_{test}$, sample no. 50400 to 51300. Original signal samples as dotted lines in (a) and in (c). The solid lines are the corresponding samples of: Preprocessor A (a) reconstruction signal ($\breve{\mathbf{X}}$), and (b) difference signal ($\mathbf{E}$). Preprocessor B (c) reconstruction signal ($\breve{\mathbf{X}}$), and (d) difference signal ($\mathbf{E}$).



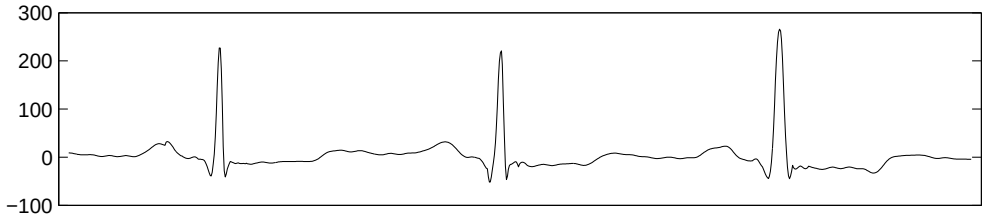
(a) Original signal, 12 bit/sample



(b) No preprocessing, 0.77 bit/sample, SNR = 21.46 dB

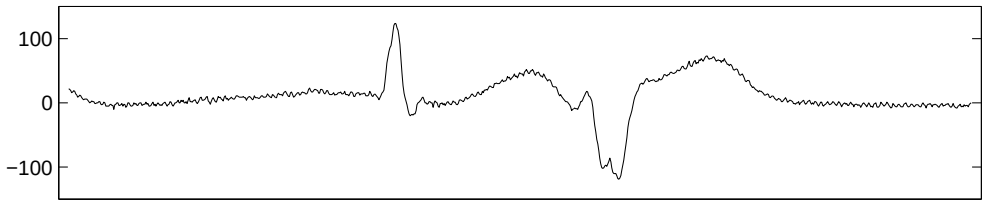


(c) Preprocessor A, 0.79 bit/sample, SNR = 22.70 dB

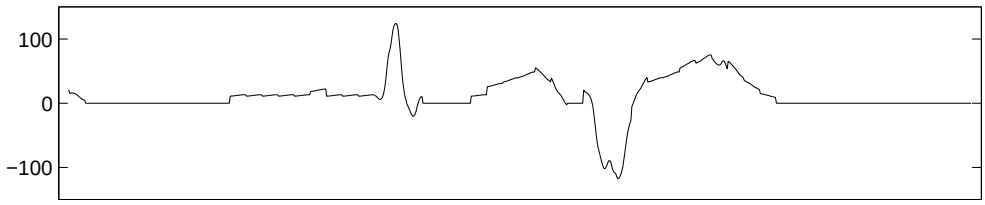


(d) Preprocessor B, 0.80 bit/sample, SNR = 22.96 dB

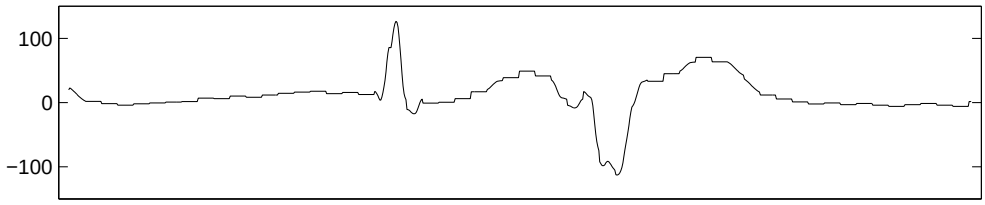
Figure 7.13: MIT100$_{test}$, sample no. 50400 to 51300. Corresponding samples from (a) the original signal, the reconstructed signals (b) with no preprocessing, (c) when Preprocessor A is used, and (d) when Preprocessor B is used.



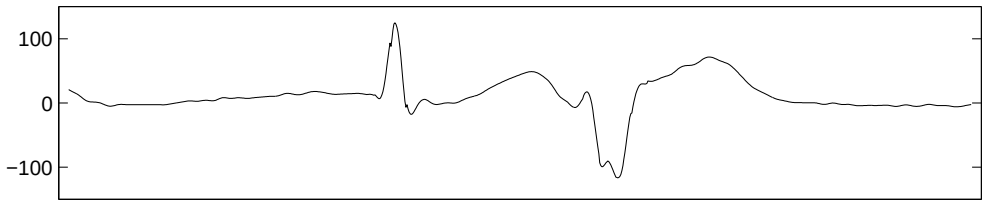
(a) Original signal, 12 bit/sample



(b) No preprocessing, 0.68 bit/sample, SNR = 22.97 dB



(c) Preprocessor A, 0.66 bit/sample, SNR = 25.56 dB



(d) Preprocessor B, 0.68 bit/sample, SNR = 26.42 dB

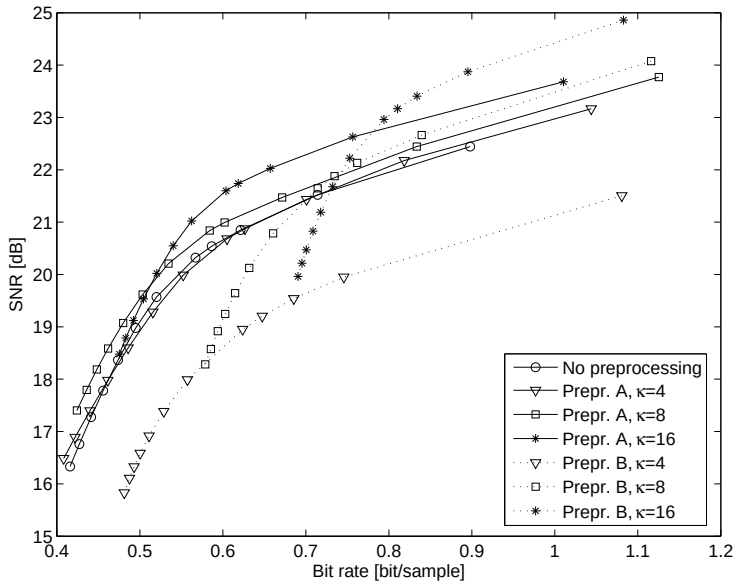
Figure 7.14: MIT207$_{test}$, sample no. 50400 to 51300. Corresponding samples from (a) the original signal, the reconstructed signals (b) with no preprocessing, (c) when Preprocessor A is used, and (d) when Preprocessor B is used.

value at same level, the number of bits in the ORDE output bit stream will be larger for a small κ. The preprocessor bit rates are approximately doubled when using preprocessor B, compared to using preprocessor A. The ORDE bit rates when using preprocessor B at $\kappa = \{8, 16\}$ are less than when using preprocessor A, but if we look at the total bit rates, preprocessor A is best at this level of SNR values. By using preprocessor A at $\kappa = 16$, we get the best SNR value and the best bit rate, 20.99 dB at 0.59 bit/sample.

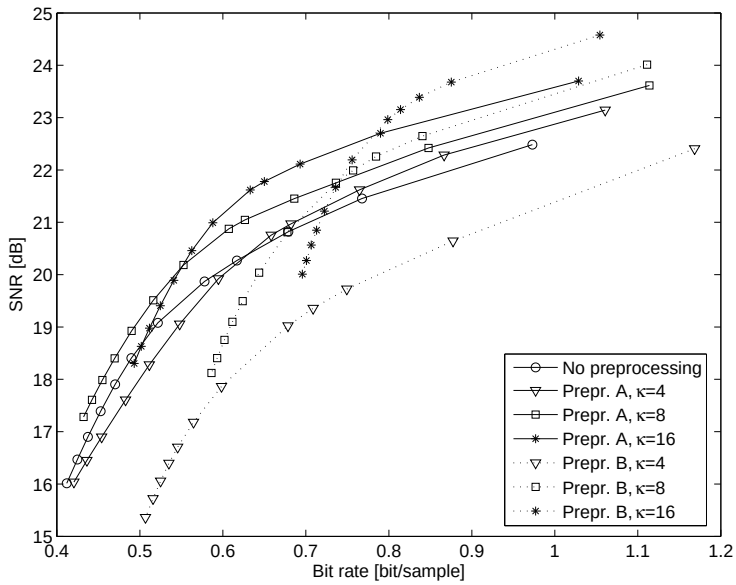
	ORDE rate [bit/sample]	Prepr. rate [bit/sample]	Total rate [bit/sample]	SNR [dB]
No preprocessing	*0.68*	-	0.68	20.82
Prepr. A, $\kappa = 4$	*0.53*	*0.13*	0.66	20.76
$\kappa = 8$	*0.43*	*0.18*	0.61	20.88
$\kappa = 16$	*0.35*	*0.24*	0.59	20.99
Prepr. B, $\kappa = 4$	*0.64*	*0.24*	0.88	20.64
$\kappa = 8$	*0.31*	*0.37*	0.68	20.81
$\kappa = 16$	*0.23*	*0.48*	0.71	20.85

Table 7.1: Bit rates at approximately same SNR value, where total bit rate is ORDE bit rate added to preprocessor bit rate. Test signal is MIT100_{test} when frame and VLC tables are trained on MITmix_{train}. Three situations are compared: Where no preprocessing is used, where Preprocessor A is used, and where Preprocessor B is used. For the preprocessor cases, three different numbers of quantization steps are tested: $\kappa = \{4, 8, 16\}$.

In order to get a view of the rate-distortion performance of using preprocessing, rate versus SNR plots from the testing where MIT100_{test} and MIT207_{test} are input signals are presented in Figure 7.15 and Figure 7.16, respectively. In Figure 7.15(a) and Figure 7.16(a), the frame and VLC tables are trained on MIT100_{train} and MIT207_{train}, respectively. In Figure 7.15(b) and Figure 7.16(b), MITmix_{train} was training signal. The seven curves in all four diagrams are: When no preprocessing is used, when Preprocessor A is used at $\kappa = \{4, 8, 16\}$, and when Preprocessor B is used at $\kappa = \{4, 8, 16\}$. For the MIT100_{test} signal, using preprocessor A at $\kappa = 8$ or 16 give best results at bit rates less than 0.78 bit/sample. For higher rates, the best result is obtained by using preprocessor B at $\kappa = 16$, regardless of the training signal used. For the MIT207_{test} signal the worst rate-distortion results are obtained when no preprocessing is used. The best results for bit rates higher than 0.6 bit/sample we get when using preprocessor B at $\kappa = 8$ or 16.

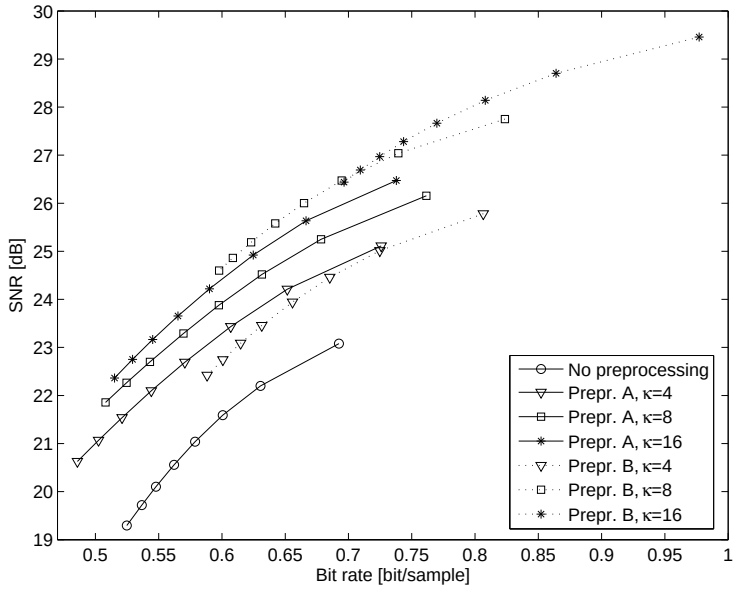


(a)

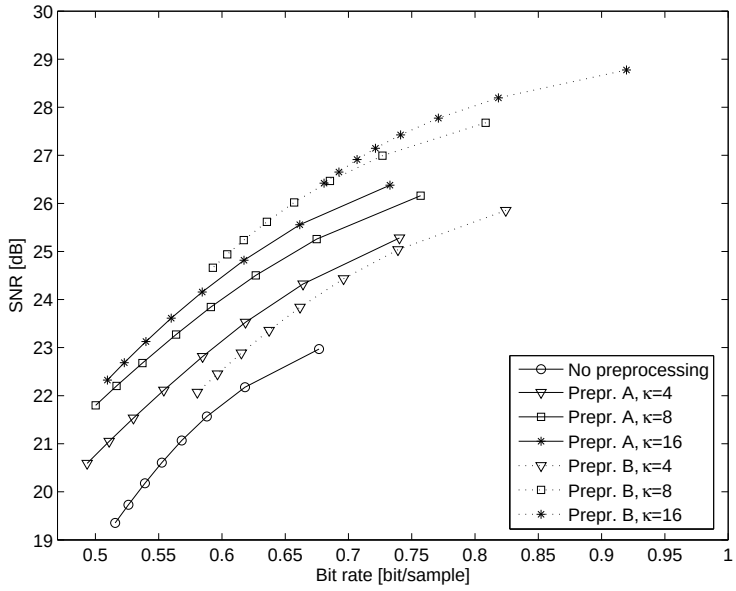


(b)

Figure 7.15: Bit rate versus SNR results from the encoding of MIT100$_{test}$, where Preprocessor A or Preprocessor B, or no preprocessor is used at all. The input signal in the training procedure is (a) MIT100$_{train}$ and (b) MITmix$_{train}$.



(a)



(b)

Figure 7.16: Bit rate versus SNR results from the encoding of MIT207$_{test}$, where Preprocessor A or Preprocessor B, or no preprocessor is used at all. The input signal in the training procedure is (a) MIT207$_{train}$ and (b) MITmix$_{train}$.

7.3.3 Experiment no. 8

This experiment compares four different coders on ECG signals: ORDE with *index* codewords (ORDE *ind*), ORDE with *run* codewords (ORDE *run*), Rate-Distortion Optimized Basic Matching Pursuit (RDO BMP), and Rate-Distortion Optimized Order Recursive Matching Pursuit (RDO ORMP). In this experiment Preprocessor A with $\kappa = 16$ is used prior to the encoding, regardless of which of the four coders that is used. The initial frame in the training scheme is Frame 3, the training of frame, VLC tables and Δ has been done at $\lambda = 0.0005$, $J = 32$, and $M_{max} = 4$ for all four coders. The test results of MIT100$_{test}$ as input signal are presented as rate versus SNR plots in Figure 7.17, where (a) MIT100$_{train}$ and (b) MITmix$_{train}$ were training input signals, respectively. In both cases ORDE is better than the RDO Matching Pursuit algorithms for the entire actual bit rate range. At a bit rate of 0.6 bit/sample, the ORDE algorithms are approximately 1.5 dB better than the Matching Pursuit algorithms in both plots.

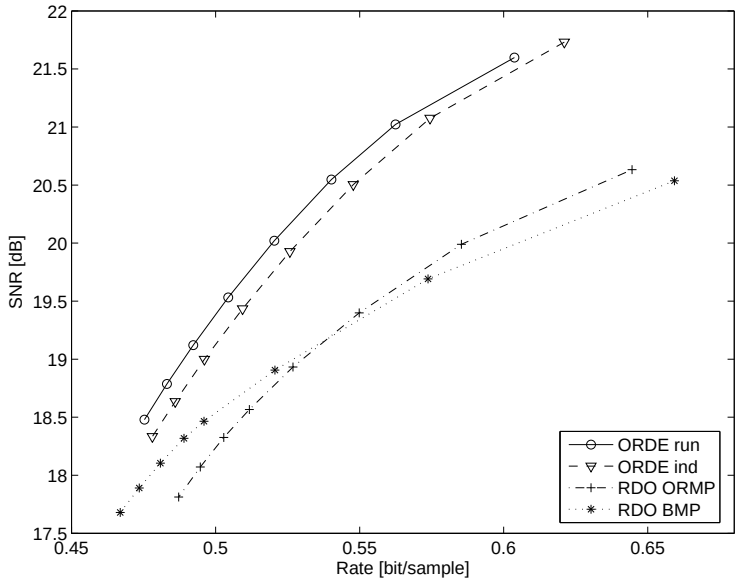
Figure 7.18 shows the test results of MIT207$_{test}$ as input signal, trained on (a) MIT207$_{train}$ and (b) MITmix$_{train}$, respectively. ORDE is better than RDO Matching Pursuit algorithms for this signal, as well. In Figure 7.18(a) RDO BMP is better than RDO ORMP. At 0.65 bit/sample ORDE *run* is 0.6 dB better than RDO BMP. In Figure 7.18(b) the ORDE algorithms are more than 1 dB better than RDO BMP at 0.65 bit/sample. It is interesting to see that ORDE *run* is better than ORDE *ind* for all bit rates in Figure 7.17 and Figure 7.18. This is a demonstration of the advantage of using run-length coding: Even though the set of solutions is less, the bit rates are lower, that all in all results in a reduced cost.

7.4 Summary

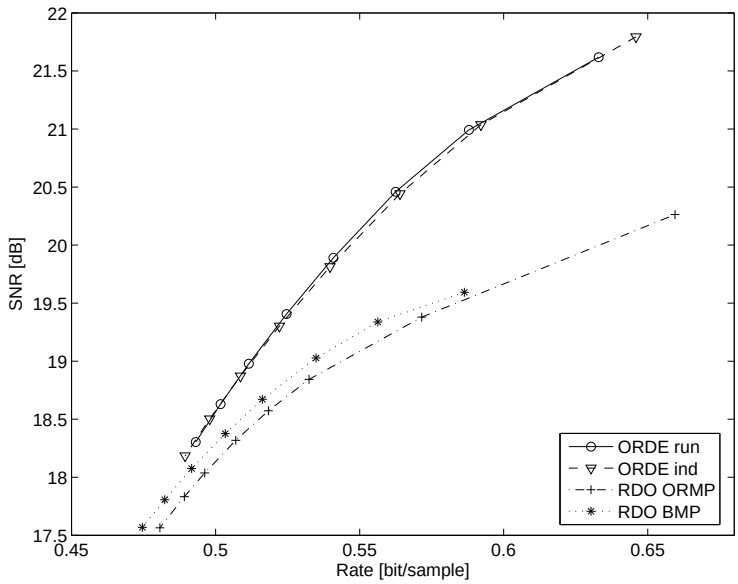
The experiments on ECG signals can be summarized as follows: (1) In addition to the training of frame and VLC tables, it is important to optimize the quantization steps between the representation values in the *value* codeword table. (2) It is preferable to preprocess signals with large dynamic variations. (3) The ORDE algorithms are more time consuming, but outperforms RDO MP algorithms when encoding ECG signals.

For each of the three experiments presented, we can summarize as follows:

Experiment no. 6:
The introduction of Δ-optimization in the training scheme resulted in a clear

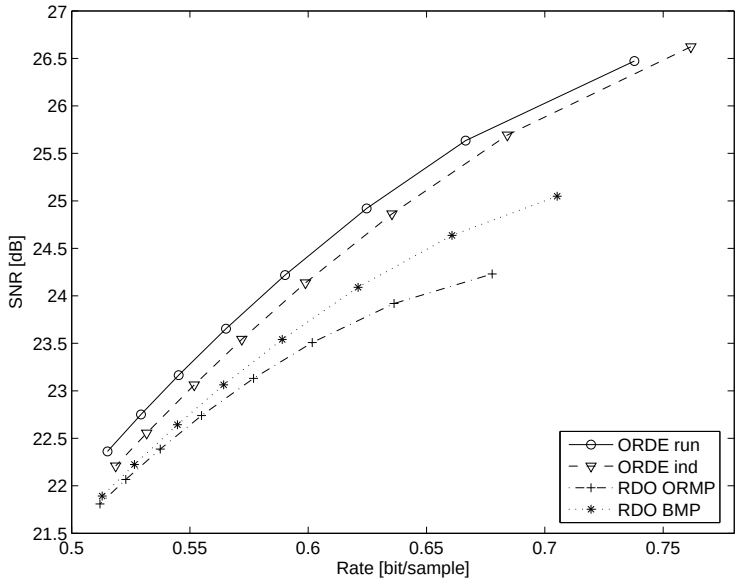


(a)

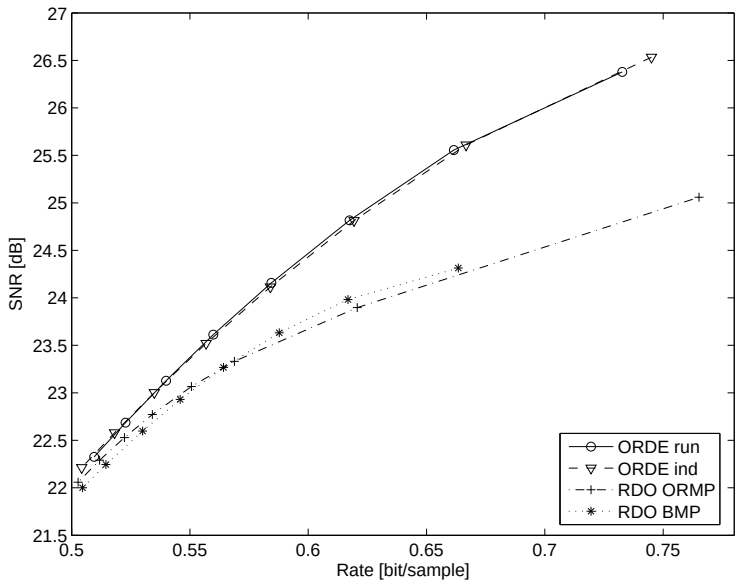


(b)

Figure 7.17: MIT100. Comparisons of four different algorithms in the rate-SNR sense. The input signal in the training procedure is (a) MIT100_{train} and (b) MITmix_{train}.



(a)



(b)

Figure 7.18: MIT207. Comparisons of four different algorithms in the rate-SNR sense. The input signal in the training procedure is (a) MIT207$_{train}$ and (b) MITmix$_{train}$.

improvement of the RD performance in the experiments presented, particularly for the test case where MIT207$_{test}$ is input signal. In general, the number of iterations in the training scheme increases, but the cost is reduced.

Experiment no. 7:
Preprocessing of the ECG signal results in an ORDE input signal with less dynamic variations. Thus, the overall cost will be reduced. Results show that preprocessing gives better rate-distortion results. For low bit rates, preprocessor A with $\kappa = 8$ would be to prefer. For higher bit rates (> 0.8 bit/sample) preprocessor B with $\kappa = 16$ gives best results.

Experiment no. 8:
The ORDE algorithms outperforms the RDO MP algorithms. The benefits of using run-length coding is demonstrated in this experiment: ORDE *run* is better than ORDE *ind* at all bit rates. ORDE *run* is less time consuming than ORDE *ind*, particularly when M_{max} is large.

Chapter 8

Conclusions and Summary

In this thesis, the main focus has been the development of an efficient method for rate-distortion optimal frame based coding. The efficiency is achieved by: a) using a QR-decomposition which results in a new set of independent decision variables; b) by choosing the right search strategy; and c) by using ordered vector selection that allows the use of run-length encoding. The overall complexity is reduced by a factor of J^{M-1}, where J is the number of different *values* the representing coefficients can take on, and M is the number of nonzero coefficients per signal block. This is a tremendous complexity reduction.

We have demonstrated that it is possible to find the RD optimal solution to the vector selection problem, within a reasonable amount of time. The problem is still NP-hard, and compared to Matching Pursuit or traditional transform coding, the encoding can be much more time consuming. But, this is asymmetric coding: Even though encoding takes time, the decoding of the signal is done very fast. The ORDE algorithm in this work generates an important quality measurement: *The rate-distortion optimal solution of the problem that the RDO ORMP algorithm addresses.* If the Lagrangian multiplier is zero ($\lambda = 0$), our approach finds the optimal solution to the ORMP problem, when the values the quantized coefficient can take on, are known in advance to vector selection.

In order to find a best possible compression result for a given signal, it is important to train the frame and the VLC tables using a training signal from the same class of signals. Experiments on AR(1) signals show that the ORDE performs better with a well trained frame. The shape of the initial frame is less significant. The most important contribution in the cost reduction of all presented learning curves, is caused by the optimization of the coefficient *value*

and *index/run* codeword bit lengths. In the coding of ECG signals, another parameter determining the entries of the *value* codewords table is optimized: The quantization step size, Δ. Since the number of *value* codewords is fixed, Δ will set the range of coefficient values. As experiments on ECG signals show, Δ-optimization is important if we do not want an in advance "trial-and-error" testing in order to find a good value of Δ.

Finally, it is demonstrated through AR(1) and ECG signals experiments, that ORDE outperforms RDO MP in the rate-distortion sense. ORDE with run-length coding (ORDE *run*) is better than ORDE with *index* coding (ORDE *ind*) in the experiments presented, even though ORDE *run* has a smaller solution set. I.e., ORDE *run* shows better performance than ORDE *ind*, both in consumption of time and in the rate-distortion sense.

8.1 Directions for future research

This work has given insight in a new field of operational rate-distortion optimal coding. Based on this insight, suggestions for future work is as follows:

- *Various parameter optimizations*
  In addition to the *value* symbol step size (Δ) optimization, more research should have been done to find the optimal value of other parameters, in the rate-distortion sense. Relevant parameters are: The number of *value* and *index/run* symbols, the number of frame vectors, and the number of samples per signal block.

- *Training scheme refinements*
  In this thesis, training Loop 1 terminates at the first iteration where the cost is no longer decreasing. Other termination criteria could have been considered. *value* VLC table optimization might be considered for inclusion in Loop 1, in addition to Loop 2. In Loop 2, a more intelligent, but still convergent, frame update algorithm could have been found.

- *Rate-distortion optimized partial search*
  While ORDE search in the entire solution tree in order to find the optimal solution to our problem, RDO ORMP follows a one path depth-first-search strategy in order to find a good sub-optimal solution. A compromise between these two search strategies is partial search. In contrast to MP, where only the best node in each level is selected, the B best nodes are picked out. In the next level, there are more than one path to be investigated, and a subtree is built. There can be different

numbers of selected nodes in each level. Let B_m be this number for level m, where $m = 1, \ldots, M_{max}$. The number of nodes to be investigated would be $1 + \sum_m \prod_m (B_m)$. Partial search is slower than RDO ORMP, but will generate a better or equal rate-distortion solution. The idea of partial search in vector selection has previously been used in [10] and [55] in the minimization of distortion. In the RDO sense, it has partly been investigated in [28].

Appendix A

Mathematical details

In this appendix we go deeper into some of the linear algebra terms mentioned in Chapter 4. All the material of this appendix is taken from the book of elementary linear algebra by Anton [3].

A.1 QR-decomposition by Gram-Schmidt

Let $\mathbf{U} = [\ \mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_M\ ]$ be an arbitrary $N \times M$ ($M \leq N$) matrix with linearly independent column vectors. A set of *orthonormalized* column vectors, $\mathbf{Q} = [\ \mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_M\ ]$ can be found from $\mathbf{U}$ by using the *Gram-Schmidt process* shown in Figure A.1. In line 4 the term $\langle \mathbf{u}_m, \mathbf{q}_j \rangle$ is the *inner product* between the two vectors, $\mathbf{u}_m$ and $\mathbf{q}_j$. If both are vectors in $\mathbb{R}^N$, $\langle \mathbf{u}_m, \mathbf{q}_j \rangle = \mathbf{u}_m^T \mathbf{q}_j$, which is the *Euclidean inner product*.

The *QR-decomposition* of $\mathbf{U}$ is defined as

$$\mathbf{U} = \mathbf{QR}, \tag{A.1}$$

where $\mathbf{Q}$ is the $N \times M$ matrix with orthonormal column vectors found by the Gram-Schmidt process, and $\mathbf{R}$ is an $M \times M$ invertible upper triangular matrix. The nonzero elements of $\mathbf{R}$ are all inner products of column vectors in $\mathbf{U}$ and column vectors in $\mathbf{Q}$ in the following way:

$$\mathbf{R} = \begin{bmatrix} \langle \mathbf{u}_1, \mathbf{q}_1 \rangle & \langle \mathbf{u}_2, \mathbf{q}_1 \rangle & \cdots & \langle \mathbf{u}_M, \mathbf{q}_1 \rangle \\ 0 & \langle \mathbf{u}_2, \mathbf{q}_2 \rangle & \cdots & \langle \mathbf{u}_M, \mathbf{q}_2 \rangle \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \langle \mathbf{u}_M, \mathbf{q}_M \rangle \end{bmatrix} \tag{A.2}$$

Gram-Schmidt process

input : $\mathbf{U}$ // *$N \times M$ matrix with linearly independent columns.*
output: $\mathbf{Q}$ // *$N \times M$ matrix with orthonormal columns.*

$\mathbf{U} = [\, \mathbf{u}_1 \;\; \mathbf{u}_2 \;\; \cdots \;\; \mathbf{u}_M \,]$
$\mathbf{q}_1 \leftarrow \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}$
for $m = 2$ **to** M **do**
    $\mathbf{q}_m \leftarrow \frac{\mathbf{u}_m - \sum_{\xi=1}^{m-1} \langle \mathbf{u}_m, \mathbf{q}_\xi \rangle \mathbf{q}_\xi}{\|\mathbf{u}_m - \sum_{\xi=1}^{m-1} \langle \mathbf{u}_m, \mathbf{q}_\xi \rangle \mathbf{q}_\xi\|}$
end
$\mathbf{Q} = [\, \mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \cdots \;\; \mathbf{q}_M \,]$

Figure A.1: The Gram-Schmidt process.

A.2 Best Approximation Theorem

Best approximation theorem: If U is a finite-dimensional subspace of an inner product space Z, and if $\mathbf{z}$ is a vector in Z, then $\text{proj}_U \mathbf{z}$ is the *best approximation* to $\mathbf{z}$ from U in the sense that

$$\|\mathbf{z} - \text{proj}_U \mathbf{z}\| < \|\mathbf{z} - \mathbf{u}\| \tag{A.3}$$

for every vector $\mathbf{u}$ in U that is different from $\text{proj}_U \mathbf{z}$.

In other words, among all vectors $\mathbf{u}$ in U the vector $\mathbf{u} = \text{proj}_U \mathbf{z}$ minimizes the distance $\|\mathbf{z} - \mathbf{u}\|$. In Figure A.2, we can see that the vector $\mathbf{z} - \text{proj}_U \mathbf{z}$ is orthogonal to the subspace U.

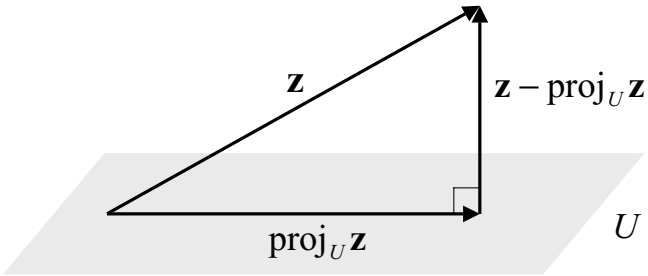


Figure A.2: The vector $\mathbf{z} - \text{proj}_U \mathbf{z}$ is orthogonal to the subspace U.

Let $\mathbf{A}$ be an $N \times M$ matrix with linearly independent column vectors that span U. The product $\mathbf{Au}$ is a linear combination of the column vectors in $\mathbf{A}$. Now, the orthogonal projection of $\mathbf{z}$ on U is

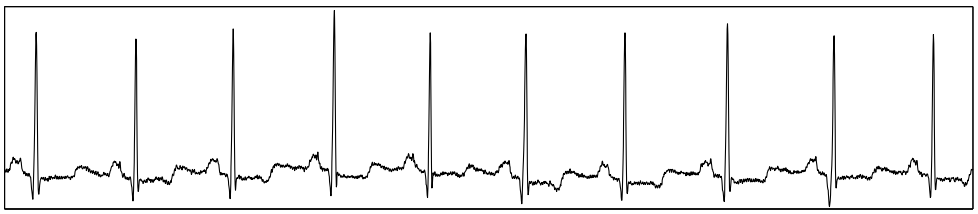
$$\text{proj}_U \mathbf{z} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{z}. \tag{A.4}$$

Appendix B

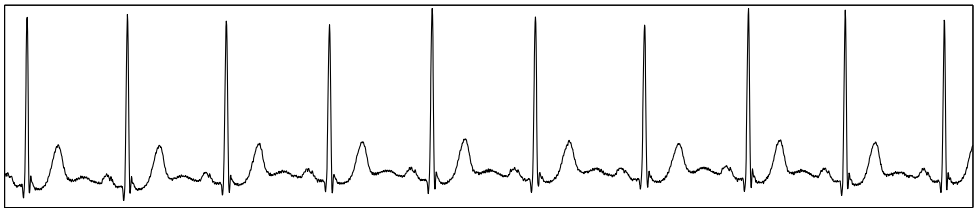
ECG compression test signals

This appendix contains short segments of the ECG signals used in the compression experiments in Chapter 7. This is to illustrate how the ECG signals vary with different heart arrhythmias.

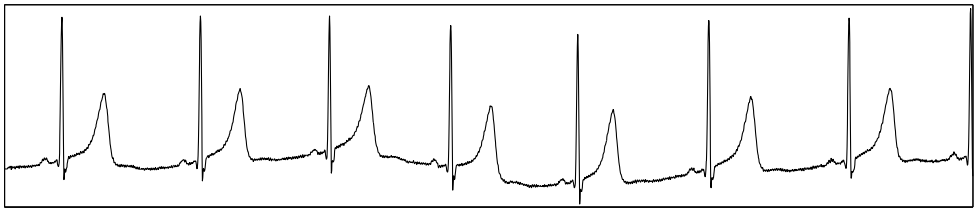
All the ECG signals are taken from the MIT-BIH Arrhythmia Database [31]. The sampling frequency is 360 Hz with a resolution of 12 bits per sample. In Chapter 7 we use both a training and a testing signal from MIT100, a training and a testing signal from MIT207, and a mixed training signal, called MITmix. This is a 5 minutes signal consisting of pieces from 5 different signals: MIT100, MIT103, MIT113, MIT207, and MIT217. While the MIT100 is normal sinus rhythm, the others are various abnormal rhythms. In Figure B.1, the first 8 seconds of each of the following signals are shown: a) MIT100, b) MIT103, c) MIT113, d) MIT207, and e) MIT217.



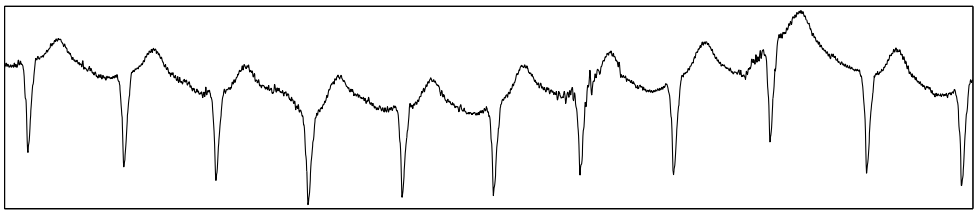
(a)



(b)



(c)



(d)

(e)

Figure B.1: The first 8 seconds of each of the following signals are shown: a) MIT100, b) MIT103, c) MIT113, d) MIT207, and e) MIT217.

Bibliography

[1] S. O. Aase, J. H. Husøy, K. Skretting, and K. Engan. Optimized signal expansions for sparse representations. *IEEE Trans. Signal Processing*, 49(5):1087–1096, May 2001.

[2] J. Adler, B. D. Rao, and K. Kreutz-Delgado. Comparison of basis selection methods. In *Proc. of the 30th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 252–257, Monterey, California, Nov 1996.

[3] H. Anton. *Elementary Linear Algebra*. John Wiley and Sons, Inc., New York, 7th edition, 1994.

[4] T. Berger. *Rate distortion theory: A mathematical basis for data compression*. Prentice Hall, 1971.

[5] R. Caetano, E. A. da Silva, and A. G. Ciancio. Matching pursuits video coding using generalized bit-planes. In *IEEE Proc. ICIP 2002*, volume 3, pages III–677 – III–680.

[6] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50(5):1873–1896, 1989.

[7] S. S. Chen. *Basis Pursuit*. PhD thesis, Stanford University, November 1995.

[8] S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado. Forward sequential algorithms for best basis selection. In *IEE Proc. - Vision, Image and Signal Processing*, volume 146, pages 235 – 244, October 1999.

[9] S. F. Cotter, M.N. M. N. Murthi, and B. D. Rao. Fast basis selection methods. In *Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1474–1478, November 1997.

[10] S. F. Cotter and B. D. Rao. Application of tree-based searches to matching pursuit. In *IEEE Proc. ICASSP 2001*, volume 6, pages 3933–3936, Salt Lake City, USA, May 2001.

[11] G. Davis. *Adaptive Nonlinear Approximations*. PhD thesis, New York University, September 1994.

[12] K. Engan. *Frame Based Signal Representation and Compression*. PhD thesis, Norwegian University of Science and Technology/ Stavanger University College, September 2000.

[13] K. Engan, S. O. Aase, and J. H. Husøy. Method of optimal directions for frame design. In *Proc. ICASSP '99*, pages 2443–2446, Phoenix, USA, March 1999.

[14] K. Engan, S. O. Aase, and J. H. Husøy. Multi-frame compression: Theory and design. *Signal Processing*, 80:2121–2140, October 2000.

[15] P. Frossard, P. Vandergheynst, R. M. Fi. Ventura, and M. Kunt. A posteriori quantization of progressive matching pursuit streams. *IEEE Trans. Signal Processing*, 52(2):525–535, February 2004.

[16] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[17] M. Gharavi-Alkhansari. A model for entropy coding in matching pursuit. In *IEEE Proc. ICIP '98*, pages 778–782, Chicago, USA, November 1998.

[18] M. Gharavi-Alkhansari and T. S. Huang. A fast orthogonal matching pursuit algorithm. In *IEEE Proc. ICASSP '98*, pages 1389–1392, Seattle, USA, May 1998.

[19] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm. *IEEE Trans. Signal Processing*, 45:600–616, March 1997.

[20] V. K. Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, pages 9–21, September 2001.

[21] V. K. Goyal and M. Vetterli. Consistency in quantized matching pursuit. In *Proc. Int. Conf. Acoust. Speech, Signal Proc.*, pages 1787–1790, Atlanta, May 1996.

[22] V. K. Goyal, M. Vetterli, and N. T. Thao. Quantization of overcomplete expansions. In *Proc. IEEE Data Compression Conf.*, pages 13–22, Utah, March 1995.

[23] D. Haugland, J.G. Heber, and J.H. Husøy. Optimisation algorithms for ECG data compression. *Medical & Biological Engineering & Computing*, 35:420–424, July 1997.

[24] D. A. Huffman. A method for the construction of minimum redundancy codes. *Proc. IRE*, 40(9):1098–1101, September 1952.

[25] J. H Husøy and T. Gjerde. Computationally efficient subband coding of ECG signals. *Medical Engineering and Physics*, 18:132–142, March 1996.

[26] MathWorks Inc. `http://www.mathworks.com/`.

[27] N. S. Jayant and Peter Noll. *Digital Coding of Waveforms*. Prentice-Hall, Englewood Cliffs, 1984.

[28] L. Koldal. Data compression using tabu search and partial search heuristics. Master's thesis, Stavanger University College, 2003 (in Norwegian).

[29] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41:3397–3415, December 1993.

[30] U. Manber. *Introduction to Algorithms, a Creative Approach*. Addison Wesley, 1989.

[31] Massachusetts Institute of Technology. *The MIT-BIH Arrhythmia Database CD-ROM*, 2nd edition, 1992.

[32] G. Melnikov, G. M. Schuster, and A. K. Katsaggelos. Shape coding using temporal correlation and joint VLC optimization. *IEEE Trans. Circuits and Systems for Video Technology*, pages 744–754, Aug 2000.

[33] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Legall. *MPEG Video: Compression Standard*. Van Nostrand Reinhold, New York, 1996.

[34] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227–234, April 1995.

[35] R. Neff and A. Zakhor. Matching-pursuit video coding - part ii: Operational models for rate and distortion. *IEEE Trans. Circuits and Systems for Video Technology*, pages 27–39, Jan 2002.

[36] R. Nygaard. *Shortest path methods in representation and compression of signals and image contours*. PhD thesis, Norges teknisk-naturvitenskapelige universitet (NTNU)/Høgskolen i Stavanger, September 2000.

[37] R. Nygaard, G. Melnikov, and A. K. Katsaggelos. A rate distortion optimal ECG coding algorithm. *IEEE Trans. Biomedical Engineering*, 48(1), Jan 2001.

[38] A. Ortega and K. Ramchandran. Rate-distortion methods for image and video compression. *IEEE Signal Processing Magazine*, pages 23–50, Nov 1998.

[39] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proc. of the 27th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 40–44, Monterey, California, Nov 1993.

[40] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1992.

[41] John G. Proakis and Dimitris M. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*. Prentice Hall Inc., New Jersey, 3rd edition, 1996.

[42] K. Ramchandran and M. Vetterli. Rate-distortion optimal fast tresholding with complete jpeg/mpeg decoder compability. *IEEE Trans. Image Processing*, 3(5):700–704, September 1994.

[43] T. Ryen, S. O. Aase, and J. H. Husøy. Finding sparse representation of quantized frame coefficients using 0-1 integer programming. In *Image and Signal Processing and Analysis, 2001. ISPA 2001*, pages 541–544, Pula, Croatia, 2001.

[44] T. Ryen, G. M. Schuster, and A. K. Katsaggelos. A frame-based rate-distortion optimal coding system using a lower bound depth-first-search strategy. In *Proc. of Nordic Signal Processing Symposium*, Tromsø, October 2002.

[45] T. Ryen, G. M. Schuster, and A. K. Katsaggelos. A rate-distortion optimal coding alternative to matching pursuit. In *IEEE Proc. ICASSP '02*, pages 2177–2180, Orlando, USA, May 2002.

[46] T. Ryen, G. M. Schuster, and A. K. Katsaggelos. Efficient frame vector selection based on ordered sets. In *IEEE Proc. ICIP '03*, pages III – 777–780 vol.2, Barcelona, Spain, September 2003.

[47] T. Ryen, G. M. Schuster, and A. K. Katsaggelos. A rate-distortion optimal alternative to matching pursuit. *IEEE Trans. Signal Processing*, 52:1352–1363, May 2004.

[48] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, San Francisco, 2nd edition, 2000.

[49] G. M. Schuster and A. K. Katsaggelos. *Rate-Distortion Based Video Compression*. Kluwer Academic Publishers, Boston, 1997.

[50] G. M. Schuster and A. K. Katsaggelos. An optimal polygonial boundary encoding scheme in the rate distortion sense. *IEEE Trans. Image Processing*, 7(1):13–26, January 1998.

[51] C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

[52] C. E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record*, 4:142–163, 1959.

[53] K. Skretting. *Sparse Signal Representation using Overlapping Frames.* PhD thesis, Norwegian University of Science and Technology/ Stavanger University College, October 2002.

[54] K. Skretting, K. Engan, J. H. Husøy, and S. O. Aase. Partial search vector selection for sparse signal representation. In *Proc. 12th Scandinavian Conf. on Image Analysis, SCIA 2001*, pages 613–620, Bergen, October 2001.

[55] K. Skretting and J. H. Husøy. Partial search vector selection for sparse signal representation. In *Proc. of Nordic Signal Processing Symposium*, Bergen, October 2003.

[56] K. Skretting, J. H. Husøy, and S. O. Aase. A simple design of sparse signal representations using overlapping frames. In *Image and Signal Processing and Analysis, 2001. ISPA 2001*, pages 424–428, Pula, Croatia, 2001.

[57] M. Vetterli and T. Kalker. Matching pursuit for compression and application to motion compensated video coding. In *IEEE Proc. ICIP 1994*, volume 1, pages 725–729.

[58] C. D. Vleeschouvwer and A. Zakhor. In-loop atom modulus quantization for matching pursuit and its application to video coding. *IEEE Trans. Image Processing*, 12(10):1226–1242, Oct 2003.

[59] Laurence A. Wolsey. *Integer programming.* John Wiley and Sons, Inc., New York, 1st edition, 1998.