Sparse Signal Representation using Overlapping Frames

by

Karl Skretting

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

Høgskolen i Stavanger
Norway

2002

Abstract

Signal expansions using frames may be considered as generalizations of signal representations based on transforms and filter banks. Frames for sparse signal representations may be designed using an iterative method with two main steps: (1) Frame vector selection and expansion coefficient determination for signals in a *training set*, – selected to be representative of the signals for which compact representations are desired, using the frame designed in the previous iteration. (2) Update of frame vectors with the objective of improving the representation of step (1). In this thesis we solve step (2) of the general frame design problem using the compact notation of linear algebra. This makes the solution both conceptually and computationally easy, especially for the non-block-oriented frames, – for short *overlapping frames*, that may be viewed as generalizations of critically sampled filter banks. Also, the solution is more general than those presented earlier, facilitating the imposition of constraints, such as symmetry, on the designed frame vectors. We also take a closer look at step (1) in the design method. Some of the available vector selection algorithms are reviewed, and adaptations to some of these are given. These adaptations make the algorithms better suited for both the frame design method and the sparse representation of signals problem, both for block-oriented and overlapping frames.

The performances of the improved frame design method are shown in extensive experiments. The sparse representation capabilities are illustrated both for one-dimensional and two-dimensional signals, and in both cases the new possibilities in frame design give better results.

Also a new method for texture classification, denoted Frame Texture Classification Method (FTCM), is presented. The main idea is that a frame trained for making sparse representations of a certain class of signals is a model for this signal class. The FTCM is applied to nine test images, yielding excellent overall performance, for many test images the number of wrongly classified pixels is more than halved, in comparison to state of the art texture classification methods presented in [59].

Finally, frames are analyzed from a practical viewpoint, rather than in a mathematical theoretic perspective. As a result of this, some new frame properties are suggested. So far, the new insight this has given has been moderate, but we think that this approach may be useful in frame analysis in the future.

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 John Håkon Husøy and professor Sven Ole Aase of Stavanger College University (Høgskolen i Stavanger, HiS), Stavanger, Norway, have been my supervisors.

The work, including compulsory courses corresponding to one year full time studies, as well as one year of undergraduate lecturer duties, has taken place in the period of August 1998 to August 2002 and was carried out at the Electrical and Computer Engineering Department of HiS. Parts of the work leading to this dissertation have been presented in [1,2,36,63,64,65].

Acknowledgments

First of all I would like to thank my supervisor, Professor John Håkon Husøy, for his inspiration, insight, invaluable support and help in finding the direction of this research work, and my second supervisor, Professor Sven Ole Aase, for his enthusiasm and help in finding practical ways of expressing the problems and solutions. They both have given considerable contributions to my professional, technical, and linguistic development.

My colleagues at Høgskolen i Stavanger have all contributed to a highly appreciated job atmosphere. Particularly I would like to thank Kjersti Engan for valuable discussion about the frame design method, the work in this thesis is to a large degree based on the work in her thesis [22].

Last but not least, I would like to thank my wife Anita for being supportive and together with our one year old daughter Maria, making these years as a PhD student the best years of my life. I would also like to thank my parents, Jofrid and Tobias, and the rest of my family and friends.

Contents

Nomenclature

$\|\cdot\|$	Norm, usually 2-norm or Frobenius (trace) norm
$\lfloor\cdot\rfloor$	the largest integer smaller or equal to its argument
$\lceil\cdot\rceil$	the smallest integer larger or equal to its argument
α	angle used as needed
β	angle between two frame vectors or between a frame vector and a signal vector
$\beta^{(min)}$	angle between the two frame vectors closest to each other
$\beta^{(avg)}$	average for each of the frame vectors of the angle to its closest neighbor
$\beta^{(mse)}$	average of all the angles between frame vectors taken in the "mean square sense"
$\beta^{(avg2)}$	average for each of the frame vectors of the angle to the center of the cluster formed by the frame vectors
$\Gamma(\cdot)$	gamma function
λ_n	eigenvalues of the frame operator
$\mathbf{\Lambda}$	eigenvalue matrix
σ_n	singular values of the frame
$\mathbf{\Sigma}$	singular value matrix
$\theta^{(avg)}$	average for the angles for all the frame vectors to its closest neighbor in another frame.
A	lower frame bound
A_s	norm of weights for a sparse representation is limited by $\|\mathbf{w}\| \leq (1/\sqrt{A_s})\|\mathbf{x}\|$
$\mathbf{A}^\dagger$	pseudoinverse of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	inverse of matrix $\mathbf{A}$
$\mathbf{A}^T$	transpose of matrix $\mathbf{A}$
$\mathbf{A}_{ij}$	entry i in column j of matrix $\mathbf{A}$
$[\mathbf{A}]_{ij}$	entry i in column j of matrix $\mathbf{A}$
$\mathbf{a}_j$	column j of matrix $\mathbf{A}$
$a_j(i)$	entry i in vector $\mathbf{a}_j$

B	upper frame bound
$\mathbf{B}$	matrix that represents an image
$\mathbf{B}_m$	image number m in a training set of images
$\tilde{\mathbf{B}}$	reconstructed image
$\mathbf{C}^*$	matrix to be inverted when a new frame is calculated, $\mathbf{C}^* = \mathbf{W}^*\mathbf{W}^{*T}$
$\mathbf{C}_p$	a submatrix of $\mathbf{C}^*$
$\mathbf{C}$	the $Q \times Q$ matrix to be inverted for a general frame
$\mathbf{D}$	product of weights and polyphase components of the signal, Equation 2.28
$\mathbf{d}$	product of weights and polyphase components of the signal,Equation 2.34
F	multi-dimensional frame
$\mathbf{F}$	frame matrix where the columns are the synthesis vectors, size $NP \times K$
$\tilde{\mathbf{F}}$	dual frame
$\mathbf{F}^{(i)}$	frame in iteration i during frame design
$\mathbf{F}_p$	part of the overlapping frame $\mathbf{F}$, Equation 2.11
$\mathbf{F}_m$	"frame" made by only some of the columns in $\mathbf{F}$, m tells which index set to use, and the index sets tell which columns to use
$\mathbf{F}_h$	horizontal part of a separable frame
$\mathbf{F}_v$	vertical part of a separable frame
$\mathbf{F}^*$	overlapping frame $\mathbf{F}$ reshaped into size $N \times KP$
$\mathbf{F}^M$	frame consisting of M repetitions of $\mathbf{F}$
$\mathcal{F}$	large frame consisting of many repetitions of $\mathbf{F}$
$\tilde{\mathcal{F}}$	part of the $\mathcal{F}$ frame matrix, Equation 7.26
$\mathbf{f}$	vector formed by all the free variables in $\mathbf{F}$
$\mathbf{f}_k$	synthesis vector, a column vector of $\mathbf{F}$
$\overline{\mathbf{f}}_n$	row of $\mathbf{F}$ reshaped into a column vector
$\overline{\mathbf{f}}_n^*$	row of $\mathbf{F}^*$ reshaped into a column vector
$\mathbf{G}$	orthonormal filter bank
$\mathcal{G}$	large matrix representation of an orthonormal filter bank
H	Hilbert space
$\mathbf{H}$	block-oriented frame
$\mathcal{H}$	large matrix representation of a block-oriented frame
i	index used as needed
$\mathbf{I}_N$	identity matrix of size $N \times N$
I_C	total number of different submatrices in matrix $\mathbf{C}^*$

j	index used as needed, or the imaginary unit $j = \sqrt{-1}$
$J(\mathbf{F}, \mathbf{W})$	objective function during frame design
J_{MS}	objective function in filter design (Mahalanobis and Singh)
J_U	objective function in filter design (Unser)
J_F	objective function in filter design (Fisher)
K	number of frame vectors in the frame
L	number of signal blocks in a training set
M	number of times a frame $\mathbf{F}$ is repeated in $\mathbf{F}^M$, or number of images in a set of training images, or number of index sets
N	length of a signal block
P	overlap factor for a frame
Q	total number of free variables in a frame
R	reorder or reshape operator
$\mathbf{R}(z)$	synthesis polyphase matrix of an overlapping frame
$\mathbf{R}$	reconstruction error or residual
$\mathbf{r}_l$	reconstruction error or residual of a signal block
$\mathbf{r}$	reconstruction error or residual for signal $\mathbf{x}$
r	relative representation error $\|\mathbf{r}\|/\|\mathbf{x}\|$.
$r_s^{(max)}$	maximal relative representation error using s frame vectors
$r_s^{(avg)}$	mean (average) relative representation error
$r_s^{(mse)}$	square root of the mean square relative representation error
$\mathbb{R}^N$	N-dimensional real Hilbert space
S	sparseness factor
S_d	sparseness factor used in frame design
S_{lp}	part of the sparseness factor from the low-pass representation
S_t	total sparseness factor, $S_t = S + S_{lp}$
$S_N(r)$	surface of a ball with radius r in $\mathbb{R}^N$
s	number of frame vectors used to represent a signal
s_l	number of frame vectors used for a particular signal block
$\mathbf{S}(z)$	polyphase matrix of of the frame operator
$\mathbf{S}$	frame operator
SNR_t	The SNR achieved during training of a frame
T	operator defined by a transform or a filter bank
$\mathbf{T}$	transform matrix
$\mathbf{t}_n$	column n of the transform matrix $\mathbf{T}$
$\mathbf{u}$	inner product vector
$\mathbf{U}$	inner product matrix
$\mathbf{U}$	left unitary matrix in a SVD of a matrix, eigenvectors in an eigenvalue decomposition

$V_N(r)$	volume of a ball with radius r in $\mathbb{R}^N$
$\mathbf{v}$	vector used when needed
$\mathbf{V}$	right unitary matrix in a SVD of a matrix
W	multi-dimensional weight matrix
$\mathbf{W}$	matrix of the weights of size $K \times L$
$\mathbf{W}^*$	large weight matrix of size $KP \times L$
$\mathcal{W}$	huge weight matrix of size $NL \times NKP$ or $NL \times Q$
$\overrightarrow{\mathbf{W}}^p$	weight matrix where the columns are (circularly) shifted p positions to the right
$\overrightarrow{W}$	multi-dimensional shifted weight matrix
$\mathbf{w}$	weight vector
$\mathbf{w}_l$	weight vector corresponding to a signal block
$\mathbf{w}^M$	concatenation of M weight vector blocks
X	multi-dimensional signal
$\mathbf{X}$	matrix where the columns are the signal blocks
$\tilde{\mathbf{X}}$	reconstruction of the matrix where the columns are the signal blocks
$\mathbf{x}$	one-dimensional signal
$\tilde{\mathbf{x}}$	reconstruction of a one-dimensional signal
$\tilde{\mathbf{x}}^M$	part of reconstructed signal using part $\mathbf{F}^M$ of $\mathcal{F}$, Figure 4.4
$\tilde{\mathbf{x}}^a$	part of reconstructed signal using part of $\mathcal{F}$ after $\mathbf{F}^M$
$\tilde{\mathbf{x}}^b$	part of reconstructed signal using part of $\mathcal{F}$ before $\mathbf{F}^M$
$\mathbf{x}_l$	signal block
$\tilde{\mathbf{x}}_l$	reconstructed signal block
$\overline{\mathbf{x}}_n$	row of $\mathbf{X}$ reshaped into a column vector, polyphase component of $\mathbf{x}$
$\overline{\mathbf{x}}$	the vector made by concatenating the N $\overline{\mathbf{x}}_n$ vectors
$\overline{\mathbf{x}}$	in Equation 1.9, the mean of the signal blocks
$y(n)$	transform, or filter bank, coefficient
$\tilde{y}(n)$	quantized transform coefficient
$\mathbf{y}$	transform, or filter bank, coefficient vector
$\tilde{\mathbf{y}}$	quantized transform coefficient vector
$\mathbf{Y}$	matrix of transform coefficients
$\tilde{\mathbf{Y}}$	reconstructed coefficients
z	complex number

List of Abbreviations

1D	one-dimensional
2D	two-dimensional
AR	autoregressive
BP	basis pursuit, an algorithm to find the weights in $\tilde{\mathbf{x}} = \mathbf{Fw}$
BMP	basic matching pursuit, a greedy vector selection algorithm
dB	decibel
DC	direct current
DCT	discrete cosine transform
ECG	electrocardiogram
ELT	extended lapped transform
FIR	finite input response
FOCUSS	focal under-determined system solver, a parallel vector selection algorithm
FOMP	fast (or fully) orthogonal matching pursuit, a greedy vector selection algorithm
FTCM	frame texture classification method
FS	the full search vector selection algorithm
GLA	generalized Lloyd algorithm
GMP	global matching pursuit, a greedy vector selection algorithm
JPEG	joint photographic expert group
KLT	Karhunen-Loève transform
LOT	lapped orthogonal transform
LP	linear programming
LVQ	learning vector quantizing

MD	multi-dimensional
MIT100	signal 100 from the MIT arrhythmia database of ECG signals
MMP	modified matching pursuit, a greedy vector selection algorithm
MOF	method of frames, an algorithm to find the weights in $\tilde{\mathbf{x}} = \mathbf{Fw}$
MP	matching pursuit, a group of greedy vector selection algorithms
MSE	mean squared error
OMP	orthogonal matching pursuit, a greedy vector selection algorithm
ONB	orthonormal basis
ORMP	order recursive matching pursuit, a greedy vector selection algorithm
pdf	probability density function
PR	perfect reconstruction
PSNR	peak signal to noise ratio
QMF	quadrature mirror filter
QRcp	QR-factorization with column pivoting, an algorithm to find the weights in $\tilde{\mathbf{x}} = \mathbf{Fw}$
SAR	synthetic aperture radar
SNR	signal to noise ratio
SRF	frame for sparse representation
SVD	singular value decomposition
TF	tight frame
UB	unit ball
UF	uniform frame
UTF	uniform tight frame
VQ	vector quantization

Chapter 1

Introduction

The demand for digital signal processing is continuously increasing. All real world properties or activities which are being recorded generate signals. Common examples are speech or audio signals, images and video signals, medical signals such as ElectroCardioGram (ECG) signals, and seismic signals. The ever increasing recording of signals gives an increased need for analysis, compression, transmission, and storage. Signal *representation* is important to do these tasks in an effective way. During recording the signal is typically sampled and amplitude quantized, this gives a digital signal. The representation of a signal through its sample values is the simplest possible representation. More effective or alternative representations or parameterizations may be desirable in many applications. By effective we here mean a representation using fewer parameters than samples, i.e. a *sparse* representation, or a representation using parameters that may be stored using few bits, i.e. signal compression, or a representation using parameters that somehow reflects important properties of the signal, i.e. a model based signal representation. This latter one is often also a sparse representation. A common method for signal representation is to transform the signal into some coefficients using a transform or a filter bank. In this thesis we will investigate an alternative method for signal representation, which is a generalization of the transform and filter bank approach. In particular we will show how to design *frames*, a frame plays a similar role in our alternative method for signal representation as a transform does in the transform domain representation. We will demonstrate how the designed frames can be used in sparse signal representation in general, and especially how frames may be applied in texture classification. In the end we will also discuss some possible frame properties.

This introduction, intended as a preview of several issues dealt with fully later on, is organized in four sections. First we review standard signal expansions

and relate them to the *overlapping frame* concept. In the second part we explain the goal and motivation for a *sparse* signal expansion. Closely related to sparse representation using a frame is the vector selection problem, this is discussed in the third section. The chapter is concluded by a section explaining the scope and contributions of this work.

1.1 Signal expansions

A signal expansion is the representation of a signal as a linear combination of some basic synthesis signals. For simplicity we only consider real, one-dimensional (1D) signals in this section.

A signal, or a block of a longer signal, $\mathbf{x} \in \mathbb{R}^N$, may be represented using some transform coefficients, denoted $\mathbf{y}$. The forward transform, which we for notational convenience denote by $\mathbf{T}^{-1}$, is used to compute the transform coefficients. They are found by the *analysis equation*, $\mathbf{y} = \mathbf{T}^{-1}\mathbf{x}$. The reconstructed signal vector is then given by the corresponding *synthesis equation*, where the tilde is used to indicate the possibility of approximated quantities:

$$\tilde{\mathbf{x}} = \mathbf{T}\,\tilde{\mathbf{y}} = [\mathbf{t}_1\mathbf{t}_2\cdots\mathbf{t}_N] \begin{bmatrix} \tilde{y}(1) \\ \tilde{y}(2) \\ \vdots \\ \tilde{y}(N) \end{bmatrix} = \sum_{n=1}^{N} \tilde{y}(n)\mathbf{t}_n. \tag{1.1}$$

The *synthesis vectors*, denoted $\{\mathbf{t}_n\}_{n=1}^N$, are the columns of the transform matrix $\mathbf{T}$. In the case of common transforms, such as the Discrete Cosine Transform (DCT) and the Karhunen-Loève Transform (KLT) [28], these synthesis vectors form an orthogonal basis for $\mathbb{R}^N$. We should note that even if the synthesis vectors are not orthogonal, they should form a basis for $\mathbb{R}^N$ since this is necessary for the inverse, $\mathbf{T}^{-1}$, to exist. The reconstructed signal is built up as a linear combination of these synthesis vectors.

Allowing for the possibility of having more than N terms in the linear combination of Equation 1.1, say $K \geq N$ terms, the collection of K vectors will be denoted as $\{\mathbf{f}_k\}_{k=1}^K$. Interpreting these vectors, collectively referred to as a *frame* or a *dictionary*, as columns of an $N \times K$ matrix $\mathbf{F}$ we have a more general situation than that of Equation 1.1. The synthesis equation for the frame case has the same form as in the transform case:

$$\tilde{\mathbf{x}} = \mathbf{F}\,\mathbf{w} = \sum_{k=1}^{K} w(k)\mathbf{f}_k. \tag{1.2}$$

Here we have replaced the transform coefficients $\mathbf{y}$ by the weights used for each synthesis vector, $\mathbf{w}$. Since a signal block of length N is reconstructed in Equation 1.1 and Equation 1.2, both are *block-oriented* signal representations. An analysis equation can not be used to find the weights in Equation 1.2 since the solution is not unique and the inverse of $\mathbf{F}$ does not exist. We will return to the problem of finding the weights in Section 1.3 and Chapter 4.

We will now look at different ways to represent the synthesis equation for the block-oriented case, both to introduce the different notations and to prepare for the more general overlapping case. A long signal, $\mathbf{x}$ (size: $NL \times 1$), is divided into L blocks of length N, where each block is represented by a column vector denoted by $\mathbf{x}_l$. The synthesis equations for the blocks,

$$\tilde{\mathbf{x}}_l = \mathbf{F}\,\mathbf{w}_l = \sum_{k=1}^{K} w_l(k)\mathbf{f}_k, \quad l = 1, 2, \ldots, L, \tag{1.3}$$

can be written as

$$N\updownarrow \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \vdots \\ \tilde{\mathbf{x}}_l \\ \vdots \\ \tilde{\mathbf{x}}_L \end{bmatrix} = \begin{bmatrix} \boxed{\mathbf{F}} & & & \\ & \ddots & \leftarrow K \rightarrow & \\ & & \boxed{\mathbf{F}} & \updownarrow N \\ & & & \ddots \\ & & & \boxed{\mathbf{F}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_l \\ \vdots \\ \mathbf{w}_L \end{bmatrix} \updownarrow K \tag{1.4}$$

$$\text{or} \qquad \tilde{\mathbf{x}} = \mathcal{F}\,\mathbf{w}. \tag{1.5}$$

Here we have introduced the large frame matrix $\mathcal{F}$ in the synthesis equation. This is a block-diagonal and band-diagonal matrix (size: $NL \times KL$), where entries outside the blocks in the diagonal band are zero. The notation in Equation 1.5 will be especially convenient for the overlapping case, to be explained shortly, but for the block-oriented case another version of the synthesis equation is usually preferred. Defining $\mathbf{X} = [\ \mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_L\ ]$, (size: $N \times L$) and $\mathbf{W} = [\ \mathbf{w}_1\ \mathbf{w}_2\ \cdots\ \mathbf{w}_L\ ]$, (size: $K \times L$) we can write

$$\tilde{\mathbf{X}} = \mathbf{F}\,\mathbf{W}. \tag{1.6}$$

When we are dealing with approximative representations, the mean squared error (MSE) can be calculated as

$$MSE = \frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}{NL} = \frac{\|\mathbf{X} - \tilde{\mathbf{X}}\|^2}{NL} = \frac{1}{NL}\sum_{l=1}^{L} \|\mathbf{x}_l - \tilde{\mathbf{x}}_l\|^2, \tag{1.7}$$

where $\|\mathbf{x}\|$ is the *norm* of $\mathbf{x}$. Generally in this thesis, if not otherwise mentioned, the 2-norm for vectors and the trace (Frobenius) norm for matrices, $\|\mathbf{A}\|^2 = \|\mathbf{A}\|_F^2 = \text{trace}(\mathbf{A}^T\mathbf{A}) = \sum_{n=1}^{N}\sum_{k=1}^{K}(\mathbf{A}_{nk})^2$ will be used. Another common error measure, useful in assessing approximate signal representations, is the Signal to Noise Ratio (SNR), usually expressed in decibel as

$$SNR = 10\log_{10}\frac{\sigma_x^2}{MSE}\ [dB] \tag{1.8}$$

where σ_x^2 is the variance of the signal. A practical way to estimate the SNR is

$$SNR = 10\log_{10}\frac{\sum_{l=1}^{L}\|\mathbf{x}_l - \overline{\mathbf{x}}\|^2}{\sum_{l=1}^{L}\|\mathbf{x}_l - \tilde{\mathbf{x}}_l\|^2}\ [dB] \tag{1.9}$$

where $\overline{\mathbf{x}}$ is a vector with the mean of the signal in all entries, $\overline{\mathbf{x}} = [\overline{x}, \overline{x}, \ldots, \overline{x}]^T$ and $\overline{x} = \frac{1}{NL}\sum_{l=1}^{L}\sum_{n=1}^{N} x_l(n)$.

In many signal processing applications critically sampled filter banks are known to perform better than block-oriented transforms. The difference between a synthesis transform and a synthesis filter bank is as follows: In the former case the linear combination of length N basis vectors of the synthesis equation, – Equation 1.1 or Equation 1.2 with $K = N$, completely describes the reconstruction of a signal block. In the latter case, the $K = N$ basis (synthesis) vectors have length NP, where P is an integer. All P length N parts of these vectors are involved in the reconstruction of a length N signal block [58]. As will be seen below this makes it natural to talk about *overlapping* synthesis vectors. Using the input-output relations for an upsampler and a linear filter [71], the input-output relation for a uniform K channel synthesis filter bank can be verified to correspond to a matrix vector product [58] and it can be written as in Equation 1.5. The expansion to a form like that of Equation 1.4 will now be

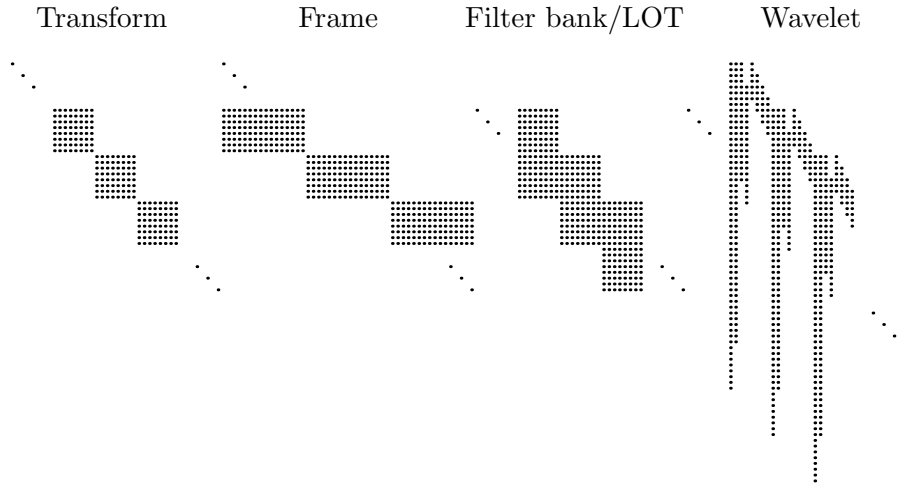


Figure 1.1: The support structure of 4 different synthesis matrices $\mathcal{F}$. Each column of dots represent one synthesis vector. Note the block-oriented structure for the transform and frame case, and the overlapping structure for the filter bank case. The shown filter bank is a 8 channel Lapped Orthogonal Transform (LOT) [46] and each filter has length 16. The wavelet structure corresponds to a three level dyadic tree with filter lengths of 7 for the high-pass synthesis filter and 9 for the low-pass synthesis filter. The resulting synthesis filters have different length.

$$\begin{bmatrix} \vdots \\ \tilde{\mathbf{x}}_l \\ \tilde{\mathbf{x}}_{l+1} \\ \tilde{\mathbf{x}}_{l+2} \\ \tilde{\mathbf{x}}_{l+3} \\ \tilde{\mathbf{x}}_{l+4} \\ \vdots \end{bmatrix} = \begin{bmatrix} \ddots & \leftarrow K \rightarrow & & \\ \ddots & \mathbf{F} & \updownarrow N & \\ \ddots & & \mathbf{F} & \\ \ddots & & & \mathbf{F} \\ & & & \ddots \\ & & & \ddots \\ & & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{w}_l \\ \mathbf{w}_{l+1} \\ \mathbf{w}_{l+2} \\ \vdots \end{bmatrix} \tag{1.10}$$

The main diagonal (when $K = N$) of the large matrix goes through the entries $[\mathbf{F}]_{n,n}$ for $n = 1, 2, \ldots, N$. The signal, the vector of weights, and the matrix $\mathcal{F}$ are still assumed to be finite, but since it is not yet defined how to reconstruct the ends of the signal, only the central part of the equation is shown. How to treat the signal ends is further discussed in Section 2.3. The columns of the

constituent matrices $\mathbf{F}$ are the synthesis filters. The length of these filters is NP. P is called the overlap factor, since P of the $\mathbf{F}$ matrices overlap each other. We note that having $P = 1$, this reduces to the block-oriented case. For an orthogonal perfect reconstruction filter bank we will have $\mathcal{F}^{-1} = \mathcal{F}^T$ and $\mathbf{F}^T\mathbf{F} = \mathbf{I}_N$ where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

The overlapping frame is not a new concept, actually it is an oversampled synthesis filter bank. The oversampled filter bank is a frame if perfect reconstruction is possible [8] [15]. Like the frame is an extension to the transform, the overlapping frame is an extension to the critically sampled synthesis filter bank. In this thesis the term overlapping frame is preferred since it relates more to the block-oriented frame concept than to the filter bank concept where the main issue often is the relationship between the analysis and the synthesis filter bank. The columns of the matrix $\mathbf{F}$ are the synthesis filters. This structure offers a great flexibility for the synthesis system, all the different structures in Figure 1.1 can be implemented. The rightmost part of Figure 1.1, the wavelet case, needs some comments: A tree structured filter bank, like a dyadic wavelet filer bank, can be implemented as a general filter bank with one level, see Figure 2.3 for details on how this is done. It is the general filter bank structure, corresponding to a three level dyadic wavelet tree where the lengths of the wavelet filters are 7 for the high-pass synthesis filter and 9 for the low-pass synthesis filter, which is illustrated in Figure 1.1.

As a summary we may classify the different forms of signal expansion:

	Complete expansions, $K = N$	**Overcomplete expansions**, $K > N$
Block-oriented expansions, $P = 1$	Transform	Block-oriented frame
Overlapping expansions, $P > 1$	Critically sampled filter bank	Overlapping frame

1.2 Sparse signal representations

Sparse signal representations are useful in different applications. In commonly used block-oriented transforms, e.g. the DCT, and the more recent wavelet based decomposition methods, the sparseness is introduced through thresholding of the transform or wavelet coefficients. Thus only a limited number of non-zero coefficients represent the signal. This introduces errors in the reconstructed signal. The goal is to minimize these errors while fulfilling a sparsity constraint making the number of non-zero coefficients small compared to the number of samples in the original signal. The non-zero coefficients as well as their position information constitute a sparse representation of the signal, useful in many applications like compression, feature extraction, modelling, and classification.

For images, this has an analogy in the human visual system, where an image is interpreted as many objects placed relative to each other. This is a sparse representation of the image. We do not perceive the millions of pixels that are received in the retina, but rather a sparse image of some few constituent parts (objects) with different shapes and textures.

For music, the notes are a sparse representation of the music. The vertical position gives the frequency, the horizontal position gives the time and the shape of the note gives the duration. The actual waveforms are given by the instrument and the musician. Each second of music may be represented by some few notes giving a very compact description of the essential properties of the music. This is actually a model based representation. Time-frequency signal analysis may be used to obtain a similar, usually sparse, description of most signals. This has been useful for signals much more "complex" than simple music, like speech signals and seismic signals.

Sparse representations using frames are achieved by allowing only a limited number of the coefficients/weights in $\mathbf{w}$ to be non-zero. The sparsity of the representation is expressed by the *sparseness factor*

$$S = \frac{\text{number of non-zero coefficients in } \mathbf{w}}{\text{number of samples in } \mathbf{x}}. \tag{1.11}$$

An overcomplete frame will allow greater flexibility in matching the signal with a sparse expansion than a complete (orthogonal) one. Having more than N frame vectors to choose from when forming the sparse representation improves the flexibility. For a given sparseness factor we should expect smaller error in

the reconstructed signal using an overcomplete frame rather than a complete expansion.

Efficient transforms and filter banks are often constructed based on the statistical properties of the signal, and the signal is often assumed to be generated by a stationary process. Common desired properties are (almost) perfect reconstruction, easy and efficient implementation, energy compaction and small artifacts, like blocking or ringing effects which often can be seen on compressed images. For K close to N we have a frame with a low degree of overcompleteness, the possibilities of such a frame will in many ways be similar to those of a transform or filter bank. Also the design targets and the design methods will be similar, but not much attention has been devoted to the design of such frames. As the ratio K/N increases the frame should become more capable of representing signals with varying statistical properties in the different signal blocks or groups of signal blocks, provided signal properties are exploited in the design of the frames.

The frame representation may alternatively be viewed as a model based system. The model is the signal viewed as a sparse linear combination of the K possible frame vectors, which may occur at different translations of factor N, which can be one. If we had a frame consisting of vectors each corresponding to the sound of a key on a keyboard, then this frame could be used to get a sparse representation of any music played on this instrument. The synthesizer is well suited to illustrate this frame model. Here several "frames" are available, each imitate a real, or artificial, instrument, and the wanted one is selected by some switches beside the keyboard.

In recent years sparse representation has found several applications: 1) Low bit rate video coding [4,51,52,72,73], where coding of the motion residual images can be done by a sparse representation usually using large overcomplete dictionaries, i.e. frames. Note that quite large errors are allowed when representing the motion residual images, thus what is represented is only the most important changes. 2) Still image compression has also been investigated [6,7,22,29,41] but here results are generally not that promising, mainly since natural images are difficult to represent in a very sparse way as opposed to motion residuals, and since what may be gained by having few non-zero weights often is lost in the entropy coding scheme where position information also has to be taken into account. 3) Other applications like template matching [30], of which road sign interpretation [35] is a special case, or medical signal analysis [21] have also been reported. In Chapter 6 in this thesis some promising results on texture classification using frames are presented.

Different kinds of frames have been used. Examples include the frame created by concatenation of the orthogonal basis vectors of the DCT and those of the

Haar transform [6], Gabor functions [51], [4], oversampled filter banks and wavelet trees [73], and Gaussian chirps [33]. These selections are motivated by their good time-frequency resolution and efficient implementation.

The requirements of the application usually set a limit on the magnitude of the representation error, and this in turn limits how small the sparseness factor can be. The signal class to be represented is of course also very important for the feasible sparseness factor. As already mentioned, the motion residual images in video coding are examples of images that can be represented in a sparse way. Natural images, general two-dimensional signals, and signals of several dimensions, are generally more sparse in nature than one-dimensional signals, and lower sparseness factor is possible without introducing more errors.

Our experience suggest a connection between the attainable sparseness factor, the appropriate signal representation, and the application that may roughly be summarized as follows:

- $S > 1$: This case gives signal expansions with more weights than signal samples. These may be used for signal analysis, and applications where error resilience is an issue. Frames, using the Method of Frames, Section 1.3, to find the weights, and oversampled filter banks and wavelets are such signal expansions.

- $S = 1$: This case should be used when close to perfect reconstruction (PR) is demanded. Orthogonal and bi-orthogonal transforms and critically sampled filter banks, including wavelets, are signal decompositions with the PR property, but rounding of the coefficients may introduce small acceptable errors. This is often used in lossless and nearly lossless signal compression.

- $0.1 < S < 1$: Signal expansions with sparseness factor a little bit lower than 1 is often found by thresholding (and quantizing) the transform or filter bank coefficients. Common applications are lossy compression and noise reduction.

- $0.01 < S < 0.25$: Applications with sparseness factors in this range may be lossy compression where a low bit rate is important and in applications that need only (some of) the essential properties of the signal, one example is texture classification. For this case we think that block-oriented or overlapping frames as presented in this work, with the degree of overcompleteness of moderate size $1 < \frac{K}{N} < 10$, will be suitable.

- $0.001 < S < 0.05$: This range of the sparseness factor can be used for similar applications as in the previous point. But to compensate for the lower sparseness factor, frames with a large degree of overcompleteness, typically $10 << \frac{K}{N}$, are used. These frames or dictionaries are built in a systematic way such that fast algorithms may be used. In fact, certain dictionaries have fast implicit algorithms, $\mathbf{Fw}$ and $\mathbf{F}^T\mathbf{x}$ may be computed fast and often also without explicit storage of the large matrices $\mathbf{F}$ and $\mathbf{F}^T$.

- $S < 0.01$: Model based representations often have a very low sparseness factor. Reconstruction of the original signal is often not possible, even if quite large errors are allowed, but construction of a similar signal, with the same properties, is an option. A text may be regarded as a model for spoken words. The sound, represented by its recorded samples, that is generated when the text is read will generally be quite different from one reader to another. Nevertheless, the "reconstructed sound signal" will contain the same essential information.

1.3 Vector Selection and Matching Pursuit

The vector selection problem (for a given sparseness factor, S) is to find the weights[1], $\mathbf{w}$, in the equation,

$$\mathbf{x} = \tilde{\mathbf{x}} + \mathbf{r} = \mathbf{Fw} + \mathbf{r}, \tag{1.12}$$

such that the norm of the error, $\|\mathbf{r}\|$, is as small as possible and the number of non-zero weights is smaller or equal to a target integer, s. Often this integer, denoted by lowercase s, rather than the sparseness factor S, is given in the problem. They are related by $S = s/N$. If the sparseness factor as defined in Equation 1.11 is given, the number of non-zero weights allowed is $s = \lfloor SN \rfloor$, where $\lfloor \cdot \rfloor$ denote the largest integer smaller or equal to its argument. Usually, we assume that the columns of $\mathbf{F}$ have norm one[2], i.e. $\|\mathbf{f}_k\| = 1$. Such a frame is denoted a uniform frame.

[1] Strictly speaking, the vectors should be indexed by l to keep notation in line with Equation 1.4. For notational simplicity, we here consider only a single signal block making the indexing unnecessary.

[2] Earlier we denoted such a frame as normalized, but the common terminology, [11] and [39], seems to use the term "a normalized frame" for a tight frame where the frame bounds, see Equation 7.1, both are one, $A = B = 1$.

The linear equation system, $\mathbf{Fw} = \mathbf{x}$, ($\mathbf{w}$ is the unknown variable), where we have the frame $\mathbf{F}$ (size: $N \times K$) with $N < K$ and $rank(\mathbf{F}) = N$, is *underdetermined*. It has many exact solutions when $N \leq s \leq K$, but for $1 \leq s < N$, generally no exact solution exists. Now, a sparseness criterion is imposed to the equation system, $s < N$. Then the choice of weights that minimizes the 2-norm of the residual (error) is called the "optimal solution" or the optimal approximation in the least squares sense. An ϵ-solution is a choice of weights such that the 2-norm of the residual is smaller than a given limit, ϵ. Davis [19] has proved that finding whether an ϵ-solution exists or not is an NP-complete[3] problem, and that finding the optimal approximation for a given s is an NP-hard[4] problem. This implies that a practical solution to this problem needs to use algorithms that are not guaranteed to find the true optimal solution.

A large amount of work has been done on this problem. Some of the algorithms that may be used are described and compared in [61], [62], [14]. We will here briefly review the different algorithms that can be used to find the weights in the equation above, both when the sparseness constraint is imposed and when it is relaxed (ignored). For the algorithms that do not return a sparse solution thresholding of the weights can be done to comply to the sparseness criterium. For some of the algorithms the number of flops (floating point operations) used by Matlab when solving a quite small ($N = 8$, $K = 16$, and $s = 4$) problem is given.

FS The *Full Search* algorithm finds the optimal solution to the problem. It examines/checks all the possible combinations for selecting s vectors out of the K frame vectors available. The number of different combinations is $M = \binom{K}{s}$. For each of these M combinations the frame vectors with indices $I_m = \{k_i^{(m)}\}_{i=1}^{s}, \quad m = 1, 2, ..., M$, are used to build a matrix $\mathbf{F}_m$ (size: $N \times s$). The solution, $\mathbf{w}_m = w(I_m)$ (size: $s \times 1$), is then found by solving the *overdetermined* equation system, $\mathbf{F}_m\mathbf{w}_m = \mathbf{x}$, in the least squares sense. If $\mathbf{F}_m$ has full rank (s) this solution is $\mathbf{w}_m = (\mathbf{F}_m^T\mathbf{F}_m)^{-1}\mathbf{F}_m^T\mathbf{x}$. The best of these M solutions is the optimal approximation for the vector selection problem. This algorithm is only practical for quite small problems. For the small example problem 3.95 million flops was used. Another example: having $N = 64$, $K = 128$, and $s = 8$ will give $M = \binom{K}{s} = 1.42 \cdot 10^{12}$ different combinations to check. Using Matlab on a 500 MHz PC, for each possible combination of selected vectors approximately 14000 flops and 0.55 milliseconds is needed. To examine all combinations requires almost 25 years!

[3] An NP-complete problem is as hard as (can be transformed in polynomial time into) another problem known to be NP-complete.

[4] An NP-hard problem is at least as hard as an NP-complete problem.

MOF The *Method of Frames* [17] use the generalized inverse or pseudoinverse to find a solution, $\mathbf{w} = \mathbf{F}^{\dagger}\mathbf{x}$ where $\mathbf{F}^{\dagger} = \mathbf{F}^T(\mathbf{F}\mathbf{F}^T)^{-1}$. This algorithm is especially useful when the frame is *tight* (see Section 7.1). Then the generalized inverse is simply $\mathbf{F}^{\dagger} = A^{-1}\mathbf{F}^T$ where A is the frame bound. The solution found by MOF is the one that minimizes the 2-norm of $\mathbf{w}$, thus it is also called the minimum-norm solution. For the small example problem 23216 flops was used.

QRcp *QR-factorization with column pivoting* is used by the backslash operator in Matlab, `w=F\x`. It finds an exact solution using N of the vectors from $\mathbf{F}$. This is one of the more computationally efficient and stable algorithms for solving an under-determined equation system, but the solution has no special properties. These N vectors (usually) form a basis of $\mathbb{R}^N$, and the solution is $\mathbf{x}$ expressed using this basis. For the small example problem 3110 flops was used.

BP *Basis Pursuit* [12] finds an optimal basis using N of the K column vectors of $\mathbf{F}$, the basis is optimal in the sense that the 1-norm of the solution, $\|\mathbf{w}\|_1$, is minimized. The problem can be expressed as a Linear Programming (LP) problem, and solved using LP methods. Due to recent advances in LP this problem can be solved for quite large frames. One advantage with this solution is that it is often sparse, but wether it is or not is not known until the solution is found. For the small example problem 268304 flops was used.

FOCUSS *FOCal Under-determined System Solver* [32] is a parallel vector selection algorithm. While standard FOCUSS search for an exact solution, a more practical (when signal has noise) variant is Regularized FOCUSS [22] which search for a good approximation. Both variants try to minimize an object function which include a term similar to the p-norm, $\|\mathbf{w}\|_p^p = \sum_{k=1}^{K} |w(k)|^p$. The p-norm is defined for $p \geq 1$ but in FOCUSS the term is used for values of $p \leq 1$. As p get close to zero, this term get close to the numerousity measure of $\mathbf{w}$, i.e. the number of non-zero elements in $\mathbf{w}$. Engan [22] found that Regularized FOCUSS usually finds a good sparse solution, but it is computationally demanding.

None of the algorithms above are suitable when a sparseness constraint is to be satisfied. FS is only feasible for very small problems, MOF and QRcp are not sparse, BP and FOCUSS are often sparse but give no (exact) control of the number of non-zero weights used. By thresholding the smallest weights a solution fulfilling the sparseness criterium can be found, but the cost will be that an often too large error is introduced.

Basic Matching Pursuit algorithm:

1. Initialize: $\mathbf{r} := \mathbf{x}$, $\mathbf{w} := \mathbf{0}$
2. Find the inner products: $\mathbf{u} := \mathbf{F}^T \cdot \mathbf{r}$
3. Find k such that $|u(k)| = \max_i |u(i)|$
4. Update weight k: $w(k) := w(k) + u(k)$
5. Update residual: $\mathbf{r} := \mathbf{r} - u(k) \cdot \mathbf{f}_k$
6. Repeat step 2-5 until $\mathbf{w}$ has s non-zero entries.

Figure 1.2: The BMP algorithm for a uniform frame. The stop criterium in step 6 could also be that a predefined number of iterations, for example s, is done or that the norm of the error is smaller than a given limit.

Another approach to solve the vector selection problem is the greedy algorithms collectively referred to as Matching Pursuit (MP) algorithms. These algorithms select one vector from $\mathbf{F}$ at each iteration, and then in the next iteration another vector is selected. The algorithms are quite similar to each other, and the framework in Figure 1.2 can, with some modifications, be used for all these algorithms. In each iteration the approximation is found as a linear combination of the selected frame vectors, and the error is the difference between the original vector and the approximation. The error is used in the next iteration to find the new frame vector. The *stop criterion* for the iterations can be that a certain number of iterations have been executed, that the error is below a pre-defined limit, or that a specified number of non-zero weights has been used. The different variants of MP are

BMP *Basic Matching Pursuit* [44] is the simplest of these algorithms, Figure 1.2. It is sometimes also called only Matching Pursuit (MP). In this algorithm the error is orthogonal to the most recently selected frame vector, but not necessarily to all the previous selected vectors. This means that a vector already selected may be selected again, then the weight for this frame vector is adjusted but no new frame vector is selected. The error converges to zero but it may converge slowly. The advantage is that it is simple and quite fast. For analysis of this algorithm see [44] or [18]. The small example problem used 546 flops.

OMP *Orthogonal Matching Pursuit* [55] [19] is sometimes also called Modified Matching Pursuit (MMP). In this algorithm the error is orthogonal to all the previously selected vectors, the approximation is the signal

projected onto the space spanned by these vectors. This ensures that a new vector is selected at each iteration, and that the error is reduced to zero after N iterations. Orthogonalization makes this algorithm computationally more complex than MP. Fast implementations based on QR-decomposition exist, these are faster than calculating and applying the projection matrix for every iteration. For the small example problem 2386 flops was used by a straight forward (no QR-decomposition) Matlab implementation.

ORMP *Order Recursive Matching Pursuit* also orthogonalize onto the selected vectors, but it is the frame vectors presently unused in the approximation that are orthogonalized and not the residual as in OMP. Since orthogonalization is done in this algorithm too, it too is sometimes called Orthogonal Matching Pursuit. This is confusing but ORMP is essentially different from OMP and often selects other vectors usually giving a better approximation than OMP, and the two algorithms should be distinguished. Gharavi [31] developed a fast algorithm for ORMP and called it Fast (or Fully) Orthogonal Matching Pursuit (FOMP). For the small example problem 781 flops was used.

In Chapter 4 it is discussed how these algorithms can be adapted and used when designing block-oriented and overlapping frames. Also more details of vector selection will be discussed there.

1.4 The scope and contributions of this thesis

The focus of this dissertation is on the *design of frames* for *sparse signal representation*. The design method for block-oriented frames, introduced in [23], is analyzed using the compact notation of linear algebra. One step in the frame design method is to update the frame when the weights, computed with the given frame and on a *training* set of signal vectors representative of the signals to be represented, are fixed. The linear algebra approach reveals the true nature of this problem, as both the derivation of the solution and the solution itself is compactly expressed. With this understanding of the problem, extensions to overlapping frames and the inclusion of constrains (dependencies) on the frame elements and also extensions to two-dimensional and multi-dimensional signals are made. This greatly increases the possibilities of the frame design method. Another step of the frame design method is to update the weights, using the same set of training vectors as in the step above, when the frame is fixed. Here we assess the many different vector selection

algorithms available and especially look at their capabilities in the proposed frame design method. Some modifications to these algorithms are suggested to improve their capabilities in the present context.

Examples which illustrate the sparse representation properties of different frame structures are presented both for one-dimensional signals and for two-dimensional signals. One new application of the sparse representation is given: Promising results were found using the frame approach to do texture classification.

In the end of this work we try to quantify the properties of a frame, we use established frame properties and also propose some new ones. Even if these properties give a quantitative, and partly informative, description of a frame, we did not succeed, as well as we hoped, in establishing a clear connection from these properties to the sparse representation capabilities or to the texture discrimination capabilities of a frame. Consequently, Chapter 7 of the thesis should be viewed as a first step in what we believe will be a process of developing useful tools for frame analysis.

Briefly summarized, the major contributions of this thesis are:

- Using the notation of linear algebra, new insight is gained for the frame design problem. A compact solution to the "find new frame"-step is derived in a simple way.
- Solutions to the "find new frame"-step is derived for the overlapping frame and the general frame with constrains on the frame elements. These derivations are conceptually easier, more compactly expressed, and the solutions are more general than those presented previously. The frame structure possesses the ultimate flexibility in terms of enabling the specification of degree of overcompleteness (redundancy) of the frame to be designed and controlling the resolution capabilities of the signal expansions.
- The frame design methods for the one-dimensional signal are extended to two- and multi-dimensional signals.
- Adaptations to the vector selection algorithms are given. These make the algorithms better suited for both the frame design method and the sparse representation of signals using block-oriented and overlapping frames.
- The experiments show that sparse representations using frames can be done with smaller errors, for the same sparseness factor, than using traditional signal expansions, both for one-dimensional signals (electrocardiograms) and images.

- Texture classification using frames shows excellent overall performance, for many test images the number of wrongly classified pixels is more than halved, in comparison to state of the art texture classification methods presented in [59].
- Frames are analyzed from a practical viewpoint, rather than in a mathematical theoretic perspective. As a result of this, some new frame properties are suggested. So far, the new insight this has given has been moderate, but we think that this approach may be useful in frame analysis in the future.

Chapter 2

Linear Algebra Approach to Frame Design

One of the main topics in signal processing the last fifty years has been the design of signal expansions, like the expansions introduced in Section 1.1. Most of the work has been concentrated on complete expansions, transforms and filter banks, especially the orthogonal variants. These expansions are often adapted to the statistics of the signal class of interest. For the overcomplete expansions, the most desired feature of the expansion method has often been computational effectiveness for the many inner-products between the frame vectors and the signal. This is especially important when the degree of overcompleteness is large. This leads to frames built in a systematic manner, selected to get good time-frequency resolution and effective implementation. Examples are Gabor functions [51], [4], [3] filter banks and wavelet trees [73], and Gaussian chirps [33]. Chou et al. designed a frame by adapting it to a training sequence using techniques of the shape-gain product vector quantizer [13], this frame is overcomplete by a factor of 400.

Frames for sparse signal representations may be designed using an iterative method with two main steps: (1) Frame vector selection and expansion coefficient determination for signals in a *training set*, – selected to be representative of the signals for which compact representations are desired, using the frame designed in the previous iteration. (2) Update of frame vectors with the objective of improving the representation of step (1). This method for frame design was used by Engan et al. [24] for block-oriented signal expansions, i.e. generalizations of block-oriented transforms, and by Aase et al. [1] for non-block-oriented frames, – for short *overlapping frames*, that may be viewed

as generalizations of critically sampled filter banks. Here we solve the *general frame design problem* using the compact notation of linear algebra. This makes the solution both conceptually and computationally easier, especially for the overlapping frame case. Also, the solution is more general than those presented earlier, facilitating the imposition of constraints, such as symmetry, on the designed frame vectors. We show that adjusting the coefficients of the frame, step (2), is a linear problem and that in all cases it may be formulated by the linear equation $\mathbf{Ax} = \mathbf{b}$, the solution is then compactly expressed in the known variables. Much of this work was presented in [64].

2.1 Problem formulation

The frame design methodology presented here was first used in the context of block-oriented frame design [24]. The frame should be adapted to a class of signals, represented by a large set of training vectors, $\{\mathbf{x}_l\}_{l=1}^{L}$, in a way that makes the frame well suited for a sparse representation for this class of signals. The training vectors are consecutive blocks of a training signal. For our frame design method it is convenient to collect the training vectors, the weight vectors and the synthesis vectors into matrices,

$$\begin{array}{rcl} \mathbf{X} & = & [\ \mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \cdots \quad \mathbf{x}_L\ ], \\ \mathbf{W} & = & [\ \mathbf{w}_1 \quad \mathbf{w}_2 \quad \mathbf{w}_3 \quad \cdots \quad \mathbf{w}_L\ ], \\ \mathbf{F} & = & [\ \mathbf{f}_1 \quad \mathbf{f}_2 \quad \cdots \quad \mathbf{f}_K\ ]. \end{array} \tag{2.1}$$

The synthesis equation may now be written as

$$\tilde{\mathbf{X}} = \mathbf{F}\,\mathbf{W}. \tag{2.2}$$

Frame design, or the problem of seeking the optimal frame for a given class of signals and a given sparseness factor, is briefly summarized below. More details can be found in [24]. The objective is to find the frame, $\mathbf{F}$, that minimizes the approximation error expressed as:

$$J(\mathbf{F}, \mathbf{W}) = \|\mathbf{X} - \tilde{\mathbf{X}}\|^2 = \|\mathbf{X} - \mathbf{FW}\|^2, \tag{2.3}$$

subject to a sparsity constraint on $\mathbf{W}$.

Finding the optimal solution to this problem is difficult. A practical optimization strategy, not necessarily leading to a global optimum, but with established good performance [24], [1], can be found by splitting the problem into two parts which are alternately solved within an iterative loop. The method is inspired by the generalized Lloyd algorithm [28] and can in fact be interpreted as a generalization of this algorithm.

The approach starts with a user-supplied initial frame $\mathbf{F}^{(0)}$ and then proceeds to improve it by iteratively repeating two main steps, and a third optional step, using the training signals. The i-th iteration can be described as:

1. $\mathbf{W}^{(i)}$ is found by vector selection and weight computation based on frame $\mathbf{F}^{(i)}$, where the objective function is $J(\mathbf{W}) = \|\mathbf{X} - \mathbf{F}^{(i)}\mathbf{W}\|^2$. Note that a sparseness constraint is imposed on $\mathbf{W}$. This problem is known to be NP-hard [18], [50]. Nevertheless several practical approaches employing *matching pursuit* algorithms, Section 1.3, are known to work well in the vector selection and weight computation. In the present work we usually employ, as the core part of vector selection, an order recursive matching pursuit algorithm described in [31].

2. $\mathbf{F}^{(i+1)}$ is found from $\mathbf{X}$ and $\mathbf{W}^{(i)}$, where the objective function is $J(\mathbf{F}) = \|\mathbf{X} - \mathbf{F}\mathbf{W}^{(i)}\|^2$.

3. The synthesis vectors of $\mathbf{F}^{(i+1)}$ are normalized and the weights in $\mathbf{W}^{(i)}$ are adjusted such that the reconstructed signal is exactly the same as before normalization, i.e. $\mathbf{F}^{(i+1)}\mathbf{W}^{(i)}$ is unchanged. The iteration number, i, is incremented and we proceed again with step 1 above unless some stopping criterion is satisfied.

In this chapter, we address and solve the problem in step 2, i.e. we find the expression for the optimal $\mathbf{F}$ based on the signals in the training set and the weights computed in step 1. The method is generalized to two or more dimensions in Chapter 3. The problem in step 1 is discussed in Chapter 4.

2.2 Design of block-oriented frames

A one-dimensional signal, represented as a column vector $\mathbf{x}$ of length NL, can be divided into L length N consecutive blocks, denoted $\{\mathbf{x}_l\}_{l=1}^{L}$. An approximation to a signal block based on a sparse representation is formed as a linear combination of a few of the synthesis vectors, Equation 1.3. Matrices

are used to represent the signal, the weights, and the frame vectors or synthesis vectors, as in Equation 2.1. Transposing the objective function, Equation 2.3, gives

$$J(\mathbf{F}) = \|\mathbf{X}^T - \mathbf{W}^T\mathbf{F}^T\|^2. \tag{2.4}$$

We denote the n-th column of $\mathbf{F}^T$ as $\overline{\mathbf{f}}_n$ and use the same overline notation for the columns of $\mathbf{X}^T$. Since $\mathbf{X}$ is a matrix where the columns are consecutive blocks of the original signal, the $\overline{\mathbf{x}}_n$ vectors have elements given by L samples of the n-th polyphase component [71] of the original signal. Using these notational conventions, the frame $\mathbf{F}$ can be written as

$$\mathbf{F} = [\mathbf{f}_1 \cdots \mathbf{f}_K] = \begin{bmatrix} \overline{\mathbf{f}}_1^T \\ \vdots \\ \overline{\mathbf{f}}_N^T \end{bmatrix}, \tag{2.5}$$

and we can write Equation 2.4 as

$$J(\mathbf{F}) = \|[\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_N] - \mathbf{W}^T[\overline{\mathbf{f}}_1, \ldots, \overline{\mathbf{f}}_N]\|^2. \tag{2.6}$$

Using a property of the trace norm, $\|\mathbf{A}\|^2 = \sum_j \|\mathbf{a}_j\|^2$ where $\mathbf{a}_j$ is a column of the $\mathbf{A}$ matrix, Equation 2.6 can be written as the sum of N terms:

$$J(\mathbf{F}) = \sum_{n=1}^{N} \|\overline{\mathbf{x}}_n - \mathbf{W}^T\overline{\mathbf{f}}_n\|^2. \tag{2.7}$$

Thus, minimizing $J(\mathbf{F})$ corresponds to minimizing each term of Equation 2.7 separately, giving rise to N separate least squares problems. The overdetermined (more equations than unknowns, here L is larger than K) linear equation system is well known in linear algebra, it is usually written $\mathbf{Ax} = \mathbf{b}$, but using the symbols of Equation 2.7 it is written $\mathbf{W}^T\overline{\mathbf{f}}_n = \overline{\mathbf{x}}_n$. The solutions of these N problems are given by [67]

$$\overline{\mathbf{f}}_n = (\mathbf{W}\mathbf{W}^T)^{-1}\mathbf{W}\overline{\mathbf{x}}_n \qquad n = 1, 2, \ldots, N. \tag{2.8}$$

This may more compactly be written as

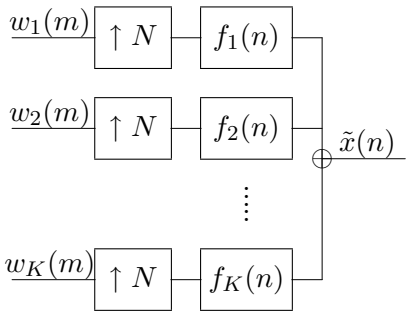


Figure 2.1: A uniform synthesis filter bank of K filters, upsampling factor is N for each filter. When $K > N$ the filter bank is oversampled. The filters are assumed to be of the same length, PN.

$$\begin{aligned} \mathbf{F}^T &= (\mathbf{W}\mathbf{W}^T)^{-1}\mathbf{W}\mathbf{X}^T \qquad \text{or} \\ \mathbf{F} &= \mathbf{X}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}. \end{aligned} \tag{2.9}$$

We note that each row of $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}\mathbf{W}$ is formed by projecting the corresponding row of $\mathbf{X}$ onto the space spanned by the rows of $\mathbf{W}$. The solution in Equation 2.9 is equivalent to the solution given in [24]. The solution may also be written using the pseudoinverse of $\mathbf{W}$, $\mathbf{W}^\dagger = \mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}$, this gives the simple form: $\mathbf{F} = \mathbf{X}\mathbf{W}^\dagger$.

The solutions in Equation 2.8 and Equation 2.9 assume that $\mathbf{W}^T$ has full rank, else the inverse of $(\mathbf{W}\mathbf{W}^T)$ does not exist. Experiments done have shown that this hardly ever occur. The cases where this happens will be when the set of training vectors is too small (normally it should be at least $L > 5K$) and a synthesis vector of $\mathbf{F}$ is so that it will not be used for the sparse representation of *any* of the training vectors, then a row of $\mathbf{W}$ will consist of zeros only and $\mathbf{W}^T$ will be rank deficient. If this happens the solution can be found by removing the unused synthesis vector from $\mathbf{F}$ and the zero row from $\mathbf{W}$ and solve the equation system for the rest of the frame vectors. The unused frame vector may be replaced by a random vector, or another frame vector (the most used one) with an addition of a small distortion, this will hopefully cause it to be used during the vector selection step in the next iteration.

2.3 Design of overlapping frames

A uniform synthesis K channel filter bank is shown in Figure 2.1. The input-output relation can be verified to correspond to a matrix vector product

$$\tilde{\mathbf{x}} = \mathcal{F}\mathbf{w}, \tag{2.10}$$

where the central part can be expanded like in Equation 1.10. The matrix $\mathcal{F}$ or $\mathbf{F}$ is what we here call an overlapping frame. Assuming synthesis filter lengths given by N times an integer P, i.e. NP, it is convenient to partition $\mathbf{F}$ into P submatrices[1], $\{\mathbf{F}_p\}_{p=0}^{P-1}$,

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_0 \\ \vdots \\ \mathbf{F}_{P-1} \end{bmatrix}, \tag{2.11}$$

where each submatrix has size $N \times K$. We refer to P as the overlap factor. Referring to [58] we can rewrite Equation 2.10 to reveal the block structure of $\mathcal{F}$ explicitly. Below we illustrate this for the case when $P = 3$ and with a causality assumption, $\mathbf{x}_l = \mathbf{0}$ and $\mathbf{w}_l = \mathbf{0}$ when $l < 1$ and $l > L$.

$$\begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \vdots \\ \tilde{\mathbf{x}}_{L-1} \\ \tilde{\mathbf{x}}_L \end{bmatrix} = \begin{bmatrix} \boxed{\begin{matrix}\mathbf{F}_0\\ \mathbf{F}_1\\ \mathbf{F}_2\end{matrix}} & & & \\ & \boxed{\begin{matrix}\mathbf{F}_0\\ \mathbf{F}_1\\ \mathbf{F}_2\end{matrix}} & & \\ & & \ddots & \\ & & & \boxed{\begin{matrix}\mathbf{F}_0\\ \mathbf{F}_1\end{matrix}}\ \boxed{\mathbf{F}_0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_{L-1} \\ \mathbf{w}_L \end{bmatrix}. \tag{2.12}$$

At each repetition of $\mathbf{F}$ in $\mathcal{F}$, $\mathbf{F}$ is moved N positions down and K positions to the right. Note the resulting *"overlap-structure"* of $\mathcal{F}$, which is a direct consequence of the channel filters having length larger than the upsampling factor N.

The synthesis equation for a signal block can now be written in terms of the sub-matrices of $\mathbf{F}$:

$$\tilde{\mathbf{x}}_l = \sum_{p=0}^{P-1} \mathbf{F}_p \mathbf{w}_{l-p} = \mathbf{F}_0\mathbf{w}_l + \mathbf{F}_1\mathbf{w}_{l-1} + \ldots + \mathbf{F}_{P-1}\mathbf{w}_{l-P+1}. \tag{2.13}$$

Equation 2.13 can conveniently be written as follows:

[1] This corresponds to the polyphase representation of the oversampled synthesis filter bank, $\mathbf{G}(z)$ in [15] and $\mathbf{R}(z)$ in [8]. In fact, the polyphase matrix is $\mathbf{R}(z) = \sum_{p=0}^{P-1} \mathbf{F}_p z^{-p}$.

$$\tilde{\mathbf{x}}_l = [\mathbf{F}_0, \mathbf{F}_1, \cdots, \mathbf{F}_{P-1}] \begin{bmatrix} \mathbf{w}_l \\ \mathbf{w}_{l-1} \\ \vdots \\ \mathbf{w}_{l-P+1} \end{bmatrix}. \tag{2.14}$$

In the following it will also be convenient to express a composite synthesis equation for all the signal blocks, 1 through L:

$$[\tilde{\mathbf{x}}_1, \cdots \tilde{\mathbf{x}}_l \cdots \tilde{\mathbf{x}}_L] = [\mathbf{F}_0, \mathbf{F}_1, \cdots, \mathbf{F}_{P-1}] \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_l & \cdots & \mathbf{w}_L \\ \mathbf{w}_0 & \cdots & \mathbf{w}_{l-1} & \cdots & \mathbf{w}_{L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{w}_{-P+1} & \cdots & \mathbf{w}_{l-P+1} & \cdots & \mathbf{w}_{L-P+1} \end{bmatrix}.$$

Based on this, with obvious definitions, and in analogy with Equation 1.6 we establish

$$\tilde{\mathbf{X}} = \mathbf{F}^* \mathbf{W}^*. \tag{2.15}$$

Note that with a causality assumption, as shown in Equation 2.12, the lower left part (the part below the "diagonal" made by the $\mathbf{w}_1$ vectors) of $\mathbf{W}^*$ consists of only zeros.

It is also convenient to define P (horizontal) partitions of $\mathbf{W}^*$ as follows:

$$\overrightarrow{\mathbf{W}}^p = [\mathbf{w}_{-p+1}, \mathbf{w}_{-p+2}, \cdots \mathbf{w}_{L-p}], \;\; p = 0, \ldots P-1, \tag{2.16}$$

in which case we can also write

$$\tilde{\mathbf{X}} = \sum_{p=0}^{P-1} \mathbf{F}_p \overrightarrow{\mathbf{W}}^p. \tag{2.17}$$

Note that the exact contents of the first few columns of $\overrightarrow{\mathbf{W}}^p$, i.e. those columns $\mathbf{w}_j$ with $j < 1$, depends on assumptions on the signal outside the range given by $\{\mathbf{x}_l\}_{l=1}^{L}$. In practice L, the number of blocks in the training signal set, is in the order of several thousands, implying that whatever assumptions are made

on the signal outside the training set is of minor importance. Since we, in our implementation of the design method, have assumed a periodic extension [58] of the training signal set we point out that in this case the definition of Equation 2.16 should be modified to read

$$\overrightarrow{\mathbf{W}}^p = [\mathbf{w}_{L-p+1}, \cdots, \mathbf{w}_1, \cdots, \mathbf{w}_{L-p}],\ p = 0, \ldots P-1. \tag{2.18}$$

In this case $\overrightarrow{\mathbf{W}}^p$ is the $\mathbf{W}$ matrix of Equation 2.1 where the columns are *circularly* shifted p positions to the right. With this signal extension the $\mathcal{F}$ matrix with overlap factor $P = 3$ becomes

$$\mathcal{F} = \begin{bmatrix} \mathbf{F}_0 & & & & \mathbf{F}_2 & \mathbf{F}_1 \\ \mathbf{F}_1 & \mathbf{F}_0 & & & & \mathbf{F}_2 \\ \mathbf{F}_2 & \mathbf{F}_1 & \ddots & & & \\ & \mathbf{F}_2 & \ddots & \mathbf{F}_0 & & \\ & & \ddots & \mathbf{F}_1 & \mathbf{F}_0 & \\ & & & \mathbf{F}_2 & \mathbf{F}_1 & \mathbf{F}_0 \end{bmatrix}. \tag{2.19}$$

As for the block-oriented frame the aim for step two in the frame design method is to find the optimal frame, $\mathbf{F}$, given some weights, $\mathbf{W}$. As a consequence of Equation 2.15, the design problem for overlapping frames can now be cast in exactly the same form as the problem in the block-oriented case. Thus, our objective function is now

$$\begin{aligned} J(\mathbf{F}^*) &= \|\mathbf{X} - \tilde{\mathbf{X}}\|^2 \\ &= \|\mathbf{X} - \mathbf{F}^*\mathbf{W}^*\|^2 \\ &= \|\mathbf{X}^T - \mathbf{W}^{*T}\mathbf{F}^{*T}\|^2 \qquad (2.20) \\ &= \|[\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_N] - \mathbf{W}^{*T}[\overline{\mathbf{f}}_1^*, \ldots, \overline{\mathbf{f}}_N^*]\|^2 \\ &= \sum_{n=1}^{N} \|\overline{\mathbf{x}}_n - \mathbf{W}^{*T}\overline{\mathbf{f}}_n^*\|^2 \qquad (2.21) \end{aligned}$$

where $(\overline{\mathbf{f}}_n^*)^T$ is a row of $\mathbf{F}^*$. The solution, – in complete analogy with Section 2.2, is

$$\overline{\mathbf{f}}_n^* = (\mathbf{W}^*\mathbf{W}^{*T})^{-1}\mathbf{W}^*\overline{\mathbf{x}}_n \quad n = 1, \ldots, N. \tag{2.22}$$

This may more compactly be written as

$$\begin{aligned} \mathbf{F}^{*T} &= (\mathbf{W}^*\mathbf{W}^{*T})^{-1}\mathbf{W}^*\mathbf{X}^T \qquad \text{or} \\ \mathbf{F}^* &= \mathbf{X}\mathbf{W}^{*T}(\mathbf{W}^*\mathbf{W}^{*T})^{-1}. \end{aligned} \tag{2.23}$$

The solution in Equation 2.23 is equivalent to the solution given in [1], but the derivation is simpler and the solution is conceptually easier to understand. This way of expressing the solution also facilitates the uncovering of structures in the matrix to be inverted, $\mathbf{C}^* = \mathbf{W}^*\mathbf{W}^{*T}$. These structures, identified below, have proven useful in devising a computationally efficient implementation of the design method.

We have

$$\begin{aligned} \mathbf{C}^* &= \mathbf{W}^*\mathbf{W}^{*T} \\ &= \begin{bmatrix} \overrightarrow{\mathbf{W}}^0 \\ \vdots \\ \overrightarrow{\mathbf{W}}^{P-1} \end{bmatrix} \begin{bmatrix} (\overrightarrow{\mathbf{W}}^0)^T & \cdots & (\overrightarrow{\mathbf{W}}^{P-1})^T \end{bmatrix}. \end{aligned} \tag{2.24}$$

A block of this matrix is $\overrightarrow{\mathbf{W}}^{p_1}(\overrightarrow{\mathbf{W}}^{p_2})^T$, where $p_1, p_2 \in \{0, 1, \ldots, P-1\}$. When periodic extension of the training signal set is assumed, i.e. when using the definition in Equation 2.18, this matrix block only depends on the relative difference of p_1 and p_2[2]. Letting $p = p_2 - p_1$ we define

$$\mathbf{C}_p = \overrightarrow{\mathbf{W}}^m(\overrightarrow{\mathbf{W}}^{m+p})^T. \tag{2.25}$$

This is valid for all values of m, letting $m = p$ it is easily seen that $\mathbf{C}_{-p} = \mathbf{C}_p^T$. Thus, the matrix to be inverted can be written

$$\mathbf{C}^* = \begin{bmatrix} \mathbf{C}_0 & \mathbf{C}_1 & \cdots & \mathbf{C}_{P-1} \\ \mathbf{C}_1^T & \mathbf{C}_0 & \cdots & \mathbf{C}_{P-2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{P-1}^T & \mathbf{C}_{P-2}^T & \cdots & \mathbf{C}_0 \end{bmatrix}. \tag{2.26}$$

[2]For other reasonable assumptions on the signal outside the training set, this, and the following statements, are approximately true.

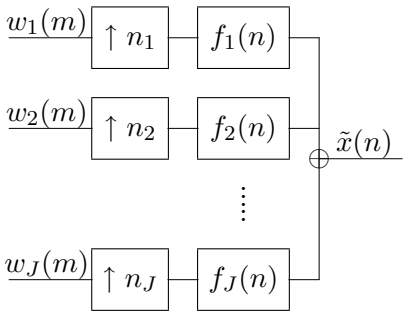


Figure 2.2: A general synthesis filter bank of J filters. The filter length, l_j, and upsampling factor, n_j, may be different for each filter.

Each element of $\mathbf{C}_p$ is an inner product of two (sparse) vectors

$$[\mathbf{C}_p]_{ij} = (\overline{\mathbf{w}}_i^*)^T(\overline{\mathbf{w}}_{pN+j}^*), \tag{2.27}$$

where $(\overline{\mathbf{w}}_i^*)^T$ is row i of the matrix $\mathbf{W}^*$. Also, the elements of the $N \times KP$ matrix $\mathbf{X}\mathbf{W}^{*T}$, which we denote $\mathbf{D}$, may be calculated in a similar manner

$$[\mathbf{D}]_{ij} = [\mathbf{X}\mathbf{W}^{*T}]_{ij} = (\overline{\mathbf{x}}_i)^T(\overline{\mathbf{w}}_j^*). \tag{2.28}$$

Having found $\mathbf{C}^*$ and $\mathbf{D}$ the solution is simply computed as

$$\mathbf{F}^* = \mathbf{D}(\mathbf{C}^*)^{-1}. \tag{2.29}$$

2.4 General overlapping frames

As pointed out previously, the frame vectors of Section 2.3 correspond to the filter unit pulse responses of the synthesis filter bank of Figure 2.1. This filter bank is restricted in the sense that all channels have the same upsampling factor, and the channel filters are all of the same length, NP. A more general synthesis filter bank structure with different upsampling ratios and different filter lengths is illustrated in Figure 2.2. This filter bank has J different FIR

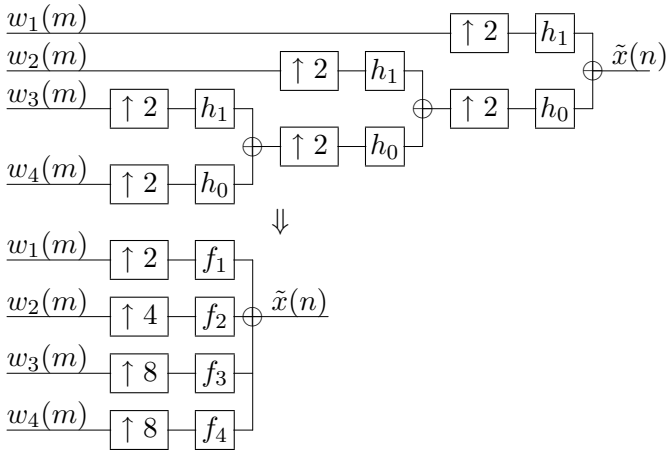


Figure 2.3: A wavelet tree filter bank, here with three levels, can be implemented as a general filter bank with one level. When the length of the high-pass filter, h_1, is 7 and the length of the low-pass filter, h_0, is 9, the derived filters f_1 (=h_1), f_2, f_3 and f_4 will have the lengths 7, 21, 49 and 57, respectively.

channel filters. The length of the filter for channel j is denoted by l_j and the upsampling factor by n_j. As detailed in [1] this structure encompasses every conceivable transform, filter bank, and wavelet decomposition expansion along with their generalizations, some possible support structures are shown in Figure 1.1. In fact this structure possesses the ultimate flexibility in terms of enabling the specification of degree of overcompleteness (redundancy) of the frame to be designed through the selection of the n_js. Also the resolution capabilities of the signal expansions can be controlled through the selection of the l_js.

To make a frame corresponding to this general filter bank structure we can proceed as outlined in the beginning of Section 2.3, by formulating the input-output relations of the various channels and collecting them using appropriately defined matrix/vector quantities. Carrying out this quite laborious, but straightforward, task, we find that the structure of the synthesis equation as given in Equation 2.12 can be maintained. We illustrate with an example: Given the dyadic, – wavelet-like, synthesis structure in the upper part of Figure 2.3. This structure is equivalent to the one shown in the lower part of the same figure when $f_1(n), \ldots, f_4(n)$ are determined from $h_0(n)$ and $h_1(n)$ in an appropriate manner [71]. The structure for the $\mathcal{F}$ matrix of Equation 2.12 in this case is shown in Figure 2.3. The following features of the figure are

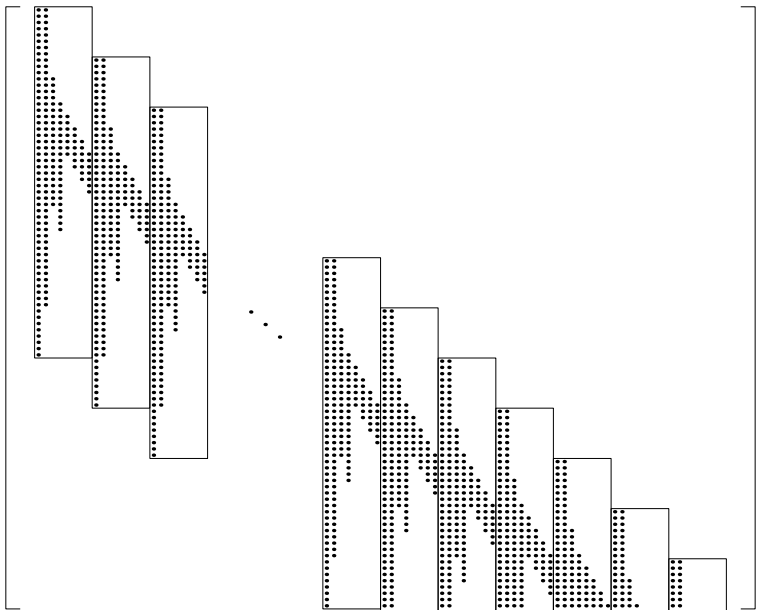


Figure 2.4: The structure of the $\mathcal{F}$ matrix, Equation 2.12, for a filter bank as in Figure 2.3. Each box indicate the $\mathbf{F}$ matrix, and the dots indicate the non-zero values in $\mathbf{F}$. The first column in $\mathbf{F}$ is for the f_4 filter, the second column for the f_3 filter, the third and fourth columns are for the f_2 filter, and columns 5 to 8 are for the f_1 filter. In this example $N = 8$, $K = 8$ and $P = 7$. The observant reader will notice that this structure is the same as the wavelet structure in the rightmost part of Figure 1.1, with one small exception: The last entry in the f_4 filter is ignored, using values form the Daubechies 9-7 biorthogonal filters in the wavelet filter bank this entry is very close to zero anyway. The difference in vertical alignment of the frame vectors in the $\mathbf{F}$ matrix is the same as a permutation of the columns of the $\mathcal{F}$ matrix. For vector selection it is better to have the vertical alignment as in this figure, the energy for the synthesis vectors in $\mathbf{F}$ are better concentrated to one part of the signal.

noteworthy: The first column of dots in each box is for the f_4 filter, the second column for the f_3 filter, the third and fourth columns are for the f_2 filter, and columns 5 to 8 are for the f_1 filter. In this example $N = 8$, $K = 8$ and $P = 7$. From this example we see that a filter where the corresponding upsampling factor is smaller than N will be repeated in N/n_j columns within $\mathbf{F}$, each time moved n_j positions down.

In general, the following modifications to the quantities N, K and P, that collectively determine the structure of $\mathcal{F}$ in Equation 2.10, must be kept in mind:

$$\begin{aligned} N &= \text{least common multiple of } \{n_j\}_{j=1}^{J} \\ K &= \sum_{j=1}^{J} \frac{N}{n_j} \\ P &= \max_j \left\lceil \frac{l_j - n_j}{N} \right\rceil + 1 \end{aligned} \tag{2.30}$$

where $\lceil x \rceil$ is the smallest integer larger or equal to x. Depending on the desired lengths, l_j, and the upsampling factors, n_j, the $\mathbf{F}$ matrix will be populated by a combination of elements of the frame vectors and zeros. Those zeros can be interpreted as constraints on the $\mathbf{F}$ matrix. Obviously these constraints must be embedded into the frame design method.

In many design problems in signal processing, the imposition of various symmetries play an important role. For example, in the design of filters for critically sampled filter banks we may desire filters with linear phase, i.e. unit pulse responses that are symmetric with respect to their midpoints. It is conceivably desirable to impose various symmetry constraints on the frame vectors. Such symmetries can be expressed by relations between pairs of elements of frame vectors of type $f(i) = af(j)$, where we have assumed that the elements of all frame vectors are indexed sequentially. Most often a will be given by 1 (to specify even symmetries) or -1 (to specify odd symmetries). In the following we reformulate the design problem presented previously in such a way as to facilitate the incorporation of the two types of constraints described above.

Recall that what we have done in Section 2.3, is to pose the problem as that of finding a least squares solution to the overdetermined set of equations

$$\mathbf{W}^{*T}\mathbf{F}^{*T} = \mathbf{X}^T. \tag{2.31}$$

The solution, which is given in Equation 2.23, is determined by solving the corresponding normal equations

$$(\mathbf{W}^*\mathbf{W}^{*T})\mathbf{F}^{*T} = \mathbf{W}^*\mathbf{X}^T. \tag{2.32}$$

Using previously defined quantities this can be written as

$$\begin{bmatrix} \mathbf{C}^* & & & \\ & \mathbf{C}^* & & \\ & & \ddots & \\ & & & \mathbf{C}^* \end{bmatrix} \begin{bmatrix} \overline{\mathbf{f}}_1^* \\ \overline{\mathbf{f}}_2^* \\ \vdots \\ \overline{\mathbf{f}}_N^* \end{bmatrix} = \begin{bmatrix} \mathbf{W}^*\overline{\mathbf{x}}_1 \\ \mathbf{W}^*\overline{\mathbf{x}}_2 \\ \vdots \\ \mathbf{W}^*\overline{\mathbf{x}}_N \end{bmatrix}. \tag{2.33}$$

With obvious definitions, this can compactly be expressed as

$$\mathbf{C}\mathbf{f} = \mathbf{d}, \tag{2.34}$$

where $\mathbf{f}$ is the vector of concatenated columns of $\mathbf{F}^{*T}$. It is also instructive to observe that the overdetermined set of equations, Equation 2.31, giving rise to Equation 2.33, can be written as

$$\begin{bmatrix} \mathbf{W}^{*T} & & & \\ & \mathbf{W}^{*T} & & \\ & & \ddots & \\ & & & \mathbf{W}^{*T} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{f}}_1^* \\ \overline{\mathbf{f}}_2^* \\ \vdots \\ \overline{\mathbf{f}}_N^* \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{x}}_1 \\ \overline{\mathbf{x}}_2 \\ \vdots \\ \overline{\mathbf{x}}_N \end{bmatrix}. \tag{2.35}$$

And again, with obvious definitions, this can compactly be expressed as

$$\mathcal{W}\mathbf{f} = \overline{\mathbf{x}}. \tag{2.36}$$

The large matrix $\mathcal{W}$ (size $NL \times NKP$) is the "square root factor" of the $\mathbf{C}$ matrix in Equation 2.34, i.e. $\mathbf{C} = \mathcal{W}^T\mathcal{W}$. Given the above, we are in a position to precisely explain the implications of the previously mentioned constraints on the problem:

1. If an element of $\mathbf{f}$ is forced to zero, i.e. $f(i) = 0$, this has the consequence of removing variable $f(i)$ in the equation set, Equation 2.36, and deleting one column of $\mathcal{W}$. Thus the problem is formulated in terms of $\tilde{\mathcal{W}}$, which is the same matrix as $\mathcal{W}$, but with column no. i removed. The $\mathbf{C}$ in Equation 2.34 is consequently replaced by $\tilde{\mathbf{C}} = \tilde{\mathcal{W}}^T\tilde{\mathcal{W}}$. A little reflection reveals that $\tilde{\mathbf{C}}$ is the same as $\mathbf{C}$, but with row no. i *and* column no. i removed. Also, the $\mathbf{d}$ in Equation 2.34 is replaced by $\tilde{\mathbf{d}}$, the same as $\mathbf{d}$, but with element no. i removed.

2. If the relation $f(j) = af(i)$ is imposed on a pair of elements in $\mathbf{f}$, this corresponds to replacing the $\mathcal{W}$ of Equation 2.36 by $\tilde{\mathcal{W}}$ which is found by adding a times column i to column j, and removing column i. The impact of this propagated to $\tilde{\mathbf{C}} = \tilde{\mathcal{W}}^T\tilde{\mathcal{W}}$ and $\tilde{\mathbf{d}}$ is obvious.

The above operations are repeated a number of times consistent with the number and type of constraints imposed by the frame design specification. Note that through these operations the symmetry of $\tilde{\mathbf{C}}$ is conserved and Equation 2.34 is modified. After the necessary operations are done, the size of the $\tilde{\mathbf{C}}$ matrix will be $Q \times Q$ and the size of $\tilde{\mathbf{d}}$ (and $\tilde{\mathbf{f}}$) will be $Q \times 1$, where Q is the number of free variables in $\mathbf{f}$. Solving this modified equation system will then directly give the free variables of $\mathbf{f}$.

Implementation of this is most easily done by generating two mapping tables, one table that maps from the $\mathbf{C}_p$ matrices of Equation 2.27 to the $\tilde{\mathbf{C}}$ matrix of Equation 2.34, and one table that maps from the $\mathbf{D}$ matrix of Equation 2.28 to the $\tilde{\mathbf{d}}$ vector of Equation 2.34. These mappings are completely given by the structure of the overlapping frame, the general filter bank in Figure 2.1.

Notice again the generality of this derivation. It is possible to represent a large number of different support structures by a general filter bank, $\tilde{\mathbf{x}} = \mathcal{F}\mathbf{w}$ where $\mathcal{F}$ is as in Equation 2.10. It is also possible to impose symmetry restrictions on the different filters. A vast set of different configurations may be optimized using the formulation above, including all possible cases for the method presented in [1].

Chapter 3

Generalizations to Two and More Dimensions

The one-dimensional (1D) frame design method in Chapter 2 can also be used on two-dimensional (2D) and multi-dimensional (MD) signals. Conceptually it is very similar but the notation is quite a bit more involved. Even for the 1D signal and an overlapping frame in section 2.3 we have matrices of several dimensions, like $\mathbf{F}$ in Equation 2.11 and $\mathbf{F}^*$ and $\mathbf{W}^*$ in Equation 2.15 which are 3D, and the $\mathbf{C}^*$ in Equation 2.26 is 4D. However, these MD matrices are treated as 2D matrices that have a structure as defined in the equations. For the 2D signal, and even more for the MD signal, this structure gets more complicated and we need a good notation to reflect it. Multi-dimensional matrices are appropriate for this use. We will here introduce this MD matrix notation, and some needed basic operations.

3.1 Notation

A MD matrix is denoted by a not-bold uppercase letter, like A. The notation is illustrated by a simple example: Let F be 3D (size: $N \times K \times P$), like $\mathbf{F}$ in Equation 2.11. An element of of F is denoted $F_{n,k,p}$. A plane cut trough the MD matrix defines a 2D matrix, colons are used to indicate the direction of the plane, we have $F_{:,:,p}$ (size: $N \times K$), $(F_{:,:,p})^T$ (size: $K \times N$) and $F_{:,k,:}$ (size: $N \times P$). Similarly, a line through the MD matrix defines a column vector (no matter the direction of the line), we have $F_{:,k,p}$ (size: $N \times 1$), $F_{n,:,p}$ (size: $K \times 1$), and $(F_{n,:,p})^T$ (size: $1 \times K$) (row vector).

MD matrices of the same size can be element-wise added, and a MD matrix can be multiplied by a scalar. Vector and matrix multiplication are only allowed on 2D submatrices or 1D subvectors of the MD matrices. We define the norm for such MD matrices as the square root of the sum of all the elements squared.

$$\|A\|^2 = \sum_{n_1=1}^{N_1} \cdots \sum_{n_m=1}^{N_m} (A_{n_1,\cdots,n_m})^2 \tag{3.1}$$

where m is the number of dimensions. For 2D matrices this norm corresponds to the trace (Frobenius) norm and for vectors it corresponds to the 2-norm.

Much of the work involved to solve a MD formulated problem is to reshape the MD matrices into 2D matrices. The goal is to express the synthesis equation as a product of 2D matrices, like in Equation 2.2 and Equation 2.15. When this is accomplished, the solution is easy to find.

Let us illustrate a MD matrix by representing an image as a 4D matrix. An image (size: $N_1L_1 \times N_2L_2$) can be divided into blocks of size $N_1 \times N_2$

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{1,1} & \mathbf{X}_{1,2} & \cdots & \mathbf{X}_{1,L_2} \\ \mathbf{X}_{2,1} & \mathbf{X}_{2,2} & \cdots & \mathbf{X}_{2,L_2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{L_1,1} & \mathbf{X}_{L_1,2} & \cdots & \mathbf{X}_{L_1,L_2} \end{bmatrix}. \tag{3.2}$$

The image $\mathbf{X}$ is represented as a matrix of matrices. When it is divided into blocks like this, it is four dimensional and can be represented by the 4D matrix, X (size: $N_1 \times N_2 \times L_1 \times L_2$).

3.2 Block-oriented frame in 2D

An approximation to an image block, $X_{:,:,l_1,l_2}$, is formed as a linear combination of some *frame images*, denoted $\{\mathbf{F}_k\}_{k=1}^K$, each of size $N_1 \times N_2$. We may collect these into a 3D matrix, F (size: $N_1 \times N_2 \times K$), a particular frame image may now be denoted $F_{:,:,k}$. A particular image block indexed by l_1 and l_2 is reconstructed as

$$\tilde{X}_{:,:,l_1,l_2} = \sum_{k=1}^{K} W_{k,l_1,l_2} F_{:,:,k} \tag{3.3}$$

where W is the 3D weight matrix (size: $K \times L_1 \times L_2$). The goal is to express this synthesis equation as a product of ordinary matrices, as Equation 2.2 for the one-dimensional case, to do this the MD matrices should be reshaped or reorganized. The blocks, $X_{:,:,l_1,l_2}$ are indexed by l, where

$$l = l_1 + (l_2 - 1)L_1 \quad \text{and} \quad l \in \{1, 2, \ldots, L\} \quad \text{and} \quad L = L_1 L_2. \tag{3.4}$$

This is also done for the l_1 and l_2 indices in the 3D weight matrix W. Each image block and each frame image (size: $N_1 \times N_2$) can be lexicographically ordered into a vector of length $N = N_1 N_2$ and indexed by n in a similar way

$$n = n_1 + (n_2 - 1)N_1 \quad \text{and} \quad n \in \{1, 2, \ldots, N\} \quad \text{and} \quad N = N_1 N_2. \tag{3.5}$$

This reshaping gives $\mathbf{X}$ (size: $N \times L$), $\mathbf{W}$ (size: $K \times L$) and $\mathbf{F}$ (size: $N \times K$). The synthesis equation is on the desired form

$$\tilde{\mathbf{X}} = \mathbf{F}\mathbf{W}. \tag{3.6}$$

The synthesis equation is exactly as Equation 2.2, and the solution is given by Equation 2.9

$$\mathbf{F} = \mathbf{X}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1} \tag{3.7}$$

The final step should be to reverse the reshape process. This reshaping and rearranging of blocks is the same procedure as the one used for designing frames for image compression in [22].

3.3 Overlapping frame in 2D

To get overlapping frames for 2D signals the frame images should be allowed to overlap each other similar to how the synthesis (frame) vectors overlap each other for the 1D signal. In Section 2.3 the frame was divided into P parts, Equation 2.11, where P is the overlap factor. In Equation 2.12 we see that the synthesis vectors of $\mathbf{F}$ overlap each other, i.e. each reconstructed signal block is the linear combination of different parts of the synthesis vectors. To extend the overlap quality of the frame from 1D to 2D the overlap should be possible in both dimensions, i.e both the vertical and horizontal direction. The overlap factor in vertical direction is denoted P_1 and the overlap factor in horizontal direction is denoted P_2. In complete analogy with the 1D case the overlapping frame in 2D can be represented by a 5D matrix F (size: $N_1 \times N_2 \times K \times P_1 \times P_2$). Each frame image, $\mathbf{F}_k$ (size: $N_1P_1 \times N_2P_2$), is a matrix of matrices. Indexing of the submatrices of the frame images is done like for the overlapping frame $\mathbf{F}$ in Equation 2.11, i.e. from 0 to $(P_1 - 1)$ and 0 to $(P_2 - 1)$. A frame image is the 2D analogy to the 1D frame vector, it can be written

$$\mathbf{F}_k = \begin{bmatrix} F_{:,:,k,0,0} & F_{:,:,k,0,1} & \cdots & F_{:,:,k,0,P_2-1} \\ F_{:,:,k,1,0} & F_{:,:,k,1,1} & \cdots & F_{:,:,k,1,P_2-1} \\ \vdots & \vdots & \ddots & \vdots \\ F_{:,:,k,P_1-1,0} & F_{:,:,k,P_1-1,1} & \cdots & F_{:,:,k,P_1-1,P_2-1} \end{bmatrix}. \tag{3.8}$$

The signal, X (4D), and the weights, W (3D), are as in Section 3.2. Postponing the treatment of the edge problem, the synthesis equation for an image block (not close to the edge) corresponding to Equation 2.13 is now

$$\tilde{X}_{:,:,l_1,l_2} = \sum_{p_1=0}^{P_1-1} \sum_{p_2=0}^{P_2-1} \sum_{k=1}^{K} W_{k,l_1-p_1,l_2-p_2} F_{:,:,k,p_1,p_2}. \tag{3.9}$$

Just as for the overlapping 1D case, each image block is a linear combination of *blocks* of the frame images.

We could now proceed from from Equation 3.9 just as we did from Equation 2.13. Carrying out this task, would be quite laborious and difficult to present in clear equations, but straightforward. We will here jump directly to Equation 2.18, here a periodic extension is assumed and the shifted weight matrices, $\overrightarrow{\mathbf{W}}^p$, are defined for the overlapping 1D case. The shifted weight matrices in the 2D case are defined in a similar way, but now shifts in both directions are needed. Remember that the weight matrix, W, is now 3D and its

size is $K \times L_1 \times L_2$. The shifted weight matrix, denoted $\overrightarrow{W}_{:,:,:,p_1,p_2}$, is defined as the weight matrix, W, shifted p_1 positions forward (down) in the second dimension and p_2 positions forward (right) in the third dimension. Thus we have $\overrightarrow{W}_{:,:,:,0,0} = W$. These shifted matrices are defined for $p_1 = 0, 1, \ldots, P_1 - 1$ and $p_2 = 0, 1, \ldots, P_2 - 1$. This is the most complicated step in this derivation so to illustrate these shifts we give three examples

$$
\begin{aligned}
\overrightarrow{W}_{:,:,:,1,0} &= \begin{bmatrix} W_{:,L_1,1} & W_{:,L_1,2} & \cdots & W_{:,L_1,L_2} \\ W_{:,1,1} & W_{:,1,2} & \cdots & W_{:,1,L_2} \\ \vdots & \vdots & \ddots & \vdots \\ W_{:,L_1-1,1} & W_{:,L_1-1,2} & \cdots & W_{:,L_1-1,L_2} \end{bmatrix} \\
\overrightarrow{W}_{:,:,:,0,1} &= \begin{bmatrix} W_{:,1,L_2} & W_{:,1,1} & \cdots & W_{:,1,L_2-1} \\ W_{:,2,L_2} & W_{:,2,1} & \cdots & W_{:,2,L_2-1} \\ \vdots & \vdots & \ddots & \vdots \\ W_{:,L_1,L_2} & W_{:,L_1,1} & \cdots & W_{:,L_1,L_2-1} \end{bmatrix} \\
\overrightarrow{W}_{:,:,:,1,1} &= \begin{bmatrix} W_{:,L_1,L_2} & W_{:,L_1,1} & \cdots & W_{:,L_1,L_2-1} \\ W_{:,1,L_2} & W_{:,1,1} & \cdots & W_{:,1,L_2-1} \\ \vdots & \vdots & \ddots & \vdots \\ W_{:,L_1-1,L_2} & W_{:,L_1-1,1} & \cdots & W_{:,L_1-1,L_2-1} \end{bmatrix}
\end{aligned}
\tag{3.10}
$$

These $P_1 P_2$ shifted weight matrices are collected into one large 5D matrix $\overrightarrow{W}$. Before the reshaping and reduction of dimensionality we have

$$
\left.
\begin{array}{lll}
X & \text{4D of size} & N_1 \times N_2 \times L_1 \times L_2 \\
\overrightarrow{W} & \text{5D of size} & K \times L_1 \times L_2 \times P_1 \times P_2 \\
F & \text{5D of size} & N_1 \times N_2 \times K \times P_1 \times P_2
\end{array}
\right\}
\tag{3.11}
$$

These matrices are now reshaped, the l and n indices, Equation 3.4 and Equation 3.5, are used, and the p_1 and p_2 indices are replaced by the p index,

$$
p = p_1 + p_2 P_1 \quad \text{and} \quad p \in \{0, 1, \ldots, P-1\} \quad \text{and} \quad P = P_1 P_2. \tag{3.12}
$$

The size of the reshaped matrices will now be $\mathbf{X}$ (size: $N \times L$), $\overrightarrow{W}$ (size: $K \times L \times P$) and F (size: $N \times K \times P$). The "starred" matrices, $\mathbf{F}^*$ and $\mathbf{W}^*$, are defined similar to the way they were used in Equation 2.15

$$\begin{aligned} \mathbf{F}^* &= [F_{:,:,0}, F_{:,:,1}, \cdots, F_{:,:,P-1}] \\ \mathbf{W}^* &= \begin{bmatrix} \overrightarrow{W}_{:,:,0} \\ \overrightarrow{W}_{:,:,1} \\ \vdots \\ \overrightarrow{W}_{:,:,P-1} \end{bmatrix}. \end{aligned} \tag{3.13}$$

The synthesis equation can now be written

$$\tilde{\mathbf{X}} = \sum_{p=0}^{P-1} F_{:,:,p} \overrightarrow{W}_{:,:,p} = \mathbf{F}^* \mathbf{W}^* \tag{3.14}$$

and the solution is like in Equation 2.23

$$\begin{aligned} \mathbf{F}^{*T} &= (\mathbf{W}^* \mathbf{W}^{*T})^{-1} \mathbf{W}^* \mathbf{X}^T \\ \mathbf{F}^* &= \mathbf{X} \mathbf{W}^{*T} (\mathbf{W}^* \mathbf{W}^{*T})^{-1}. \end{aligned} \tag{3.15}$$

Let us look a little bit closer on the matrix to be inverted, we call it $\mathbf{C}^* = \mathbf{W}^* \mathbf{W}^{*T}$, (size: $KP \times KP$). For the 1D case $\mathbf{C}^*$ has a special structure, as shown in Equation 2.26. A similar but not quite that simple structure exist for the 2D case. $\mathbf{C}^*$ may be expressed by a 4D matrix, C (size: $K \times K \times P \times P$).

$$\mathbf{C}^* = \begin{bmatrix} C_{:,:,0,0} & C_{:,:,0,1} & \cdots & C_{:,:,0,P-1} \\ C_{:,:,1,0} & C_{:,:,1,1} & \cdots & C_{:,:,1,P-1} \\ \vdots & \vdots & \ddots & \vdots \\ C_{:,:,P-1,0} & C_{:,:,P-1,1} & \cdots & C_{:,:,P-1,P-1} \end{bmatrix} \tag{3.16}$$

$$C_{:,:,i,j} = \overrightarrow{W}_{:,:,i} (\overrightarrow{W}_{:,:,j})^T, \quad i,j \in \{0,1,\ldots,P-1\} \tag{3.17}$$

where $\overrightarrow{W}$ (size: $K \times L \times P$) is the 5D matrix of Equation 3.11 reshaped into a 3D matrix by $(l_1, l_2 \rightarrow l)$ and $(p_1, p_2 \rightarrow p)$.

$\mathbf{C}^*$ has P^2 submatrices, each of size $K \times K$. Fortunately, as for the 1D signal, many of the matrices in Equation 3.17 are equal (or the transposed of another matrix). The important thing is the *relative translation* between $\overrightarrow{W}_{:,:,i}$ and $\overrightarrow{W}_{:,:,j}$. The 2D translation is not as simple as the 1D translation. It can be shown that the number of different matrices $C_{:,:,i,j}$ which need to be calculated, is $\left(\frac{1}{2}(2P_1-1)(2P_2-1)+\frac{1}{2}\right)$, the general formula for the multi-dimensional case is given in Equation 3.24. For $P_1 = P_2 = 2$ this gives 5 different submatrices, much less than 16. For $P_1 = P_2 = 3$ this gives 13 different submatrices, much less than 81. The matrix for $P_1 = P_2 = 2$ is shown below

$$\mathbf{C}^* = \begin{bmatrix} C_{:,:,0,0} & C_{:,:,0,1} & C_{:,:,0,2} & C_{:,:,0,3} \\ (C_{:,:,0,1})^T & C_{:,:,0,0} & C_{:,:,1,2} & C_{:,:,0,2} \\ (C_{:,:,0,2})^T & (C_{:,:,1,2})^T & C_{:,:,0,0} & C_{:,:,0,1} \\ (C_{:,:,0,3})^T & (C_{:,:,0,2})^T & (C_{:,:,0,1})^T & C_{:,:,0,0} \end{bmatrix} \tag{3.18}$$

To further illustrate the structure in the $\mathbf{C}^*$ matrix, the example for $P_1 = P_2 = 3$ is shown below, only the significant last two indices (ij) of each of the $C_{:,:,i,j}$ matrices are written out, and the lower left part is left out since the matrix is symmetric. The first occurrence of each submatrix is bold

$$i,j = \left[\begin{array}{ccc|ccc|ccc} \mathbf{00} & \mathbf{01} & \mathbf{02} & \mathbf{03} & \mathbf{04} & \mathbf{05} & \mathbf{06} & \mathbf{07} & \mathbf{08} \\ & 00 & 01 & \mathbf{13} & 03 & 04 & \mathbf{16} & 06 & 07 \\ & & 00 & \mathbf{23} & 13 & 03 & \mathbf{26} & 16 & 06 \\ \hline & & & 00 & 01 & 02 & 03 & 04 & 05 \\ & & & & 00 & 01 & 13 & 03 & 04 \\ & & & & & 00 & 23 & 13 & 03 \\ \hline & & & & & & 00 & 01 & 02 \\ & & & & & & & 00 & 01 \\ & & & & & & & & 00 \end{array}\right] \tag{3.19}$$

$\mathbf{C}^*$ is now given by the different matrices $C_{:,:,i,j}$ each of size $K \times K$ and calculated as the matrix product of two sparse (implying that effective algorithms may be used) matrices, Equation 3.17. We can also define matrix $\mathbf{D}$ similar as for the 1D case, $\mathbf{D} = \mathbf{X}\mathbf{W}^{*T}$. As for the 1D signal, Equation 2.29, the solution can be written

$$\mathbf{F}^* = \mathbf{D}(\mathbf{C}^*)^{-1} \tag{3.20}$$

3.4 General overlapping frame in 2D

The general frame can be designed also for a 2D signal using a similar method to that in Section 2.4. Some points where special care should be taken are mentioned below. Especially the indexing problem is difficult to avoid, since the conventional ways of indexing "are consumed" the used notation may seem awkward, for example for the placement of the j index below, but with some care it should be possible to decode the used notation.

The general 2D nonseparable filter bank of J filters corresponds to a synthesis system where we have J different synthesis frame images. The translations of the synthesis filters may vary and it may be different along each of the directions. At each repetition frame image j, of size $l_1^j \times l_2^j$, is moved n_1^j positions down and n_2^j positions to the right. n_1^j and n_2^j is the upsampling factors for filter j in each of the two dimensions. For the overlapping frame the 2D analogy to Equation 2.30 in 1D is

$$\begin{aligned}
N_1 &= \text{least common multiplier of } \{n_1^j\}_{j=1}^J \\
N_2 &= \text{least common multiplier of } \{n_2^j\}_{j=1}^J \\
N &= N_1 N_2 \\
K &= \sum_{j=1}^{J} \frac{N}{n_1^j n_2^j} \\
P_1 &= \max_j \left\lceil \frac{l_1^j - n_1^j}{N} \right\rceil + 1 \\
P_2 &= \max_j \left\lceil \frac{l_2^j - n_2^j}{N} \right\rceil + 1 \\
P &= P_1 P_2
\end{aligned} \tag{3.21}$$

Now the process will be as described before. The matrices $\mathbf{F}^*$ and $\mathbf{W}^*$ are defined as for the overlapping frame in Equation 3.13. The linear equation system is formed as Equation 2.34 was formed, and the dependencies in $\mathbf{f}$ are removed using the steps described in Section 2.4.

3.5 Multi-dimensional signal

The extension from 1D to 2D may easily be further extended to the multi-dimensional (MD) signal. The main problem is that the size of the problem

increase dramatically as more dimensions are introduced. For the time being it is not possible to see any practical use of these methods for more than the 2D signals, possibly with an exception for a small 3D block-oriented frame. However, to make the theory complete we here include some equations for the MD signal.

3.5.1 Block-oriented frame

The signal will be MD and may be divided into blocks similar to a 2D signal Equation 3.2. The sizes of the MD matrices are

$$\left.\begin{array}{lll} X & \text{(2M)D of size} & N_1 \times \cdots \times N_M \times L_1 \times \cdots \times L_M \\ W & \text{(M+1)D of size} & K \times L_1 \times \cdots \times L_M \\ F & \text{(M+1)D of size} & N_1 \times \cdots \times N_M \times K \end{array}\right\} \tag{3.22}$$

Collecting the N-indices together and the L-indices together this is reduced to 2D matrices. The solution is as in Equation 3.7.

3.5.2 Overlapping frame

Also the extension to overlapping frame for MD signals is very similar as the extension from block-oriented to overlapping frame in the 2D case. The sizes of the MD matrices are

$$\left.\begin{array}{lll} X & \text{(2M)D of size} & N_1 \times \cdots \times N_M \times L_1 \times \cdots \times L_M \\ W & \text{(M+1)D of size} & K \times L_1 \times \cdots \times L_M \\ \overrightarrow{W} & \text{(2M+1)D of size} & K \times L_1 \times \cdots \times L_M \times P_1 \times \cdots \times P_M \\ F & \text{(2M+1)D of size} & N_1 \times \cdots \times N_M \times K \times P_1 \times \cdots \times P_M \end{array}\right\} \tag{3.23}$$

Collecting the N-indices together and the L-indices together and the P-indices together this is reduced to 3D matrices and the solution is as in Equation 3.15. The structure of the C^* matrix will also be regular in a way similar to that of the 1D and 2D signals. It will consist of P^2 submatrices, where $P = \prod_{m=1}^{M} P_m$, it will be symmetric, and the structure will be like "Toeplitz matrices within each other", this is illustrated in Equation 3.19 for the 2D case. Each submatrix is the product of two of the submatrices of $\mathbf{W}^*$, and it is the relative position in the MD space that is important. The total number of different submatrices is

$$I_C = \Big(\frac{1}{2}\prod_{m=1}^{M}(2P_m - 1)\Big) + \frac{1}{2} \tag{3.24}$$

where we have also accounted for the symmetry. The extension to the general MD frame will be just as in Section 3.4.

3.6 Non-linear frames

Non-linear in this section means that the synthesis system is not linear in the free variables of the frame elements, a frame element is an entry in the frame matrix. A frame element may be the product of several free variables. The synthesis system is still linear in the expansion coefficients, i.e. the reconstructed signal is still formed as a linear combination of some frame vectors or frame signals. Note that these frames constitute a subset of those possible using previous theory formulations, i.e. frames where all elements are free variables.

Some of the most common signal expansions are of this type, the tree-structured filter bank (wavelet) is a typical example, and the separable transforms and filter banks commonly used in image processing are other examples. One effect of the structure in the synthesis system is that the number of free variables in the frame (transform or filter bank) is reduced. The main advantage is that the structure usually makes computation easier. The non-linear frames will not be thoroughly handled here, but we include one example that illustrate the increased complexity a non-linear structure of the synthesis system gives for the frame design method presented in Chapter 2.

3.6.1 Separable block-oriented frame in 2D

For the separable frame and the 2D signal, all the frame images are built as outer products of two column vectors, $F_{:,:,k} = \mathbf{u}_k \mathbf{v}_k^T$ or $F_{i,j,k} = u_k(i)v_k(j)$. It is also possible to use all the different combinations of column vectors to build the frame images, for example 8 different $\mathbf{u}$ vectors and 8 different $\mathbf{v}$ vectors gives 64 different combinations and 64 frame images. Each value in the frame is the product of two variables, and thus non-linear in the free variables. The problem of finding the optimal frame given the weights and the original signal can not be reduced to a linear problem. However, the problem may be formulated as two linear problems which may be solved iteratively in a way similar to the method described in Section 2.1. Step 2 in this method will be split into step 2.1 and step 2.2. Since the optimal solution is found in each of these smaller steps, the algorithm for step 2 will converge to a *local* minimum. The convergence properties of the total design method will be even more uncertain than for the linear cases described earlier, (where the *global* optimal solution is obtained in step 2).

For the block-oriented separable frame, each frame image is the outer product of two column vectors. The column vectors used in the vertical dimension

are collected into a frame, $\mathbf{F}_v$ (size: $N_1 \times K_1$). The column vectors used in the horizontal dimension are collected into another frame, $\mathbf{F}_h$ (size: $N_2 \times K_2$). There will be $K = K_1 K_2$ possible frame images, which may be indexed by $k \in \{1, 2, \ldots, K\}$ or by $k_1 \in \{1, 2, \ldots, K_1\}$ and $k_2 \in \{1, 2, \ldots, K_2\}$ where $k = k_1 + (k_2 - 1)K_1$. A frame image is

$$F_{:,:,k_1,k_2} = (\mathbf{F}_v)_{:,k_1}(\mathbf{F}_h)^T_{:,k_2} \tag{3.25}$$

$$F_{n_1,n_2,k_1,k_2} = (\mathbf{F}_v)_{n_1,k_1}(\mathbf{F}_h)_{n_2,k_2} \tag{3.26}$$

Note that the free variables are in $\mathbf{F}_v$ and $\mathbf{F}_h$ and not in F, each entry in F is formed as a product of one entry in $\mathbf{F}_v$, which is a free variable, and one entry in $\mathbf{F}_h$, which is another free variable. The dimensions for the different matrices are

$$\left.\begin{array}{lll} X & \text{4D of size} & N_1 \times N_2 \times L_1 \times L_2 \\ W & \text{4D of size} & K_1 \times K_2 \times L_1 \times L_2 \\ F & \text{4D of size} & N_1 \times N_2 \times K_1 \times K_2 \\ \mathbf{F}_v & \text{2D of size} & N_1 \times K_1 \\ \mathbf{F}_h & \text{2D of size} & N_2 \times K_2 \end{array}\right\} \tag{3.27}$$

The synthesis equation, corresponding to Equation 3.3, is

$$\tilde{X}_{:,:,l_1,l_2} = \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} W_{k_1,k_2,l_1,l_2} F_{:,:,k_1,k_2} \tag{3.28}$$

$$\tilde{X}_{:,:,l_1,l_2} = (\mathbf{F}_v) W_{:,:,l_1,l_2} (\mathbf{F}_h)^T \tag{3.29}$$

The last of these equations illustrates the reduced computationally complexity that can be achieved using a separable frame as the synthesis system.

To find the optimal values for the indices of $\mathbf{F}_v$ and $\mathbf{F}_h$ the problem is split into an iterative method of two steps. In step one $\mathbf{F}_h$ is considered to be constant and the best solution for $\mathbf{F}_v$ is found, for step two the roles of $\mathbf{F}_h$ and $\mathbf{F}_v$ are changed.

W_v is defined as 4D matrix (size: $K_1 \times N_2 \times L_1 \times L_2$)

$$(W_v)_{:,:,l_1,l_2} = W_{:,:,l_1,l_2}(\mathbf{F}_h)^T \tag{3.30}$$

Putting this into Equation 3.29 gives the synthesis equation for a column of the block

$$\tilde{X}_{:,n_2,l_1,l_2} = (\mathbf{F}_v)(W_v)_{:,n_2,l_1,l_2} \tag{3.31}$$

The indices n_2, l_1, l_2 are collected into one index i, where

$$i = n_2 + (l_1 - 1)N_2 + (l_2 - 1)L_1N_2,\ i \in \{1, 2, \ldots, I\},\ I = N_2L_1L_2. \tag{3.32}$$

This reshapes W_v into a 2D matrix (size: $K_1 \times I$), and X into a 2D matrix (size: $N_1 \times I$). Since the 4D matrices are reshaped into 2D matrices, they are denoted by bold letters in Equation 3.33. The problem is now just like for the block-oriented 1D case. The solution is given in Equation 2.9 and is

$$\mathbf{F}_v = \mathbf{X}\mathbf{W}_v^T(\mathbf{W}_v\mathbf{W}_v^T)^{-1} \tag{3.33}$$

Step two is done the same way as step one. W_h is defined as a 4D matrix (size: $N_1 \times K_2 \times L_1 \times L_2$).

$$(W_h)_{:,:,l_1,l_2} = (\mathbf{F}_v)W_{:,:,l_1,l_2} \tag{3.34}$$

Putting this into Equation 3.29 gives a matrix equation for each block,

$$\tilde{X}_{:,:,l_1,l_2} = (W_h)_{:,:,l_1,l_2}(\mathbf{F}_h)^T. \tag{3.35}$$

Looking at one row of this matrix (left side of Equation 3.35), and transposing to get a column vector gives

$$\tilde{X}_{n_1,:,l_1,l_2} = (\mathbf{F}_h)(W_h)_{n_1,:,l_1,l_2} \tag{3.36}$$

Note that the row is represented as a column vector both for the $\tilde{X}$ and W_h MD matrices, this is how a line through a MD matrix was defined in Section 3.1. The indices n_1, l_1, l_2 are collected into one index j, where

$$j = n_1 + (l_1 - 1)N_1 + (l_2 - 1)L_1N_1,\ j \in \{1, 2, \ldots, J\},\ J = N_1L_1L_2. \quad (3.37)$$

Collection these J vectors into a 2D matrix reshapes W_v into a 2D matrix (size: $K_2 \times J$), and X is reshaped into a 2D matrix (size: $N_2 \times J$). Here too, the solution is given by Equation 2.9 and is

$$\mathbf{F}_h = \mathbf{X}\mathbf{W}_h^T(\mathbf{W}_h\mathbf{W}_h^T)^{-1}. \quad (3.38)$$

In the experiments done, Section 5.2, the iterations within this step converged fast, usually 3-6 iterations was enough. The convergence rate seemed to be geometric with the factor a in the range from 0.01 to 0.1, i.e. $\|\mathbf{F}_v^{i+1} - \mathbf{F}_v^i\| \approx a\|\mathbf{F}_v^i - \mathbf{F}_v^{i-1}\|$ and the same for $\mathbf{F}_h$.

At the end of this section let us just add that the case for the overlapping separable frame can be solved by combining the results in this section with the results in Section 3.3, the challenge is the increased number of indices to keep track of.

Chapter 4

Vector Selection in Frame Design

The iterative method for frame design, Section 2.1, has two main steps. In Chapter 2 and 3 the optimal solution for step 2 was found for a large number of different frame structures. In this chapter we focus on the first step: How to find the weights when the frame is fixed. This is the most computationally demanding step in the frame design method. When the weights are found some constraints may be given which the weights must meet. In this chapter it will be a sparseness constraint, but also other constraints could be used (combined with the sparseness constraint), for example in compression the weights should be quantized and represented by a limited number of bits. The vector selection step is also important since almost all the computing time is spent on this part, at least when the sparseness criterion is used. Different algorithms for vector selection was presented in the introduction, Section 1.3. We will now look more in detail how these can be used for frame design when a sparseness criterion, as defined in Equation 1.11, is imposed. The size of the problem makes it necessary to use sub-optimal vector selection algorithms, which may give poor convergence properties in the iterative design method. In the iterative design method it is possible to take advantage of the fact that the weights from the previous iteration are available when selecting weights for the current iteration. This may be helpful especially if the current frame has only minor changes from the previous iteration.

Let us look closer at the problem size. The frame matrix $\mathcal{F}$ as introduced in Equation 1.5, an example for the overlapping frame in Equation 2.19, is of size $NL \times KL$. L is the number of training vectors, it is large enough to make the training data representative for the signal class, typically $1000 < L < 100000$.

This makes the frame matrix $\mathcal{F}$ very large, and impractical to use directly in vector selection. The common way to avoid this problem is to use a block-oriented frame and to select a fixed number of frame vectors, denoted by a lowercase s, to represent each signal block. Referring to Equation 1.4 each $\mathbf{x}_l$ is then represented as a linear combination of s of the column vectors of the frame $\mathbf{F}$, giving the sparseness factor $S = s/N$. Note that the sparseness factor, as defined in Equation 1.11, allows for more flexibility since it is a global definition, the number of frame vectors to use to represent $\mathbf{x}_l$ can be s_l where $0 \leq s_l \leq N$ and $\sum_{l=1}^{L} s_l = SNL$. Under many circumstances it will be favorable to have a global vector selection algorithm, or at least an algorithm that allows for some flexibility when selecting the positions for the non-zero weights.

The general problem is how to find how many non-zero weights that should be used for each of the signal blocks, a discussion of this is given in Section 4.2. A natural approach is to select weights for a few of the signal blocks, let us say M blocks, in "one operation", see Subsection 4.2.4. This should allow for full flexibility in selecting the non-zero weights within these M blocks. This approach may also be viewed as splitting the large problem of size L into smaller problems each of size M.

For the overlapping frame it is more complicated to divide the large problem into several smaller ones, since the smaller problems are overlapping and will influence each other. One way of handling this inter-block influence is described in Section 4.3. Another approach for the overlapping frame is to do the design in some coefficient domain as described in Section 4.4.

Even when the large problem is divided into smaller problems, the problem size is still much too large for the Full Search algorithm. The medium sized example ($N = 64$, $K = 128$, and $s = 8$) in the introduction was estimated to 25 years, and the problems to be solved during frame design are typically even larger than this, especially for the overlapping frames. An algorithm that do partial search was developed during the work of this thesis. This algorithm can be scaled to examine only one combination, then it works exactly as ORMP, see Section 1.3, up to examine all the possible combinations of non-zero weights, then it is as the full search algorithm. The partial search algorithm is described in Subsection 4.1.1. One disadvantage is that quite often many combinations must be searched to get better results than ORMP and then the computation cost is too high. The experience gained through design of many frames is that the sub-optimal vector selection algorithms should be used, and generally ORMP is the preferred one.

The iterative design method will generally not converge when sub-optimal vector selection algorithms are used. This means that a small change in the

frame in step 2 may give a large change in the weights in step 1 in the next iteration and the norm of the representation error may even be increased. To improve the convergence properties of the iterative frame design method a hybrid vector selection algorithm is proposed in Subsection 4.1.3. The main idea is that the weights and the error from the previous iteration are stored, and when new weights are found the new error is compared to the previous one and only if there is an improvement the new weights are used. Another advantage by using the previous weights is also possible. When the design method comes close to convergence, the change for the frame is small from one iteration to the next, we then expect that most of the weights are also used again. Doing vector selection by keeping some of the weights from the previous iteration will save computation time and give a faster algorithm, this is described in Section 4.1.2. However, we should note that the optimal solution may be a completely new set of frame vectors even for a very small change in the frame.

In the end of this introduction we should mention that in some applications, like compression, it is a problem that vector selection may give $\|\mathbf{w}\| >> \|\mathbf{x}\|$. The vector selection algorithm used has some influence on the size of $\|\mathbf{w}\|$, BMP is in this respect better than OMP and ORMP [18]. It seems difficult to get a good control of this effect, but the size of $\|\mathbf{w}\|$ is also determined by the frame properites discussed in Subsection 7.3.4.

4.1 Three new algorithms for vector selection

Here we suggest three new algorithms for vector selection, that are all based on the algorithms presented in the introduction, Section 1.3, and they may be regarded as variants of these.

4.1.1 Vector selection by partial search

The *Partial Search* algorithm examines some of the possible combinations for selecting s vectors out of the K frame vectors available, it can be tuned to examine from one to all combinations. If only one combination is examined this algorithm is identical to ORMP, if all combinations are examined then full search is done.

The algorithm can be described by a recursive function as shown in Figure 4.1. The input to this function is the uniform frame, $\mathbf{F}$ of size $N \times K$, the signal vector, $\mathbf{x}$, and the number of vectors to select, s. We also include a fourth

Vector selection by partial search algorithm:

```
1   [r, i] = PartialSearch(F, x, s, c)
2       N and K found from size of F
3       u := F^T · x
4       k is found by sorting u, |u(k(1))| ≥ |u(k(2))| ≥ ··· ≥ |u(k(K))|
5       if s = 1
6           i := k(1)
7           r := ‖x − u(k(1)) · f_{k(1)}‖
8       else
9           r := ∞, s' := s − 1, and c' := c \ {c(1)}
10          for j := 1 to c(1)
11              x' := x − u(k(j)) · f_{k(j)}
12              F' := F \ {f_{k(j)}}
13              for m := 1 to (K − 1)
14                  f'_m := f'_m − (f^T_{k(j)} · f'_m) f_{k(j)}
15                  f'_m := f'_m / ‖f'_m‖
16              end
17              [r', i'] := PartialSearch(F', x', s', c')
18              if r' < r
19                  r := r'
20                  i := {i'(1), i'(2), ..., k(j), ..., i'(s') + 1}
21              end
22          end
23      end
24      return
```

Figure 4.1: The Partial Search algorithm for a uniform frame. Bold uppercase letters are used for matrices, bold lowercase letters are used for column vectors and index sets. Note that $\mathbf{f}'_m$ is a column vector of the $\mathbf{F}'$ matrix, like $\mathbf{f}_{k(j)}$ is a column vector of the $\mathbf{F}$ matrix. More comments to the algorithm are to be found in the text.

argument that gives how many combinations to be examined. This is given as a vector $\mathbf{c}$ of size $s \times 1$, where $c(1)$ is the number of vectors to check as candidates for the first vector to select, $c(2)$ is the number of vectors to check as candidates for the second vector to select, and so on. The total number of combinations to try is then $\prod_{i=1}^{s} c(i)$. The values returned by the recursive function are the norm of the residual, r, and the indices of the frame vectors that were selected, $\mathbf{i}$. The weights are not returned by the function, since they can easily be found when we know which frame vectors to use.

Inside the function the first thing that is done is that the inner products between the signal vector and the column vectors of the frame are calculated, line 3 in Figure 4.1. The inner products are sorted according to their absolute value, the indices for the sorted order is stored in $\mathbf{k}$, line 4. Then two cases are handled.1) If only one vector should be selected, $s = 1$, the situation is easy: the selected frame vector is the one for which the inner product with the signal vector is largest, line 6. Note that $\mathbf{i}$ is the set of selected indices, even if it only has one element in line 6 it is denoted by a bold letter. In line 7 the norm of the error is calculated, the error is the signal vector subtracted the projection of itself onto the selected frame vector, i.e. *orthogonalized* to the selected frame vector. 2) If more than one vector should be selected, $s > 1$, there is more work to be done. In line 9 r is set to a large number just to make sure that the test in line 18 will be true the first time. Also s' and $\mathbf{c}'$ are set in line 9, the tagged variables are the ones that will be used as arguments when the function is called again in line 17. In lines 10 to 22 each of the $c(1)$ most promising frame vectors, i.e. the first ones from the sorted list, are tried. Using frame vector $\mathbf{f}_{k(j)}$ the remaining part of the signal, $\mathbf{x}'$, is found in line 11. A modified frame, $\mathbf{F}'$, is found by removing the current frame vector $\mathbf{f}_{k(j)}$, line 12, then orthogonalizing each of its frame vectors to the removed frame vector $\mathbf{f}_{k(j)}$, line 14, and finally adjusting the length of the new frame vector, line 15. We are now ready to search in the next level, this is done by a call to the same function, line 17. If the error returned is smaller than the smallest error so far, the returned error is stored as the smallest error so far, line 19. Also the indices $\mathbf{i}$ referring to the columns of $\mathbf{F}$ are found from the indices $\mathbf{i}'$ referring to the columns of $\mathbf{F}'$ and the index of the current frame vector $k(j)$, line 20. Note that the indices in $\mathbf{i}$ and $\mathbf{i}'$ are in ascending order, and that $k(j)$ is put into the correct position, and that the values of $\mathbf{i}'$ after $k(j)$ must be incremented when they are used in $\mathbf{i}$. This concludes the function.

Within the function the update from $\mathbf{F}$ to $\mathbf{F}'$ will be like one step of the QR factorization in linear algebra, this means that the function can be implemented quite effectively. But we should not forget that the amount of work to be done is approximately proportional to the number of combinations to

try, and this number may be very large. If full search of a reasonably sized problem is wanted another algorithm should be used, as this function may examine different combinations of the same frame vectors, i.e. permutation on the order the frame vectors are selected in.

The extension of the search is set by the values in **c**. If all entries of **c** are one, the function is an implementation of the ORMP algorithm, though not a fast one. Our experience with this algorithm is quite good. The returned solution is the same as the one found by ORMP, or it is a better one. In the experiments done we used from 50 to 500 different combinations and in approximately 75% of the function calls this function returned better results than ORMP.

4.1.2 Vector selection using previous weights

Matching Pursuit that use weights of the previous iteration of the design method is a simple modification to the BMP algorithm. The difference is that the previous weights are used as input argument instead of number of vectors to select, the number of vectors to select is implicit given as the number of non-zero previous weights. By keeping, for example the three largest weights, a part of the work in vector selection is already done. Then BMP, or another MP algorithm, is used to select the rest of the weights. This means that fewer iterations than full BMP is needed. But it also means that the chance of finding better weights is reduced since completely new weights will not be found by this algorithm.

4.1.3 Improving convergence for the design method

The convergence properties of the iterative frame design method, Section 2.1, can be improved using a vector selection algorithm that always returns better weights, or more exactly never returns weights that gives a larger residual than the residual in the previous iteration. Looking closer at the method in Section 2.1 we see that this will guarantee convergence. In step 2 the optimal frame for the given weights is found, for example by Equation 2.9. If the frame is changed the error is always reduced. The normalization of the frame vectors in step 3 do not change the error. In the next iteration, vector selection in step 1 use the weights from step 3, and only if better weights are found the old weights will be replaced. This makes sure that the error is reduced (or unchanged) also for step 1. The fact that the norm of the error never increases ensures that the design method converges to a local minimum of the object function.

Obviously, the weights from the previous step, in addition to the frame and the signal vector, are needed as input argument in this vector selection algorithm. The algorithm finds the new weights using one of the previously mentioned algorithms, and the new residual is compared to the old residual. The weights corresponding to the smallest residual are returned from the function. Often this algorithm will just return the old weights, but when some new weights are selected the result will be an improvement, the norm of the error will be reduced.

The new weights could be found by choosing a vector selection algorithm in a "random" way, where the computationally easy algorithms, like the algorithm using previous weights in Section 4.1.2 or the ORMP algorithm, are more likely to be used than computationally more expensive algorithms, like the partial search, BP or the OMP. The benefit of choosing the vector selection algorithm randomly is that we do not know which one of the many vector selection algorithms that works best for the particular signal vector and frame. Trying another algorithm than the one used in previous iteration of the design method may be a smart thing to do, especially if the frame is almost unchanged, as it will be after some iterations, when convergence is getting closer. This way of choosing vector selection algorithm ensures that many different algorithms are tried without increasing the computational cost. This hybrid algorithm is the one that was used in the experiments in Chapter 5.

It would be interesting to know if the suggested improvements to vector selection for frame design have any practical effect. We test this by designing three frames where the only difference is the vector selection algorithm. The frames were designed as described in Section 5.1. The frame structure is the overlapping frame as in point (d) in the same section, using target sparseness factor $\frac{1}{16}$. In Figure 4.2 the SNR, Equation 1.9, after step one in each iteration is plotted as a function of iteration number. For each of the three vector selection algorithms 500 iterations are shown. We note that neither BMP nor ORMP (FOMP) converge properly, while the hybrid algorithm seems to converge quite well. Also, after approximately 100 iterations both the BMP and the ORMP algorithms have reached close to their maxima, it seems not necessary to do more iterations, while the hybrid algorithm improves all the time. To further investigate the convergence properties of the hybrid algorithm 5000 iterations were done, and the SNR was further increased 0.33 decibel or almost 1 decibel better than by ORMP. Looking closer at the SNR curve, Figure 4.3, it has improvements in small steps. Since the hybrid algorithm occasionally use the partial search algorithm (and the BP algorithm) better weights may be found at each iteration even if the frame is unchanged. This is what I think makes these small steps in the SNR-curve, and the change in the frame

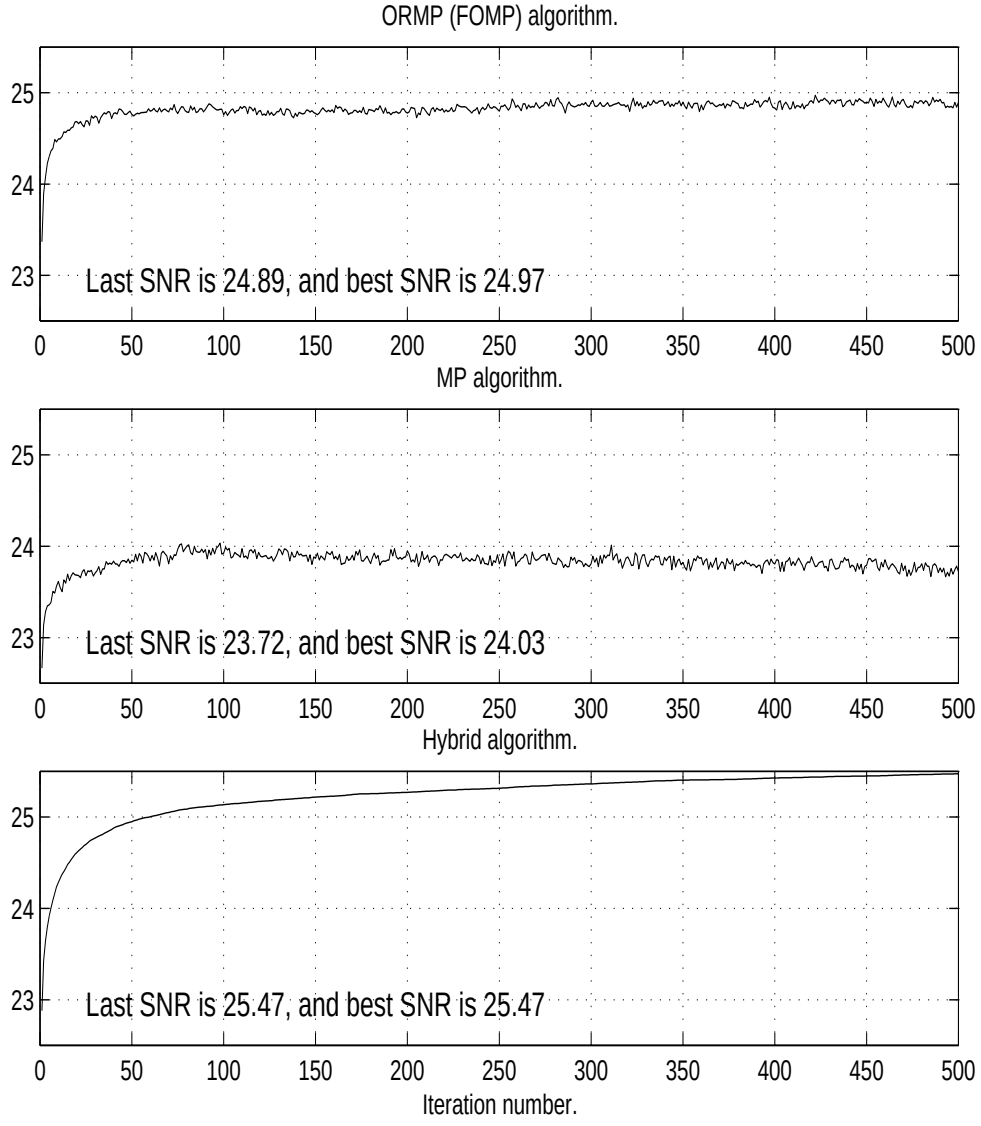


Figure 4.2: SNR is plotted as a function of training iterations. The training is for an overlapping frame, frame (d) in Section 5.1, the target sparseness factor is $\frac{1}{16}$.

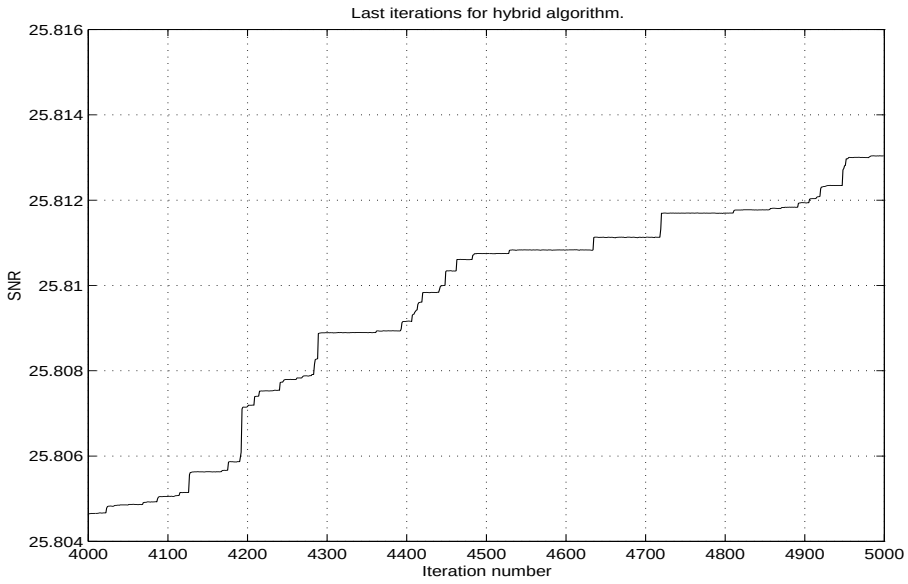


Figure 4.3: SNR for iteration 4000 to 5000 for the hybrid vector selection algorithm The same frame as in Figure 4.2, frame (d) in Section 5.1.

is very small for the last thousands of iterations. Thus, the frame is probably very close to the final one after some hundreds iterations also for the hybrid algorithm.

The time until "convergence" is reached is also relevant, the BMP and ORMP use approximately the same time for each iteration, for the current example 9 seconds, while the hybrid algorithm uses approximately 4 times longer. But of course the time for the hybrid algorithm is highly dependent on how it is set up, particulary how often partial search is done and how many combinations to examine, it may be scaled to run as fast as ORMP (or BMP) by only using this algorithm. The conclusion is that the hybrid algorithm should be used for frame design, especially since the computational time is not very important for design.

4.2 Distribution of non-zero weights

First, let us point out that by distribution of non-zero weights we mean how many non-zero weights used on each of the signal blocks, not the distribution

of the weight values. A signal $\mathbf{x}$ (size: $NL \times 1$) can be approximated by a linear combination of some frame vectors from the frame matrix $\mathcal{F}$, introduced in Equation 1.5 for the block-oriented frame and Equation 2.12 for the overlapping frame. The frame $\mathcal{F}$ is of size $NL \times KL$, so the size for the vector selection problem is generally very large. The sparseness factor, S, as defined in Equation 1.11 is used in this context. To solve this large vector selection problem it must be divided into smaller problems, this is quite easy for the block-oriented frame and also possible for the overlapping frame.

The division into smaller problems introduce another problem: how should the total number of vectors to select, $S \cdot (\text{number of samples in } \mathbf{x}) = SNL$, be distributed among the subproblems, i.e. among the blocks? If we let s_l be the number of non-zero weights allowed, or used, in block l of the weights, the distribution of non-zero weights is defined by these numbers $\{s_l\}_{l=1}^{L}$, where $\sum_{l=1}^{L} s_l = SNL$. Several different methods may be used to find this distribution.

4.2.1 Replace sparseness constraint by error constraint

The simplest alternative is to ignore the strict and global sparseness constraint and instead use a constraint on the error on each block , $\|\mathbf{r}_l\| = \|\mathbf{x}_l - \tilde{\mathbf{x}}_l\| < \epsilon\|\mathbf{x}_l\|$. The MP vector selection algorithms (BMP, OMP and ORMP) may all use the magnitude of the error as a stop criterion. This approach gives good control of the error and indirect control of the sparseness factor. This is a good way to distribute the non-zero weights along the signal, and was used in [22].

4.2.2 Distribute the weights evenly

Another easy way to distribute the non-zero weights is to distribute them evenly among the signal blocks. This gives $s_l = s = SN$ for all blocks, where it is assumed that SN is integer. If SN is not integer we can round it either up or down for each s_l such that the total sum is SNL. For stationary signals, for example an AR(1) signal[1], this may be a good approach, but for more irregular signals having some regions with much activity and some relatively flat regions, for example images, this is not a good idea. More frame vectors should be used on the parts of the image rich with details than on the parts of the image that are flat.

[1] An AR(1) signal is the output of an autoregressive (AR) process of order 1.

4.2.3 Global matching pursuit

The main disadvantage of the vector selection algorithms is that they are impractical for large problems. But when the large frame has a structure, as in Equation 1.4 for the block-oriented frame and in Equation 2.12 and Equation 2.19 for the overlapping frame, the Matching Pursuit algorithms can be adapted to do Global Matching Pursuit (GMP) without a large, i.e. much smaller than the problem size would indicate, increase in computing time. This is most easily done for the block-oriented frame based on the Matching Pursuit algorithm, and the algorithm for this case is described below. It was also described in [63].

This algorithm will return the same weights as if BMP is done using the synthesis equation with the large frame, $\tilde{\mathbf{x}} = \mathcal{F}\mathbf{w}$, but here the starting point is the L synthesis equations using the smaller frame $\mathbf{F}$, compactly written as $\tilde{\mathbf{X}} = \mathbf{FW}$ Equation 2.2. The steps in GMP are[2]

1. Set the current weight matrix to zero, $\mathbf{W} = \mathbf{0}$, and the current residual matrix to the signal matrix, $\mathbf{R} = \mathbf{X}$. Then we find all the inner products, $\mathbf{U} = \mathbf{F}^T\mathbf{R}$, (size: $K \times L$). $U(k,l)$ is the inner product of frame vector k and column l in $\mathbf{R}$.

2. Find the largest (in absolute value) $\lfloor SL \rfloor$ inner products of $\mathbf{U}$ but not more than one from each column, giving the best vector for those blocks that are most in need of one of the $\lfloor SL \rfloor$ vectors to be distributed. This gives a set of indices, denoted $\{(k_i, l_i)\}_{i=1}^{\lfloor SL \rfloor}$.

3. For $i = 1$ to $\lfloor SL \rfloor$, i.e. for each of the $\lfloor SL \rfloor$ different pairs of indices, use the frame vector given by k_i to approximate column l_i of $\mathbf{X}$:

   (a) Update the weight,
   $W(k_i, l_i) = W(k_i, l_i) + U(k_i, l_i)$.

   (b) Update column l_i of $\mathbf{R}$,
   $R(:, l_i) = R(:, l_i) - U(k_i, l_i) \cdot F(:, k_i)$

   (c) Stop if we have selected enough weights.

   (d) Update the inner products for vector l_i of $\mathbf{R}$,
   $U(:, l_i) = \mathbf{F}^T \cdot R(:, l_i)$

4. If more frame vectors may be selected goto 2.

[2]Here we use notation similar to the notation used in Matlab, $W(k,l)$ is entry in row k and column l of matrix $\mathbf{W}$ and $W(:,l)$ is column l of matrix $\mathbf{W}$.

The modification compared with BMP, Figure 1.2, when BMP is selecting $s = SN$, SN is integer, frame vectors for each training vector, can be seen in step 2. BMP would select the L largest inner products, one for each column of $\mathbf{U}$. By reducing the number of selected inner products we make sure that the outer loop, step 2 to 4, is done at least N times, making it possible to select many frame vectors for those vectors (columns) of $\mathbf{X}$ that require so.

4.2.4 Use an extended frame

An extended frame, denoted $\mathbf{F}^M$ (size: $MN \times MK$), is built by repeating the frame $\mathbf{F}$ M times

$$\mathbf{F}^M = \begin{bmatrix} \boxed{\mathbf{F}} & & & \\ & \boxed{\mathbf{F}} & & \\ & & \ddots & \\ & & & \boxed{\mathbf{F}} \end{bmatrix} \tag{4.1}$$

To use an extended frame to find the distribution of the non-zero weights is a good alternative to the GMP algorithm. Using the larger extended frame we have a larger vector selection problem, but the non-zero weights may be distributed freely among the M repetitions of $\mathbf{F}$ in $\mathbf{F}^M$. One minor drawback is that an initial distribution of the weights is needed. The computing time of the vector selection algorithms increases considerably as the size of the problem, i.e. frame, increases, so the extended frame can not be too large. To understand the rather trivial extended frame for the block-oriented case will be helpful when the more complicated overlapping frame is presented in the next section.

An extended signal block is built by concatenating M of the original signal blocks, $\{\mathbf{x}_{l_i}\}_{i=1}^{M}$. For the block-oriented frame these blocks do not need to be consecutive blocks, they are indexed by $\{l_i\}_{i=1}^{M}$. The corresponding weights and the number of vectors to use for each block are indexed the same way. The number of non-zero weights within each of the M blocks may differ. The vector selection algorithm finds an approximation to the extended signal block using $s = \sum_{i=1}^{M} s_{l_i}$ non-zero weights. During vector selection the number of non-zero weights for a single block may be changed, but their sum should still be the same. If this is done many times, each time using different signal blocks to build the extended signal block, the final result will be an updated, and

presumably better, distribution of the weights. This method is well suited for the iterative frame design method. The synthesis equation for M blocks is

$$\tilde{\mathbf{x}}^M = \begin{bmatrix} \tilde{\mathbf{x}}_{l_1} \\ \tilde{\mathbf{x}}_{l_2} \\ \vdots \\ \tilde{\mathbf{x}}_{l_M} \end{bmatrix} = \begin{bmatrix} \boxed{\mathbf{F}} & & & \\ & \boxed{\mathbf{F}} & & \\ & & \ddots & \\ & & & \boxed{\mathbf{F}} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{l_1} \\ \mathbf{w}_{l_2} \\ \vdots \\ \mathbf{w}_{l_M} \end{bmatrix} = \mathbf{F}^M \mathbf{w}^M. \qquad (4.2)$$

One may ask how large the number M should be in the extended frame. A large value will result in a good distribution of the non-zero weights with fewer iterations than a smaller value, but a small value will make vector selection computationally less demanding and the complexity highly depends on the used vector selection algorithm. The rule of thumb I found from the many experiments done during the work on this thesis is that the number of vectors to select "at once" from this extended frame should not exceed 50. For large overlapping frames it was sometimes necessary to have this number as high as 100, but then vector selection was rather demanding.

4.3 Overlapping vector selection

Vector selection using an overlapping frame is more difficult than for the block-oriented frame. Remember that all the vector selection algorithms in Section 1.3 are block-oriented. They must be updated or adapted and wrapped into another algorithm before they can be used in the overlapping frame case. One way to do this is to use an extended frame similar to the extended frame introduced in Subsection 4.2.4. To see how the use of an extended frame can "reduce" the large overlapping frame vector selection problem into smaller block-oriented problems is one of the more difficult things in this thesis; we think the best way to explain this is to illustrate it by a "simple" example. We chose a frame where the overlap factor is $P = 3$ and the extended frame, $\mathbf{F}^M$, is built by repeating the frame, $\mathbf{F}$ as in Equation 2.11, $M = 5$ times.

The central part of the synthesis equation, $\tilde{\mathbf{x}} = \mathcal{F}\mathbf{w}$, is shown in Figure 4.4. Let us start the explanation at the right part of the figure. A part of the weight vector, $\mathbf{w}^M$, is the concatenation of M blocks from the weight vector, $\{\mathbf{w}_{l+i}\}_{i=1}^{M}$. At the left side of the figure the reconstructed signal vector, $\tilde{\mathbf{x}}$, is split into three terms:
1) $\tilde{\mathbf{x}}^b$ is built using the weights before $\mathbf{w}^M$ $(\mathbf{w}_i, i \leq l\,)$,

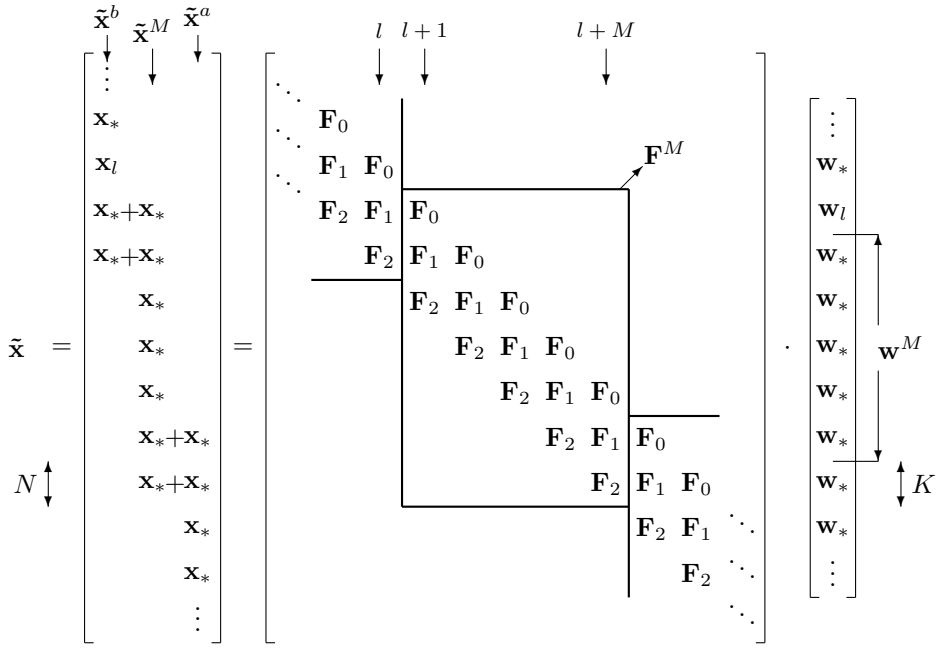


Figure 4.4: A closer look at the synthesis equation $\tilde{\mathbf{x}} = \mathcal{F}\mathbf{w}$. The overlap factor is $P = 3$, and the extended frame, $\mathbf{F}^M$, is made by repeating $\mathbf{F}$ $M = 5$ times. $\mathbf{w}^M$ is the concatenation of M weight blocks $\{\mathbf{w}_{l+i}\}_{i=1}^{M}$. The star index used in the blocks of $\mathbf{w}$ indicate the appropriate index, i.e. $\mathbf{w}_* = \mathbf{w}_{l+i}$ where i is an integer. For the reconstructed signal block the star index is a block or one part of a block, $\tilde{\mathbf{x}}_{l+i}$. The indices written above the $\mathcal{F}$ matrix are used to show the proper alignment of weights to synthesis vectors. The first l, with an arrow pointing down to the columns of $\mathcal{F}$ just to the left of $\mathbf{F}^M$, indicate that the weights within the block with this index, $\mathbf{w}_l$, are the weights used for the K synthesis vectors in this part of $\mathcal{F}$.

2) $\tilde{\mathbf{x}}^M$ (size: $N(M+P-1)\times 1$) is built using the weights in $\mathbf{w}^M$, and
3) $\tilde{\mathbf{x}}^a$ is built using the weights after $\mathbf{w}^M$ $(\mathbf{w}_i, i > (l+M))$.
They overlap each other by $(P-1)$ blocks as illustrated in the figure. The frame, $\mathcal{F}$, is expressed by its constituent matrices, $\{\mathbf{F}_p\}_{p=0}^{P-1}$ as in Equation 2.12 and Equation 2.19. The extended matrix, $\mathbf{F}^M$ (size: $N(M+P-1)\times KM$), is the frame matrix, $\mathbf{F}$, repeated $M=5$ times and it is illustrated by a surrounding box in Figure 4.4. The synthesis equation for the central block is

$$\tilde{\mathbf{x}}^M = \mathbf{F}^M \mathbf{w}^M, \tag{4.3}$$

where the reconstructed signal is an approximation to $\mathbf{x}^M$, i.e.

$$\tilde{\mathbf{x}}^M \approx \mathbf{x}^M = (\mathbf{x} - \tilde{\mathbf{x}}^b - \tilde{\mathbf{x}}^a). \tag{4.4}$$

Vector selection for this central block should find an approximation to $\mathbf{x}^M$ as a linear combination of some of the column vectors of $\mathbf{F}^M$. This is a "simple" block-oriented vector selection problem, with one small exception, the target signal $\mathbf{x}^M$ is not exactly known as it depends on the weights before and after the central block of weights, $\mathbf{w}^M$. Let us assume that we have given some weights $\{\mathbf{w}_l\}_{l=1}^L$ and a distribution of the non-zero weights $\{s_l\}_{l=1}^L$ Then $\tilde{\mathbf{x}}^b$ and $\tilde{\mathbf{x}}^a$ are known and the target signal $\mathbf{x}^M$ is found. Block-oriented vector selection, based on Equation 4.3, can now be used to find the weights, $\mathbf{w}^M$. The initial weights could be only zeros and the initial distribution could be evenly distributed, Subsection 4.2.2.

The procedure for overlapping vector selection can be described by starting at a random offset for the central part, i.e. $(l+1)$, as seen in Figure 4.4. The weights $\mathbf{w}^M$ or $\{\mathbf{w}_{l+i}\}_{i=1}^M$ are updated using a block-oriented vector selection algorithm where the frame $\mathbf{F}^M$ is used and the number of vectors to select is given by $s = \sum_{i=1}^M s_{l+i}$. The target signal is $\mathbf{x}^M$ as in Equation 4.4. Only the relevant $(P-1)$ blocks of $\tilde{\mathbf{x}}^b$ and $\tilde{\mathbf{x}}^a$ are needed, and to calculate these the corresponding blocks of weights before and after $\mathbf{w}^M$ are needed. The vector selection algorithm updates the weights $\mathbf{w}^M$ and implicitly also the distribution $\{s_{l+i}\}_{i=1}^M$. Then a new offset is selected, for example $l = (l+M) \bmod L$, and the procedure is repeated until all the weight blocks are determined, updated at least once.

Note that a new offset does not need to be the previous offset incremented by M, the important thing is that all the weights get a chance to be updated. In

fact, the experiments done have shown that it is often better to use a smaller increment, this often decrease the "blocking-effects" and allow for the non-zero weights "to drift" towards the more detailed regions of the signal, i.e. update the distribution of weights. An increment larger than M could also be used with much the same results. The new offset could also be selected "at random", the main issue is that all blocks of the weight vector are given a chance to get updated.

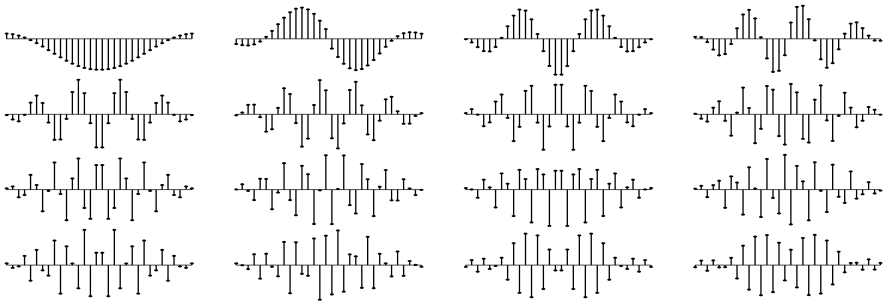
This algorithm works quite well, but it is much more computationally demanding than the block-oriented algorithms. Mainly because the extended frame used must be larger, only $(M-(P-1))$ blocks are unaffected by the weights which are outside of $\mathbf{w}^M$. If the initial weights are all zero each weight block should be selected at least two times to assure that the last time the weights which are outside of $\mathbf{w}^M$ have been set.

To summarize: The algorithm described in this section is flexible and general. It may be used for both block-oriented frames and overlapping frames, and the extended block size can be set to an appropriate value, we should probably have M larger or equal to P and small enough to make the vector selection problem of reasonable size. Any of the many possible vector selection algorithms can be used as the core of this algorithm. Some initial weights should be given for the overlapping frame cases. These are needed to calculate the target signal, $\mathbf{x}^M$, or more specifically the reconstructed signal before, $\tilde{\mathbf{x}}^b$, and after, $\tilde{\mathbf{x}}^a$, an extended block. It is correct to say that for the overlapping frame the described algorithm is an algorithm to improve weights, while for the block-oriented frame this algorithm do not need any initial weights, it selects/finds the weights. For both cases the wanted distribution of the weights is needed initially, and possibly updated during the procedure.

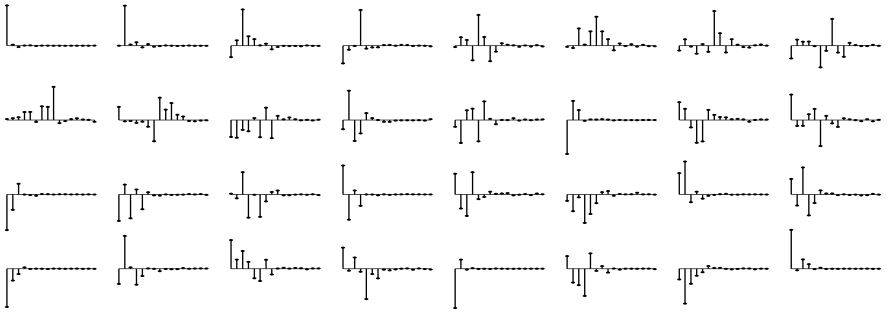
4.4 Frame in coefficient domain

The algorithm in the previous section works better for block-oriented frames than for overlapping frames, both because the extended block size can be smaller and because the problem with interference of weights before and after current extended block of weights is eliminated. In [65] we proposed a method that makes it possible to design overlapping frames where block-oriented vector selection can be done. This makes vector selection for the overlapping frame as easy as for the block-oriented frame.

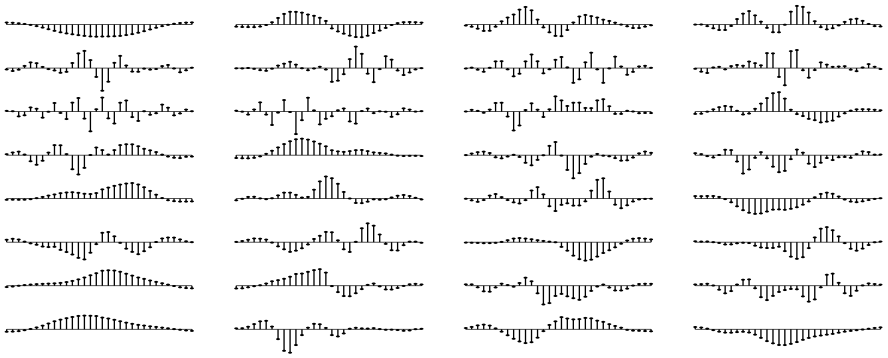
In arriving at the proposed method we posed the following question: Given a desired overlapping frame structure as shown in Equation 1.10, is it possible to decompose it into the product of one block-oriented structure as in



The LOT synthesis vectors, $\mathbf{G}$ of size 32×16



The frame vectors, $\mathbf{H}$ of size 16×32

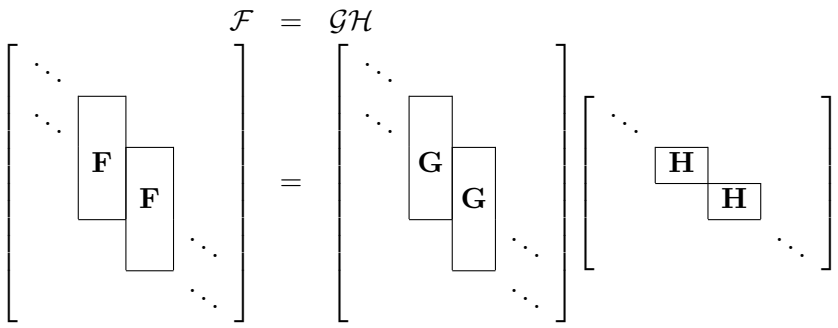


The resulting synthesis vectors, $\mathbf{F} = \mathbf{GH}$ of size 32×32

Figure 4.5: The synthesis matrix $\mathbf{F}$ is made up by the matrix product of $\mathbf{G}$ and $\mathbf{H}$. Each column of $\mathbf{F}$ is a linear combination of the columns of $\mathbf{G}$, the coefficients are given by the corresponding column of $\mathbf{H}$.

Equation 1.4 and another matrix? If so, could this other matrix be fixed, predefined prior to the design process, while the design effort is spent on the block structured part of the decomposition?

It is easily verified that setting $\mathcal{F} = \mathcal{G}\mathcal{H}$, that is



(4.5)

with $\mathcal{G}$ as in Equation 1.10 (size of $\mathbf{G}$: $PN \times N$), and $\mathcal{H}$ as the block-oriented frame in Equation 1.4 (size of $\mathbf{H}$: $N \times K$), gives the overlapping frame $\mathcal{F}$ with the desired structure, (size of $\mathbf{F}$: $PN \times K$). Note that the structure of the first matrix, $\mathcal{G}$, corresponds to the synthesis matrix of a critically sampled FIR synthesis filter bank. The constituent matrices of $\mathcal{F}$ are defined by

$$\mathbf{F} = \mathbf{G}\mathbf{H} = \begin{bmatrix} \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_P \end{bmatrix} \mathbf{H} = \begin{bmatrix} \mathbf{G}_1\mathbf{H} \\ \vdots \\ \mathbf{G}_P\mathbf{H} \end{bmatrix}. \tag{4.6}$$

The signal representation is now $\tilde{\mathbf{x}} = \mathcal{F}\,\mathbf{w} = \mathcal{G}\mathcal{H}\,\mathbf{w}$.

For a given class of signals, specified by a large vector $\mathbf{x}$ containing an appropriate training set of signal segments, the task of designing $\mathcal{F}$ can be divided into two parts: selecting a reasonable $\mathcal{G}$, which we then keep fixed, and finding a $\mathcal{H}$ (or equivalently its constituent matrices $\mathbf{H}$) using the design method for block-oriented frames. The object function for the second step in the iterative design method, will now be $J = J(\mathbf{H}) = \|\mathbf{x} - \mathcal{G}\mathcal{H}\mathbf{w}\|$.

Suppose that the columns of $\mathcal{G}$'s constituent matrices, $\mathbf{G}$, are chosen as the synthesis vectors (filter responses) of an *orthogonal* perfect reconstruction filter bank, then $\mathcal{G}^{-1} = \mathcal{G}^T$ and the norm is conserved, $\|\mathbf{x}\| = \|\mathcal{G}\mathbf{x}\| = \|\mathcal{G}^{-1}\mathbf{x}\|$. This implies that

$$J = \|\mathbf{x} - \mathcal{G}\mathcal{H}\mathbf{w}\| = \|\mathcal{G}^{-1}(\mathbf{x} - \mathcal{G}\mathcal{H}\mathbf{w})\| = \|\mathcal{G}^T\mathbf{x} - \mathcal{H}\mathbf{w}\|, \tag{4.7}$$

and the frame $\mathcal{H}$ can be designed in exactly the same manner as the design of any block-oriented frame. The only difference is that we use $(\mathcal{G}^T\mathbf{x})$ rather than $\mathbf{x}$ as the training signal. That is, we do the approximation in the coefficient domain rather than in the signal domain. One example: $\mathbf{G}$ is selected as the synthesis vectors of a 32 tap 16 channel Lapped Orthogonal Transform (LOT), [46]. This has overlap factor $P = 2$ and $N = 16$. $\mathbf{H}$ is a 16×32 block-oriented frame, it is overcomplete by a factor of $K/N = 2$. The resulting overlapping frame $\mathbf{F}$ has $P = 2$, $N = 16$ and $K = 32$. This example is illustrated in Figure 4.5.

This change of problem means that it is possible to use block-oriented vector selection algorithms instead of the more complicated overlapping scheme. The synthesis vectors (i.e. the columns) of the sub-matrices of $\mathcal{F}$, $(\mathbf{F})$, are still orthogonal to all columns in a translated sub-matrix (as they are in $\mathcal{G}$). This is the same as that $(\mathcal{F}^T\mathcal{F})$ is a block diagonal matrix and not a band diagonal matrix, $(\mathcal{G}^T\mathcal{G})$ is an identity matrix. Weights in one block have no effect on the weights in neighboring blocks. This is what makes it possible to use block-oriented vector selection algorithms.

The total synthesis system, specified by $\mathbf{F} = \mathbf{GH}$, has one fixed part and one part with free variables. For the example in Figure 4.5, $\mathbf{F}$ has a total of 1024 parameters, but only the 512 in $\mathbf{H}$ are free variables. We may ask if this reduction in degrees of freedom is important. To answer this we compared the synthesis system designed using the proposed method, with a similar 32×32 overlapping frame having all variables free, designed using the method in [2], i.e. the update of the frame step was like described in Section 2.3 and the update of the weights step as described in Section 4.3. We found that the overlapping frame designed with the present method performs best, [65]. The SNR for representation using different sparseness factors is shown in Figure 4.6. Here also the SNR for a block-oriented frame (of size 16×32) is presented. From this figure we can see that what is lost in degrees of freedom, for the tested case at least, is more than compensated for by what is gained in the vector selection step. It is more likely that the sub-optimal vector selection algorithms will find a solution close to the optimal solution for small problems than for larger problems. This test indicate that just having more free variables in an optimization problem does not necessarily give a better solution if the optimization algorithm can not handle the increase of free variables in a good way.

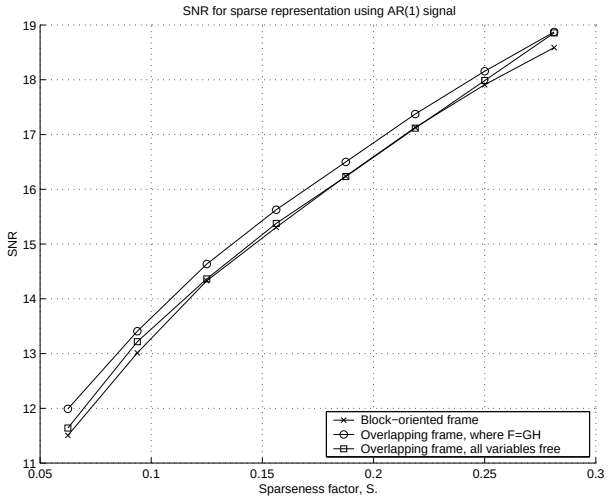


Figure 4.6: Approximation quality, measured by Signal to Noise Ratio (SNR), plotted as a function of sparseness factor. Here we compare the results on an AR(1) signal for a block-oriented frame ($\times$), for an overlapping frame with all variables free ($\square$), and for an overlapping frame with $\mathbf{F} = \mathbf{GH}$ as in Figure 4.5 ($\circ$).

We should also point out that the synthesis vectors we design are tied to the choice of orthogonal filter bank, $\mathcal{G}$. The columns of $\mathbf{F}$, the synthesis vectors of length PN, will be in the N dimensional subspace of $\mathbb{R}^{PN}$ spanned by the N columns of $\mathbf{G}$. To summarize: the main idea of this design method is that we may design an overlapping frame by selecting an appropriate orthogonal filter bank, $\mathbf{G}$, and design the frame $\mathbf{H}$ using established design procedures for block-oriented frames.

Chapter 5

Sparse Representation Experiments

In this chapter we present some experiments done to illustrate the capabilities of the proposed frame structures. Another goal is to present the complete design process using these examples. The focus throughout this chapter will be on sparseness alone.

5.1 Sparse representation of an ECG signal

In the first experiment a sparse representation is found for an electrocardiogram (ECG) signal. The purpose is to illustrate the sparse representation capabilities for different frame structures, not to exploit the sparse representation any further, for example in compression. Consequently, frames with different structures, denoted (a) to (e) and explained below, are designed for an ECG signal. To design a frame the following items must be handled:
1) decide the structure and size of the frame,
2) prepare the set of training vectors or the long training signal,
3) set the initial frame,
4) select a target sparseness factor used during frame design, denoted S_d.
5) and finally decide which vector selection algorithm to use.
When these decisions are made the frame can be designed by the iterative method outlined in Section 2.1.

The designed frames are then used to make sparse representations of a similar signal. The items to handle now are quite the same as during design, but the choices must be done in the context of the choices made during design.

1) choose one of the designed frames,
2) prepare the test vectors (or test signal) in a similar manner as the training vectors,
3) select the desired sparseness factor, to distinguish it from S_d this one is denoted S. As will be seen later, Figure 5.6, S could be in a quite wide range around the value of S_d.
4) decide which vector selection algorithm to use, usually this is the same as the algorithm used during frame design.
Now, sparse representations for each of the five frame structures can be made. In the end of this section the representation errors are compared.

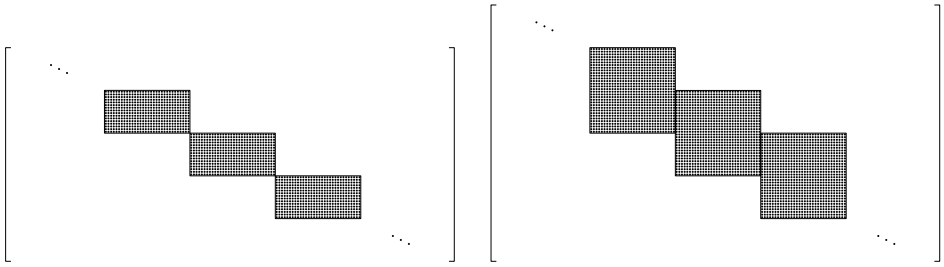
The five different frame structures are illustrated in Figure 5.1 and described in the following list

(a) The first structure is a simple block-oriented frame with size $N = 16$, $K = 32$ and $P = 1$. The number of free variables (all are free) is $Q = NKP = 512$. The design procedure for this frame is as described in Section 2.2. Note that this frame will not be suitable for a very small sparseness factor, having $S = \frac{1}{32}$ allows only one frame vector for every second signal block on average.

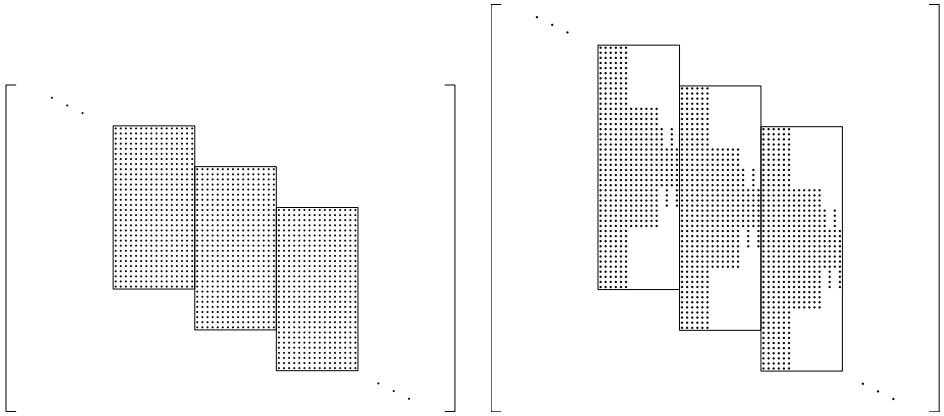
(b) This is an overlapping frame with size $N = 16$, $K = 32$ and $P = 2$. The number of free variables (all are free) is $Q = NKP = 1024$, and the design procedure for this frame is as described in Section 2.3. This frame is very similar to frame (a), if the last half of each frame vector is set to zero the situation should be exactly the same, thus it should be expected that frame (b) always performs as well as or better than frame (a).

(c) Another overlapping frame, this one with size $N = 8$, $K = 16$ and $P = 4$. The number of free variables (all are free) is $Q = NKP = 512$, the same number of free variables as in frame (a). The design procedure for this frame is as for frame (b).

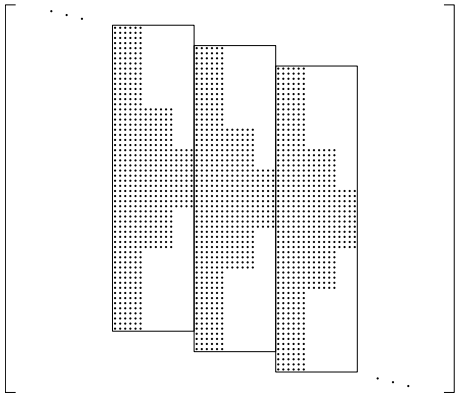
(d) This is a general overlapping frame with size $N = 8$, $K = 16$ and $P = 6$. A structure is imposed on this frame and the number of free variables is $Q = 246$. The design procedure for this frame is as described in Section 2.4. The structure is: Frame vectors 1-4 are equal except for translation of two samples, using $6 \cdot 8 + 6 = 54$ free variables. Frame vectors 5-6 are all free, $(2 \cdot 6 \cdot 8 = 96)$ variables. Frame vectors 7-12 are are all of length 24 and forced to be either odd or even symmetric, $(6 \cdot (24/2) = 72)$ free variables. Frame vector 13 and 15 are of length 12



(a) $N = 16$, $K = 32$, $P = 1$ **(b)** $N = 16$, $K = 32$, $P = 2$

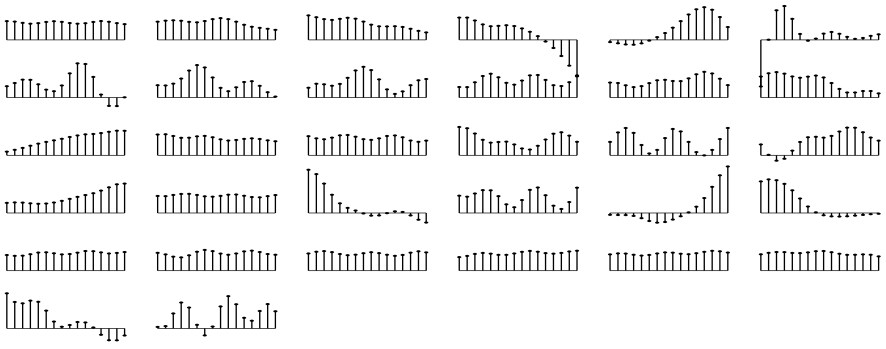


(c) $N = 8$, $K = 16$, $P = 4$ **(d)** $N = 8$, $K = 16$, $P = 6$

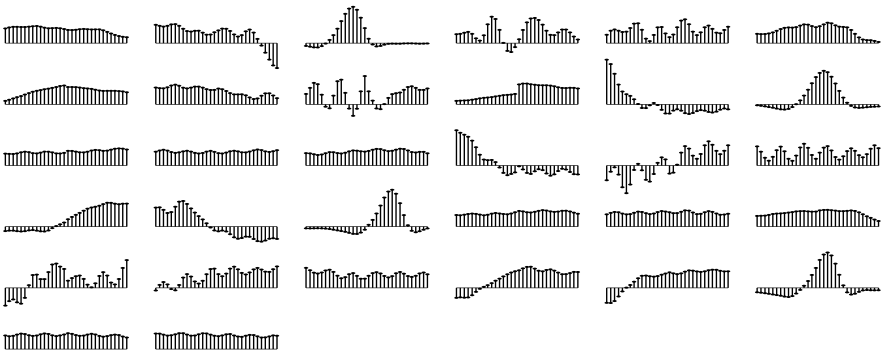


(e) $N = 4$, $K = 16$, $P = 15$

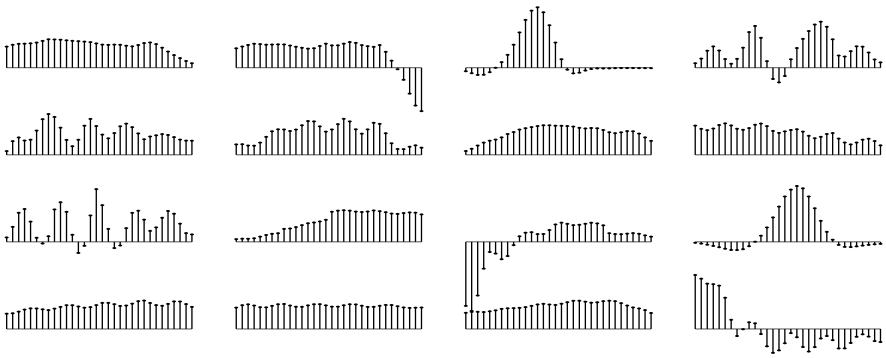
Figure 5.1: The structure of the five frames for sparse representation of ECG signal. For each frame the central part of $\mathcal{F}$ (3 repetitions of $\mathbf{F}$) is displayed, the non-zero entries are plotted as a dot and $\mathbf{F}$ is indicated by a box.



(a) $N = 16,\ K = 32,\ P = 1$

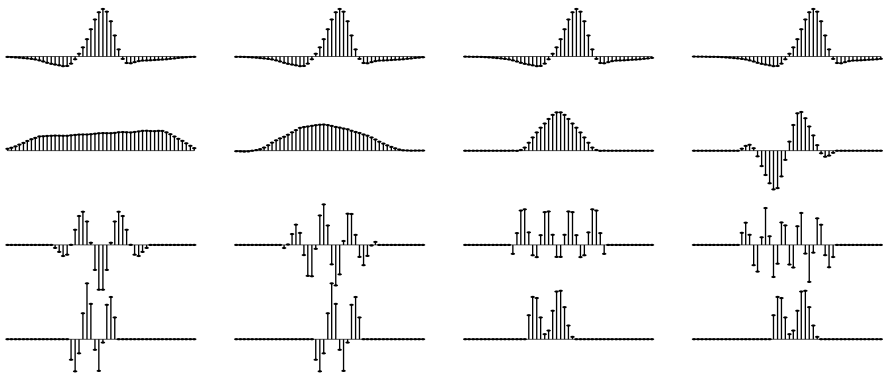


(b) $N = 16,\ K = 32,\ P = 2$

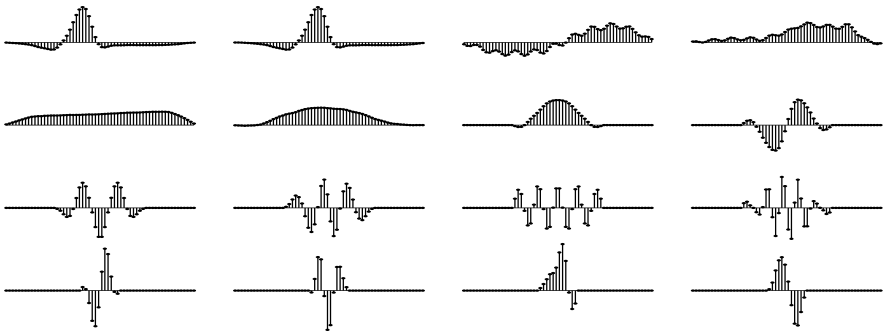


(c) $N = 8,\ K = 16,\ P = 4$

Figure 5.2: The synthesis vectors for the frames (a), (b) and (c) in Figure 5.1. The column vectors of the **F** matrix are plotted. The frames shown here have all $S_d = \frac{1}{8}$. Since the signal is mainly low frequency we see that the low frequency frame vectors are predominant. Note that the length of the synthesis vectors is 16 for (a), and 32 for (b) and (c).



(d) $N = 8$, $K = 16$, $P = 6$



(e) $N = 4$, $K = 16$, $P = 15$

Figure 5.3: The synthesis vectors for the frames (d) and (e) in Figure 5.1 and $S_d = \frac{1}{8}$. The column vectors of the $\mathbf{F}$ matrix are plotted. Note that some of the frame vectors, i.e. entries in the $\mathbf{F}$ matrix are forced to be equal, are symmetric or equal to parts of other frame vectors. In each frame, the first frame vectors, translated two samples relative to each other, strikingly resemble the tops in the ECG signal, Figure 5.4. The structures are further described in the text.

and all 24 variables are free, while frame vectors 14 and 16 are equal to 13 and 15 but translated 4 positions. The frame vectors after training are shown in the upper part of Figure 5.3.

(e) This is another general overlapping frame, this one with size $N = 4$, $K = 16$ and $P = 15$. The number of free variables is $Q = 434$, and the design procedure for this frame is as for frame (d). The imposed structure is: Frame vectors 1-2 are equal except for translation of two samples. Frame vectors 3-6 are all free variables. Frame vectors 7-12 are are all of length 28 and forced to be either odd or even symmetric. Frame vectors 13-16 are of length 12 and all free variables. This frame is different from the others in that the degree of overcompleteness, K/N, is 4 while it is 2 for the first four frames. The frame vectors after training are shown in the lower part of Figure 5.3.

Having decided for the frame structures to use, the steps 2-5 in the design procedure were set:

2) The first four minutes, 86400 samples, of an ECG signal, the "MIT100" signal from the MIT arrhythmia database [53], was used as training data. The mean of the signal was subtracted before the training vectors were made.

3) The frame design method starts with an initial frame. This initial frame is obviously important since the design method, if it converges at all, converges towards a local minimum of the object function. Most of the frame vectors was initialized by using segments from the training signal, exceptions are frame vectors 7-16 in frame (d) and (e), where basis vectors from the Lapped Orthogonal Transform (LOT) [46] or the Discrete Cosine Transform (DCT) was used.

4) For each of the five frame structures 10 frames were designed, where S_d varied from $\frac{1}{32}$ to $\frac{10}{32}$ with steps of $\frac{1}{32}$.

5) Vector selection was done by the algorithm in Section 4.3 using $M = 12$, 12, 14, 15 and 40 blocks for the frames (a) to (e) respectively. The core of the vector selection algorithm is the hybrid algorithm, Section 4.1.3. As seen in Figure 4.2 this is the one that performs best for training of frames. Within this hybrid algorithm again the ORMP algorithm is most frequently used.

Finally, training of the frames were done by the iterative method outlined in Section 2.1, more than 400 iterations were done on each frame.

Some comments can be given to the results of training. For each of the 50 designed frames, the training curve is very similar in shape to the bottom curve in Figure 4.2, the hybrid algorithm. The best SNR, Equation 1.8, values are

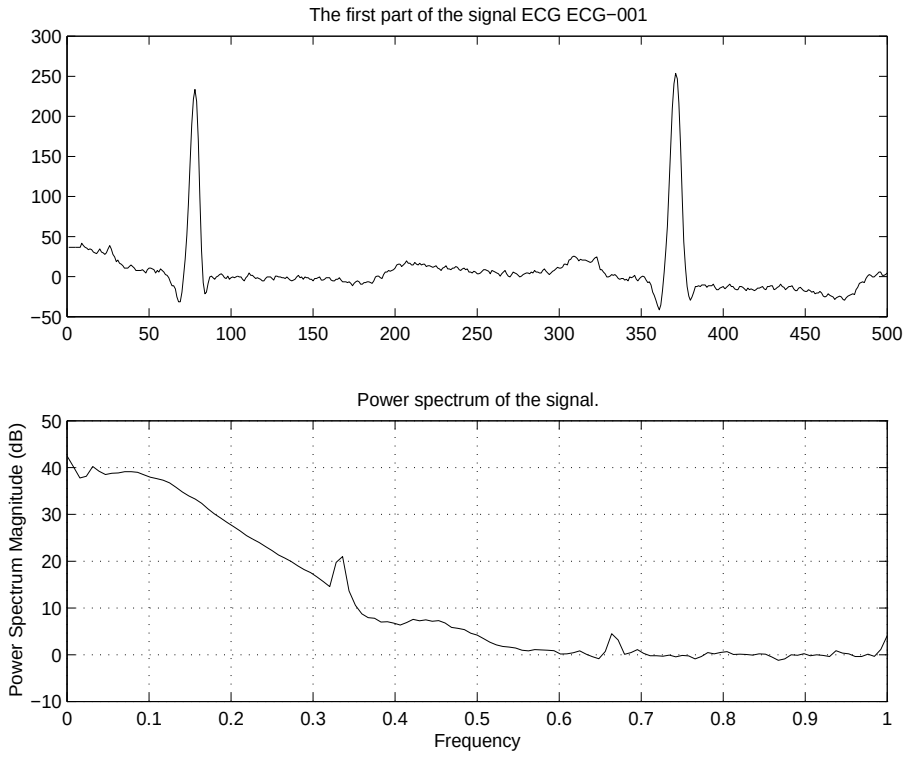


Figure 5.4: The first part of the ECG signal. In the power spectrum plot the frequency is normalized so 1 corresponds to 180 Hz, i.e. half the sampling frequency.

plotted in Figure 5.5. We note that frame (a) performs badly at low sparseness factor, this is as expected since the frame vectors are quite short. When the sparseness factor is higher though frame (a) do better than both frame (b) and frame (c). That frame (a) do better than frame (c), both frames have the same number of free variables, can be explained by assuming that to have many vectors to select from is more important than to have longer vectors and overlap. But frame (b) should ideally never do worse than frame (a), as easily can be seen by setting the bottom half of the frame matrix $\mathbf{F}$ to zero ($\mathbf{F}_1 = \mathbf{0}$) for frame (b), then frame (b) has exactly the same structure as frame (a). Nevertheless frame (a) do better than frame (b) for some sparseness factors. This can only be explained by assuming that the vector selection algorithm is more difficult for overlapping frames than for block-oriented frames. This was also observed in Section 4.4, see Figure 4.6. It can also be noted that the overlapping frame with structure, frame (d), do better than the frames

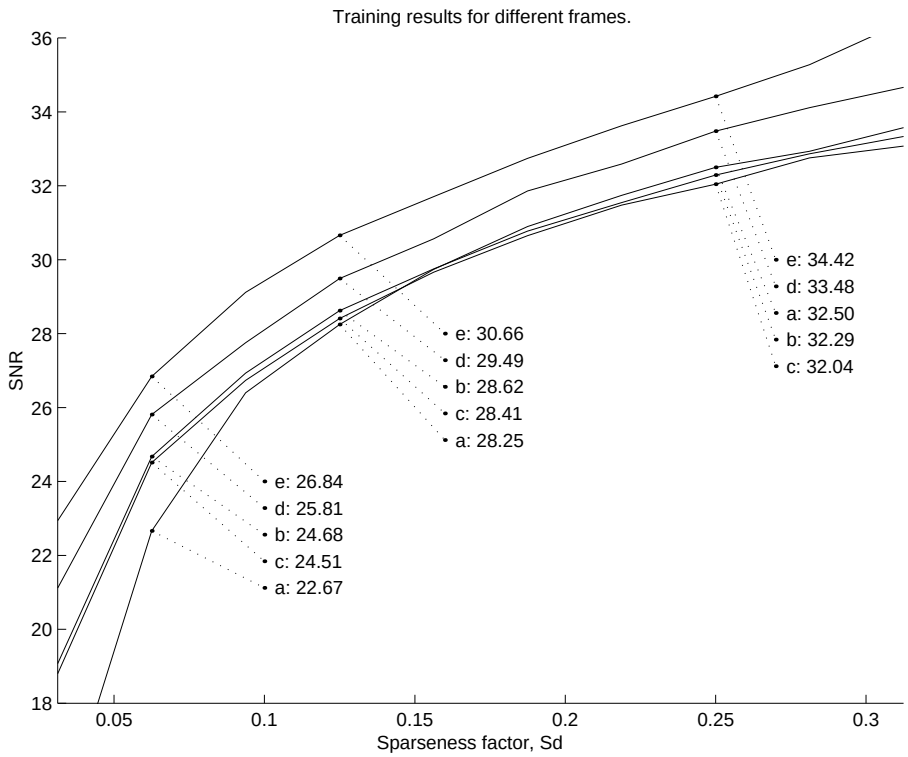


Figure 5.5: Final Signal to Noise Ratio (SNR) for the different frame structures in Figure 5.1 achieved during training. S_d is along the x-axis. Each of the 50 points, of which only 15 are marked and their respective values written, making up the graph is the final SNR during training of the frame, for example the point for frame (d) and $S_d = \frac{1}{16}$ (25.81) is the final SNR in Figure 4.3. Since the points for S_d are so dense we have permitted ourselves to draw lines through the points belonging to the same structure.

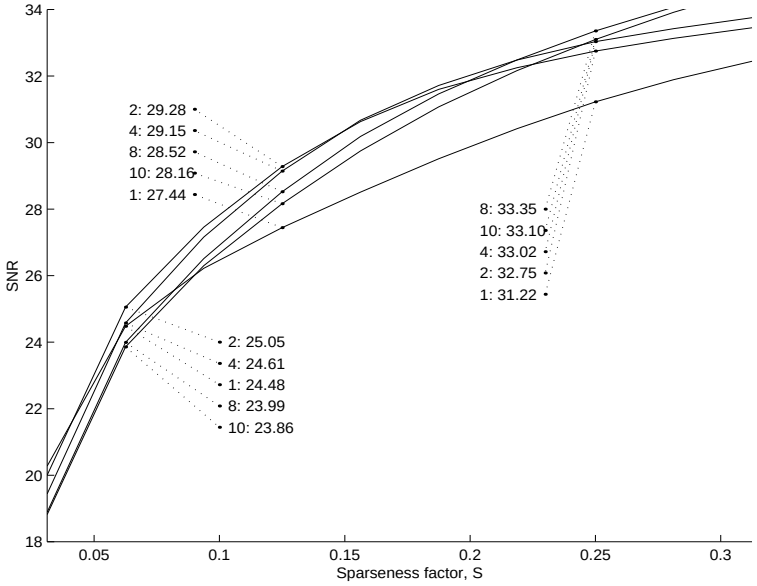


Figure 5.6: SNR values achieved on the test signal, using 5 different frames, all with structure (d) in Figure 5.1. S is along the x-axis. The frames were designed using $S_d = \frac{1}{32}, \frac{2}{32}, \frac{4}{32}, \frac{8}{32}$ and $\frac{10}{32}$. The numerators in these fractions are used in the figure to mark the plotted lines, this is done at the points where $S = \frac{1}{16}$, $S = \frac{1}{8}$, and $S = \frac{1}{4}$.

(a) to (c), even if frame (d) has fewer free variables than the other frames. It performs noticeably better at all values of S_d. The structure is more important than the number of free variables. Frame (e) is even better than frame (d), but remember that this frame has a larger degree of overcompleteness, here $K/N = 4$, twice as much as for the other frames.

We will now test the designed frames on a test signal. The test signal is from the same ECG signal (MIT100) as used during training, but from 5 minutes to 10 minutes, and the length is 108000 samples. In Figure 5.6 five frames, all with structure (d) and different values for S_d ($\frac{1}{32}$, $\frac{2}{32}$, $\frac{4}{32}$, $\frac{8}{32}$ and $\frac{10}{32}$), are compared. The five frames were used to make sparse representations of the test signal, now with sparseness factor varying from $S = \frac{1}{32}$ to $S = \frac{10}{32}$ with steps of $\frac{1}{64}$. The SNR values are presented in Figure 5.6. We see that the frame with $S_d = \frac{1}{32}$ performs quite poorly for $S > 0.1$, and the frames with $S_d = \frac{8}{32}$ and $S_d = \frac{10}{32}$ is not as good as the other frames for small values of S. The frames with $S_d = \frac{2}{32}$ and $S_d = \frac{4}{32}$ are quite good for $0.05 < S < 0.2$. The conclusion that can be made from this is that a frame can be used for a wider range of

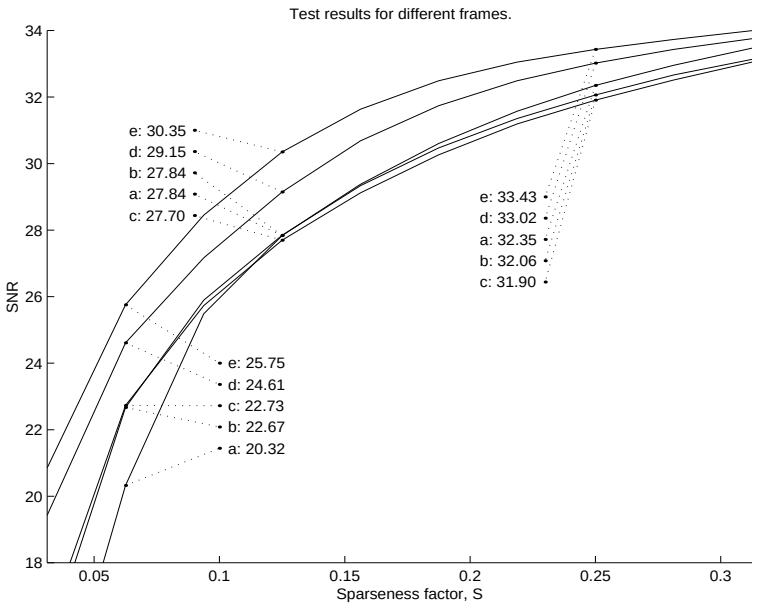


Figure 5.7: The achieved SNR for sparse representation of the test signal for the different frame structures in Figure 5.1. S is along the x-axis. For these five frames $S_d = \frac{4}{32}$. Note that line (d) in this figure is the same as line (4) in Figure 5.6

sparseness factors than the specific one it was designed for, for example we may have $S_d/3 < S < 3S_d$. If only the frames with $S_d = \frac{4}{32}$ are used we will expect the representation results to be quite close to the representation results achieved if the frames were designed with "correct" sparseness factors. This will be so for almost the whole range of S used in the figures in this section.

In Figure 5.7 frames with different structures are compared. The frames are shown in Figure 5.2 and Figure 5.3, they were designed using $S_d = \frac{4}{32}$. The SNR values here are a little bit below, approximately 0.5 decibel, the values achieved during training, Figure 5.5. The conclusion both for training and for representation of the test signal is that the overlapping frames with structure, (d) and (e), perform best. What about alternative sparse representations one may ask. The answer is that the frames perform considerably better than thresholding the coefficients of a transform or a filter bank, for $S = \frac{1}{8}$ the SNR achived by thresholding coefficients for the DCT, the LOT and Extended Lapped Transform (ELT) [45] filter banks, and wavelet filter banks are in range from 23.8 to 24.6. As seen in Figure 5.7 the corresponding range for the frames is from 27.7 to 30.3.

5.2 Sparse representation of images

The goal of this experiment is to demonstrate the capabilities of sparse representation of images for some of the frame structures described in the earlier chapters. The well known Lena image is used as the test image. Four frame structures are used and for the sake of comparison we also include a fifth case where we do thresholding of the coefficients of an orthogonal filter bank. Some of the results here has been presented earlier, [63].

5.2.1 The different frame structures

The five frame structures presented here are

1. A block-oriented frame where we have $N = 64$, i.e. the block size is $N_1 \times N_2 = 8 \times 8$, $K = 128$ and $P = 1$. The number of free variables is $Q = NKP = 8192$.

2. An overlapping frame constructed as a block-oriented frame in the coefficient domain, Section 4.4. The filter bank used is the ELT [45], it has 64 different basis images which are shown in Figure 5.8. Each frame image is a linear combination of these basis images. For the constructed block-oriented frame, denoted $\mathbf{H}$ in Section 4.4, we have $N = 64$, $K = 128$ and $P = 1$. The number of free variables is $Q = NKP = 8192$.

3. An overlapping two-dimensional (2D) frame as in Section 3.3. The size of the frame (after reshaping) is $N = 64$, $K = 128$ and $P = 4$, i.e. the overlap in vertical and horizontal direction is $P_1 = P_2 = 2$. Each of the $K = 128$ frame images, is of size 16×16. The number of free variables is $Q = NKP = 32768$, which is a quite large number for an optimization problem.

4. A separable 2D block-oriented frame as in Chapter 3.6.1, $\mathbf{F}_v$ and $\mathbf{F}_h$ have size 8×16. There is $K = K_1 \cdot K_2 = 16 \cdot 16 = 256$ frame images each of size $N_1 \times N_2 = 8 \times 8$. Note that the number of free variables is only 256, but the degree of overcompleteness is larger than for the three previous structures, here $K/N = 4$ while for structures 1 to 3 $K/N = 2$.

5. For the sake of comparison we also use the ELT orthogonal filter bank alone, sparseness is introduced by thresholding its coefficients. Note that this is a complete expansion, $K/N = 1$.

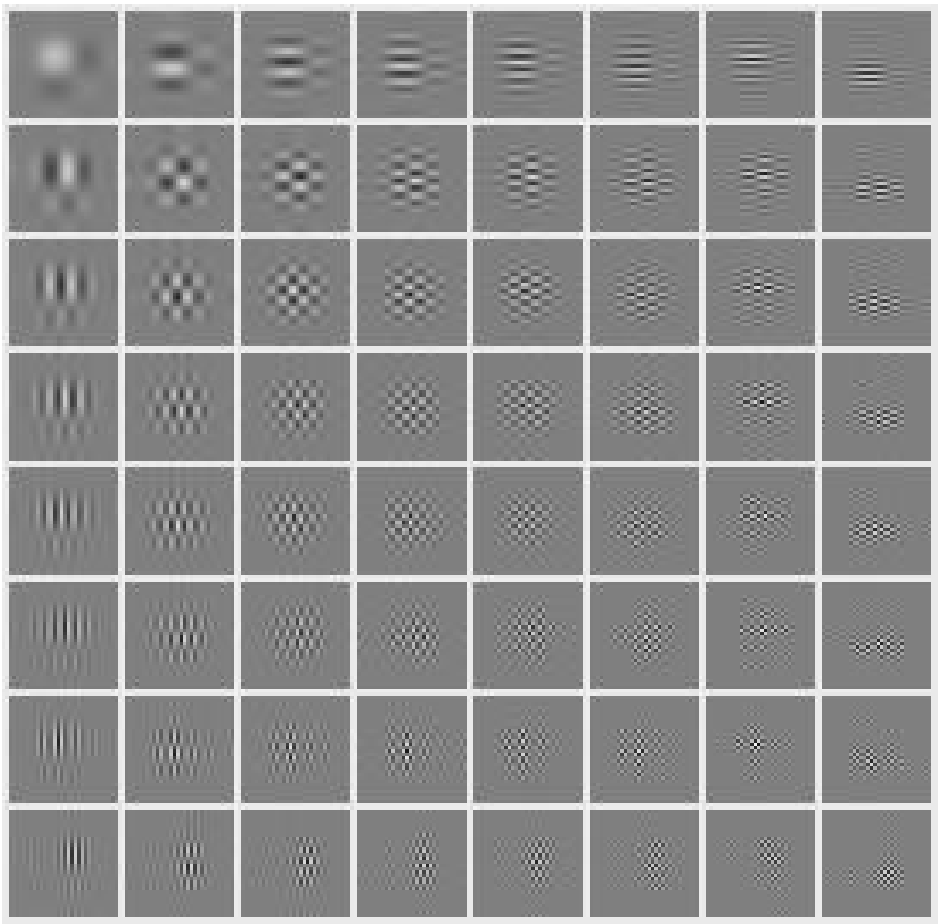


Figure 5.8: The basis images of the Extended Lapped Transform (ELT) filter bank. Each basis image has size 32×32, and are translated by factors of 8 in vertical and horizontal direction.

Figure 5.9: The training set consists of 8 images, each has size 512×512 pixels, 256 gray levels.

5.2.2 Preparing the training vectors

For the one-dimensional signal this part of the design procedure is quite trivial (subtracting the mean and chopping the training signal into blocks), but for the different frames used for the 2D signals many steps are involved. Also the training sets are prepared in a way depending on the frame structures. Let us start by looking at the training images, Figure 5.9, these are used for all the four different frame structures. The training set consists of $M = 8$ images, each has size 512×512 pixels, 256 gray levels.

Before these are used for training we subtract a low-pass image. Several observations motivate subtracting a low-pass image from the original image, the resulting image is called the "high-pass" image, even if it contains most of the information in the image, it is of the same size as the original image. Vector selection schemes have been shown experimentally to work well when the signal has zero mean, [22]. Due to the large variation in the mean in different parts of an image, it is more advantageous to subtract a locally varying mean than a global mean. Similarly, in DCT based coding schemes (JPEG) the DC components of each image block are handled separately. This process is illustrated in Figure 5.10 for the test image, but it is just the same for the training images. The lower right image is the image used to generate the training vectors (for "Lena" it will be test vectors). The low-pass image (lower left image in the

Figure 5.10: The original test image, Lena, on the top. Below: the image is split into a low-pass filtered image to the left and a high-pass (the original subtracted the low-pass filtered image) image to the right.

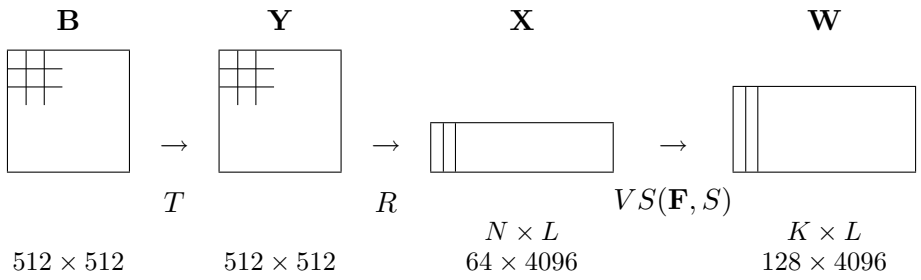


Figure 5.11: The analysis part of the representation process for an image, $\mathbf{B}$ or $\mathbf{B}_m$. T is an optional (T may be the identity operator) orthogonal filter bank that transforms the image into a coefficient domain, $\mathbf{Y}$. R is a reorder operator that organizes image blocks into column vectors. $VS(\mathbf{F}, S)$ is the vector selection function that uses the current frame $\mathbf{F}$ and the sparseness factor S (or S_d during design).

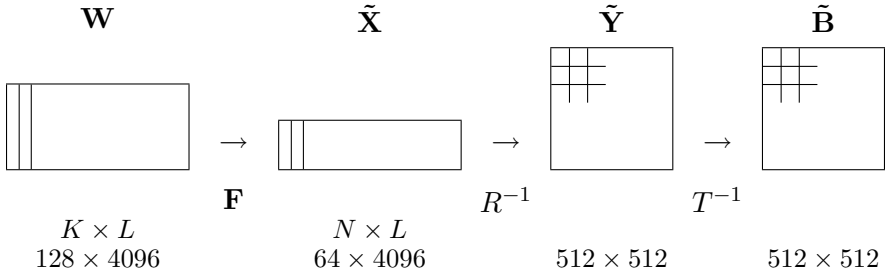


Figure 5.12: The synthesis part of the representation process generates the reconstructed image, $\tilde{\mathbf{B}}$, based on the sparse weights $\mathbf{W}$. R^{-1} and T^{-1} are the inverse operators of R and T in Figure 5.11.

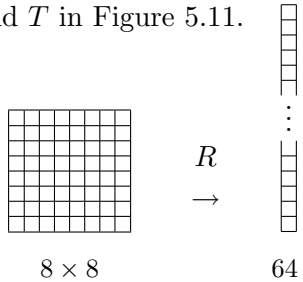


Figure 5.13: The operator R reshapes an 8×8 block into a 64×1 column vector.

figure) is represented by only one byte per 256 samples (0.03 bits per pixel), i.e. a 512×512 image is represented by only 1024 bytes. Since the frames are used only on the high-pass images, the low-pass filtered images must be stored separately, this contributes with $S_{lp} = \frac{1}{256}$ in the (total) sparseness factor. The sparseness factor used to find how many frame vectors that can be used is then $S = S_t - S_{lp} = S_t - \frac{1}{256}$, where S_t is the total sparseness factor. The S used during design is denoted S_d as for the ECG signal. The high-pass training images, denoted $\{\mathbf{B}_m\}_{m=1}^{M}$, are used during frame design.

Design of the 2D frames is done by the method outlined in Section 2.1, and the steps in this method are embellished in Chapter 2-4. But there is some special points which must be handled with care. To get a better overview of the design process for this particular case, the analysis part for a sparse representation of an image, here the high-pass image $\mathbf{B}$ or $\mathbf{B}_m$, is illustrated in Figure 5.11 and the corresponding synthesis system in Figure 5.12. T is an orthogonal linear operator, and T^{-1} its inverse. This operator is only used if frame design is done in the coefficient domain, Section 4.4, so often we may skip this. When T is not the identity operator it represents a separable orthogonal filter bank. R too is an orthogonal linear operator. It reorders each 8×8 block into vectors and puts these into a matrix $\mathbf{X}$. $VS(\mathbf{F}, S)$ is a vector selection procedure. With appropriate operators these figures for the analysis and synthesis system can illustrate the procedure used for all the frame structures introduced in Subsection 5.2.1.

1. A block-oriented frame is obtained when the T operator is identity.

2. The overlapping frame constructed as a block-oriented frame in the coefficient domain, as used in this section, is obtained when T operator is the ELT orthogonal filter bank.

3. An overlapping frame with all variables free has a more complicated structure than the two previous frame structures. As for the block-oriented frame, the T operator is identity. The increased complexity is in vector selection and the calculation of $\tilde{\mathbf{X}}$, since the overlapping structure of the frame must be taken account for. The method for the 2D overlap vector selection is a 2D extension of the method in Section 4.3. During design, updating the frame is as described in Section 3.3.

4. Also for the separable 2D block-oriented frame the T operator is identity. The difference from frame structure 1 is that the frame $\mathbf{F}$ is defined by its generating parts, $\mathbf{F}_v$ and $\mathbf{F}_h$. During design, updating these matrices must be done as described in Section 3.6.1. Vector selection is block-oriented as for frame structures 1 and 2.

5. For the ELT orthogonal filter bank case we let the T operator be the ELT. Now, vector selection is just thresholding of the coefficients, and on the synthesis part the reconstructed signal vectors in the coefficient domain is simply $\tilde{\mathbf{X}} = \mathbf{W}$.

We should note that the number of free variables for the different frame structures varies much. It is quite small for the separable case, and very large for the overlapping case. The degree of overcompleteness varies from $K/N = 1$ in the ELT case to $K/N = 4$ for the separable frame.

5.2.3 Training and testing the frames

In Figure 5.14 and Figure 5.15 the results after frame design and for the sparse representation test using image Lena are plotted with Peak Signal to Noise Ratio (PSNR) as a function of sparseness factor. The PSNR is a common measure of image quality and is calculated as:

$$PSNR = 10 \log_{10}(\frac{255^2}{MSE}). \tag{5.1}$$

since 255 is the largest possible pixel value in an 8 bit per pixel image. The MSE is calculated as in Equation 1.7. The five curves are marked by numbers from 1 to 5 corresponding to each of the four different frame structures and the ELT filter bank, as described in the previous subsections.

In Figure 5.14 the average PSNR for the eight training images achieved during frame design is plotted. For each frame structure frames were designed using $S_d \in \{\frac{1}{256}, \frac{2}{256}, \frac{3}{256}, \frac{5}{256}, \frac{7}{256}, \frac{15}{256}, \frac{23}{256}\}$, a line is plotted through the points belonging to each of the frame structures. Also the average PSNR of thresholding the ELT coefficients of the training images is plotted, here more points are calculated and this line is smoother.

At a first glance we may be surprised that the PSNR is higher in the test results than in the design results, but then we should remember that the training images, Figure 5.9, are different from the test image, Figure 5.10. At a given sparseness factor the PSNR varies much from image to image, the low-pass filtered training image number 2 (lake) has PSNR=19.45 and image number 7 has PSNR=26.73. The average of the PSNR values for the 8 training images is 22.00 decibel when all weights are zero, i.e. for the low-pass filtered images. For the low-pass filtered Lena image in Figure 5.15 PSNR=23.66. In this section the test image is different (has different properties) from the

training images, while in the previous section the test signal and the training signal are just different parts of the same ECG signal.

The x-axis in the figures must also be commented on. In Figure 5.14 it is the total sparseness factor in design, $S_t = S_{lp} + S_d$, where $S_{lp} = 1/256$ is the contribution of the low-pass image and S_d is the sparseness factor actually used during design. In Figure 5.15 the x-axis is the total sparseness factor in sparse representation, $S_t = S_{lp} + S$, where S is the sparseness factor actually used in vector selection.For both figures the PSNR values for each of the curves are written in the figure for $S_t = 2/256$, ($4/256$ for Figure 5.15 only), $8/256$, and $16/256$. For the test image Lena we see that all five curves start at $S_t = S_{lp} = 1/256$ and PSNR=23.66 which is when no weights, $S = 0$, are used to represent the high-pass image, and of course this value is the same for all frames and the ELT case.

The frames used in Figure 5.15 are the ones designed with $S_d = 3/256$ ($S_t = 4/256$). The range of S for which these frames are suitable is, in accordance with the results in the end of Section 5.1, $1/256 < S < 9/256$. The results here confirm this; for S_t larger than 0.03 the ELT performs relatively better as S_t increase. In the recommended range for the sparseness factor all frames have 0.5 to 1.5 decibel higher PSNR than the ELT. The difference is only marginally larger for the training images, indicating that the frames are not overtrained to represent only the 8 test images well but that the frames will work quite well for all images "of the same class" as the training images.

Comparing the different frames to each other we get slightly different conclusions by looking at the training results rather than looking at the test results. From Figure 5.14 the overlapping frame seems best at low sparseness factors and the separable frame best at higher sparseness factors, but in Figure 5.15 the crossing point between these two curves are at a lower sparseness factor and generally the overlapping frame is not a clear winner except for lowest value of S_t. The overlapping frame (3) has many free variables, thus being more vulnerable to the risk of overtraining, while the opposite is true for the separable frame (4).

The three frames, (1), (2), and (4) achieve quite similar results, even though they are quite different from each other. For all sparseness factors the frame in the coefficient domain (2) does slightly better than the block-oriented frame (1). The separable frame (4) have more frame images to select among, the degree of overcompleteness is $N/K = 4$, while it is $N/K = 2$ for frames (1) and (2). We see that frame (4) does best for larger sparseness factors, probably the larger ratio for N/K is the main reason for this. For lower sparseness factors the rigid structure for the frame images of the separable frame is most

likely to explain that the block-oriented frame (1) with fewer frame images performs better. The fact that all variables in the frame images were freely selected during frame design for frame (1) seems to be more important when the sparseness factor is small than the degree of overcompleteness.

The overlapping frame (3) is good at small sparseness factors, but worse than the other frames for larger values of S. The main reason for this, I believe, is that vector selection gets very difficult and computationally demanding, especially when many frame images should be selected, for the 2D overlapping frame. For frames (1), (2) and (4) vector selection is done with a quite simple block-oriented algorithm. The overlapping 2D vector selection algorithm is more complicated, and it has not been used as much as the block-oriented algorithm and consequently it is not as thoroughly tested. The fact that the overlapping 2D vector selection algorithm apparently works well for small values of S but falls behind for larger values of S indicates that it is the size of the problem that cause the difficulties. For the 1D overlapping vector selection algorithm, Section 4.3, the starting point is the synthesis equation, $\tilde{\mathbf{x}}^M = \mathbf{F}^M \mathbf{w}^M$, Equation 4.3. The size of the $\tilde{\mathbf{x}}^M$ vector is wanted large to avoid relatively more of the interference from the neighborhood, but it must not be too large because of the complexity of vector selection. In the 2D case it is even more difficult to reduce the neighborhood interference, since it is imperative to restrict the size of the vector selection problem. The used size of the $\tilde{\mathbf{x}}^M$ vector (before it is reshaped to a vector it was 3×3 blocks of size 8×8) was 576×1, and this is probably too small to avoid much of the overlapping effects. The problems with 2D overlapping vector selection is more serious when many vectors should be selected, this partly explains why frame (3) performs better than the other frames for $S_d < 0.04$ and worse for $S_d > 0.05$ during design, Figure 5.14.

For a qualitative judgement of the image quality, we look at different sparse representations of the Lena image. In Figure 5.16 one part of the original Lena image and its sparse representations using the five frame structures listed in Subsection 5.2.1 are shown. The sparseness factor is $S_t = 4/256$ for all five frame structures. We clearly notice the blocking effect for the block-oriented frame (1) and the separable frame (4), which is also block-oriented. The overlapping frame (3) also show some blocking effect, this is easy to understand if we look at the frame images after training. They do not smoothly decline toward zero at the edges (as for example the ELT basis images do, Figure 5.8) and many of them are blocky alone, i.e. we see that they consist of 4 8×8 blocks. The blocking effect of the overlapping frame is less noticeable partly because it has a higher PSNR value. For "frame structure" (5), largest ELT coefficients, the image is more smooth that for the other structures, also some

ringing effects are visible, for example the edge of the hat is "ringed" to the chin below the eye. The frame in the coefficient domain (2) is perhaps the structure that gives the better visual impression. The PSNR values for the five reconstructed images are 31.52, 32.08, 32.37, 31.27 and 30.38 for the frame structures (1) to (5) respectively, these numbers can be read from Figure 5.15.

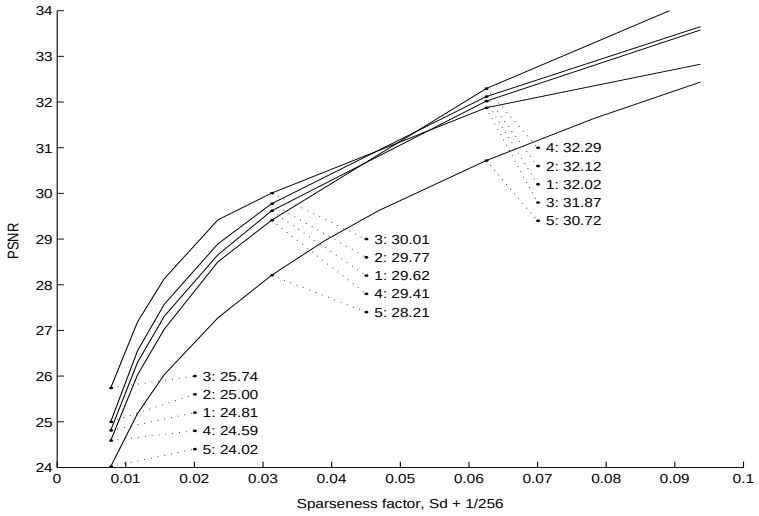


Figure 5.14: Average Peak Signal to Noise Ratio (PSNR) for the training images achieved during frame design. 1: Block-oriented frame, 2: ELT + block-oriented frame, 3: overlapping frame, 4: separable frame, and 5: thresholding ELT, note that this latter "frame structure" is not trained.

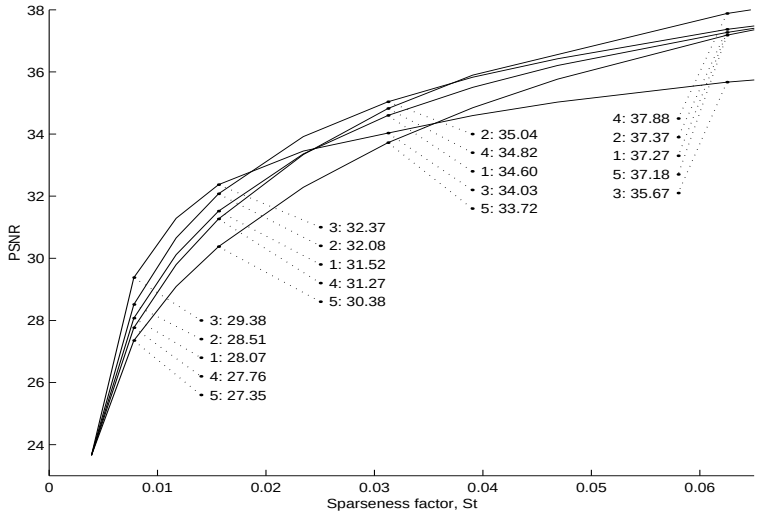


Figure 5.15: PSNR for test image Lena. 1: Block-oriented frame, 2: ELT + block-oriented frame, 3: overlapping frame, 4: separable frame, and 5: thresholding ELT.

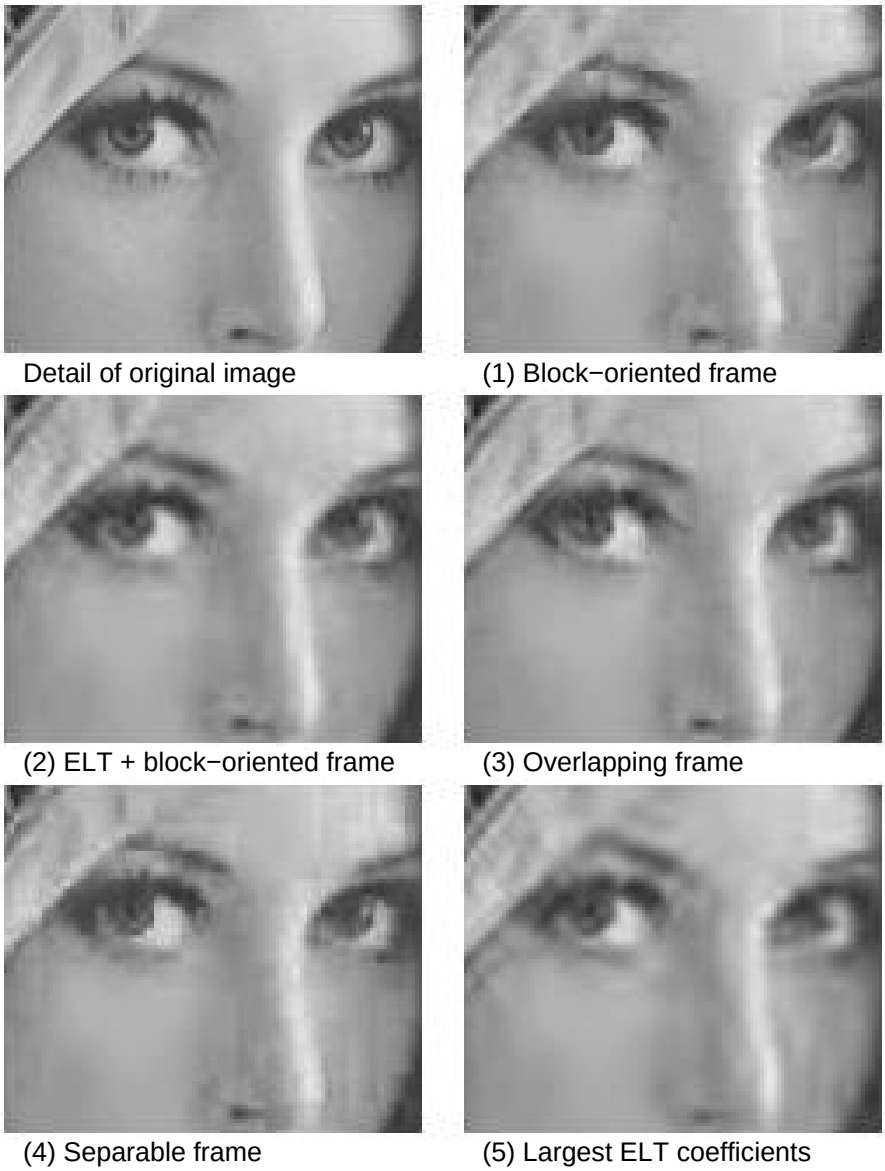


Figure 5.16: A detail of the test image Lena, different sparse representations.

Chapter 6

Texture Classification

In this chapter texture classification using frames is presented. The method is denoted as the Frame Texture Classification Method (FTCM). The main idea is that a frame trained to represent a certain class of signals is a model of this signal class. The signal class is given by many representative blocks of the class, these blocks, reshaped into vectors, constitute the training set. For image textures the training set is made from many small, typically 5×5 or 7×7, blocks of the texture image examples under consideration. Frames are trained for several textures, one frame for each texture class. A pixel of an image to be classified, – an image in a test set, is classified by processing a block around the pixel, the block size is the same as the one used in the training set. Many sparse representations of this test block (vector) are found, using each of the frames trained for the texture classes under consideration. Since the frames were trained to minimize the representation error, the tested pixel is assumed to belong to the texture for which the corresponding frame has the smallest representation error. Before the FTCM is presented in more detail in Section 6.2 a short introduction to texture classification is given in Section 6.1. Some results obtained using FTCM are presented in Section 6.3.

6.1 Introduction to texture classification

Most surfaces exhibit texture. For human beings it is quite easy to recognize different textures, but it is more difficult to precisely define a texture. A simple definition could be: a texture may be regarded as a region where some elements or primitives are repeated and arranged according to a placement rule. Tuceryan and Jain [68] list more possible definitions, and give a more

complete overview of texture classification. Texture classification using vector quantization [48] is an interesting approach since the method we propose here, FTCM, may be regarded as a generalization of the vector quantization approach.

Texture is a local property of an image, but it is not confined to a single point, this means that to decide the texture of a point a small area around it must be included. When we say that a pixel belong to a certain texture we mean that a block around that pixel has some properties that define it as belonging to this texture. If the block must be large to properly identify the texture,for example 100×100 pixels, the texture is coarse, while if the texture is well defined (recognizable) for a small block, for example 7×7 pixels, the texture is said to be fine.

Texture information together with color information and edge detection, is important for humans for identifying objects within natural images. For the same reason texture analysis, recognition and classification are often parts of machine vision systems and image processing algorithms. The possible applications are not limited to natural images, texture analysis is also used in areas as geophysical surveying [42], medical analysis [26] [56] [49] [40] and satellite Synthetic Aperture Radar (SAR) image analysis [74] [54]. Several methods for texture discrimination have been proposed, an overview is given by Tuceryan and Jain [68]. They group texture identification methods into four major categories: statistical, geometrical, model-based, and signal processing methods.

Randen and Husøy [59] have presented a comparative study of different texture classification methods. The focus of their paper is on signal processing methods (filtering), but they also include results where co-occurrence (statistical) and autoregressive (model-based) features are used. The classification experiments in this chapter are done using the same training textures and the same test images as the ones used in [59]. Also the experimental setup will be quite similar. The setup of the filtering approach system they used is presented in Figure 6.1. The first operation is filtering of the input image, this is usually done by a bank of filters, the result is several "images", each being the response of one filter. Next a local energy function is applied, consisting of a nonlinearity, usually magnitude or squaring, which is the same as the energy after filtering, succeeded by a smoothing filter. The dashed box for the second nonlinearity indicate that this operation may be omitted. For an image pixel a feature vector is formed by collecting the entries corresponding to the pixel position from the processed filter responses. The length of the feature vector will be the same as the number of filters used in the first step. The feature vectors are finally classified using a vector classification algorithm.

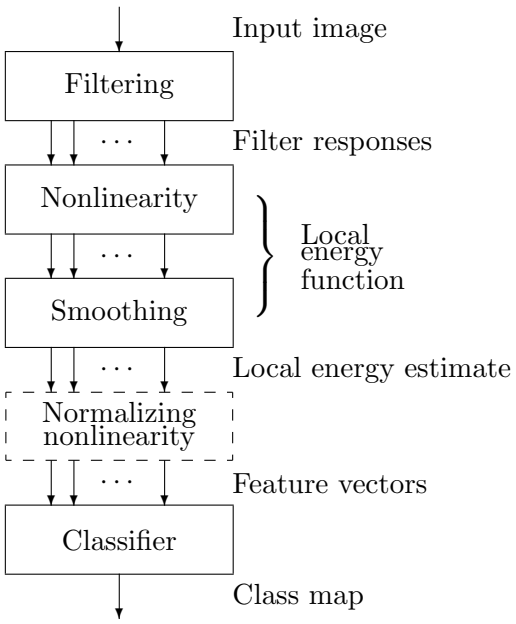
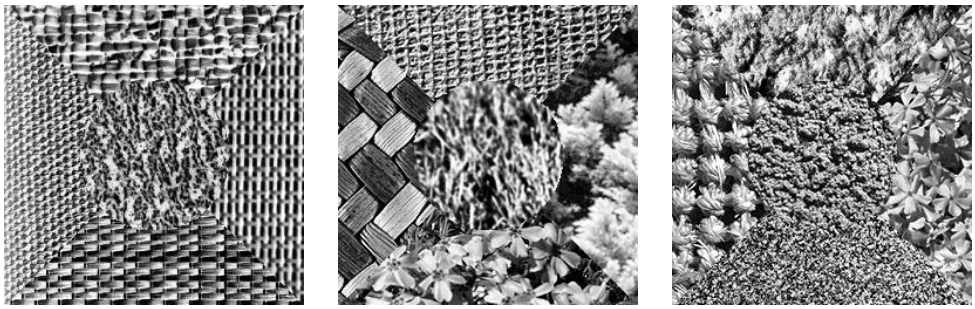


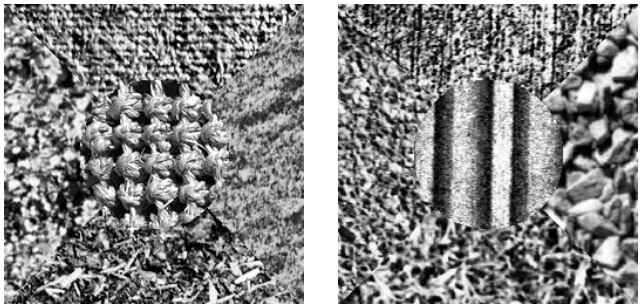
Figure 6.1: Experiment setup for the filtering approach used by Randen and Husøy.

Method	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
f8a	7.2	21.1	23.7	18.6	18.6	37.5	43.2	40.1	29.7	26.6
f16b	8.7	18.9	23.3	18.4	17.2	36.4	41.7	39.8	28.5	25.9
Daub-4	8.7	22.8	25.0	23.5	21.8	38.2	45.2	40.9	30.1	28.5
J_{MS}	16.9	36.3	32.7	41.1	43.0	47.3	51.1	59.7	49.9	42.0
J_U	12.7	33.0	26.5	34.3	43.4	45.6	46.5	35.9	30.5	34.3
Co-occurrence	9.9	27.0	26.1	51.1	35.7	49.6	55.4	35.3	49.1	37.7
Autoregressive	19.6	19.4	23.0	23.9	24.0	58.0	46.4	56.7	28.7	33.3

Table 6.1: Classification errors, given as percentage wrongly classified pixels, for different methods (rows) and different test images (columns) as presented by Randen and Husøy in [59].



(a) (b) (c)



(d) (e)

Figure 6.2: The five test images with 5 textures in each.

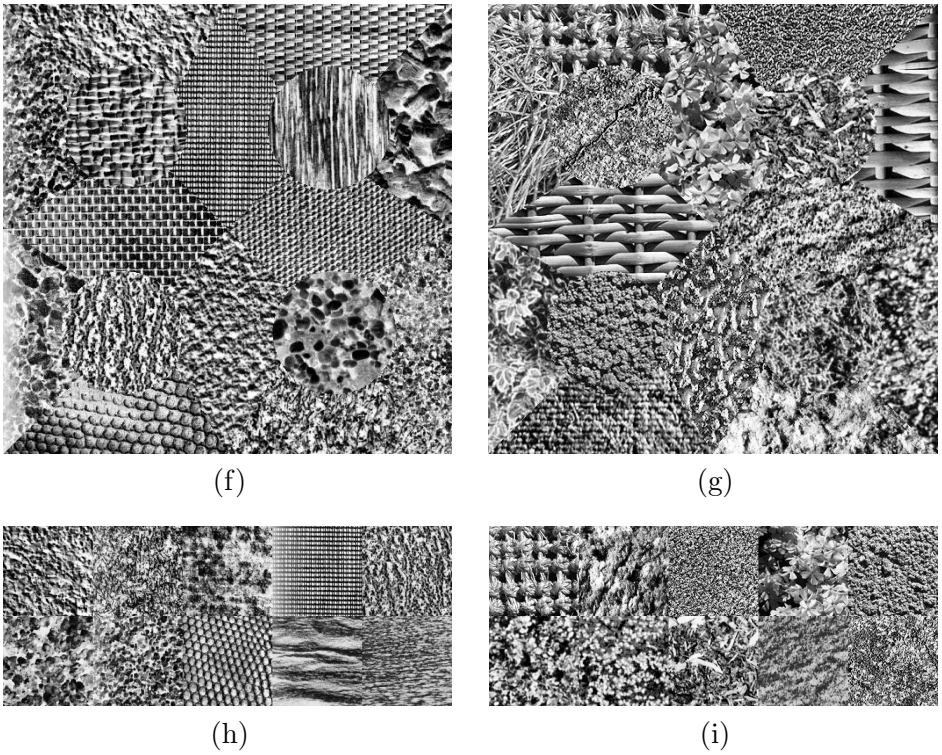
In [59] Randen and Husøy tried a lot of different filter banks in the filtering step. For smoothing they concluded that the separable Gaussian low-pass filter is the better choice. The unit pulse response for this filter is

$$h_G(n) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{n^2}{\sigma^2}}. \tag{6.1}$$

The parameter σ gives the bandwidth of the smoothing filter. For band-pass filters (in the filtering step) they calculated σ as

$$\sigma = \frac{1}{2\sqrt{2}f_0} \tag{6.2}$$

where f_0 is the radial spatial center frequency. This smoothing function was initially suggested in [37]. For the methods denoted "J_{MS}" and "J_U" in [59], referred in Table 6.1 and explained below, the value of this parameter was $\sigma = 8$.



(f) (g)

(h) (i)

Figure 6.3: The 16-textures and 10-textures test images.

For the first and second nonlinearity illustrated in Figure 6.1, in line with the conclusion of Unser [70], they used squaring in combination with a logarithmic normalizing nonlinearity. For classification they chose to use the "Type One Learning Vector Quantizing" (LVQ) supervised classifier of Kohonen [38] for most experiments.

The extensive results of [59] are presented in many tables, from which we have extracted some of their best results and presented them in Table 6.1. The table shows the classification errors, given as percentage wrongly classified pixels, for different methods (rows) and different test images (columns). The first five columns are for the 5-texture test images shown in Figure 6.2, the columns denoted (f) and (g) are for the 16-texture test images in upper part of Figure 6.3, the next two columns are for the 10-texture test images in lower part of Figure 6.3, and finally the last column is the mean of the classification error for the nine test images used. The test images in Figure 6.2 and Figure 6.3 are the same as the test images (a) to (i) in Figure 11 in [59].

The methods denoted "f8a" and "f16b" use a tree structured bank of quadra-

ture mirror filters (QMF), the filters are finite input response (FIR) filters of length 8 and 16, respectively. The method denoted "Daub-4" use the Daubechies filters [17] of length 4, and the same structure as used for the QMF filters, the referred results use the non-dyadic subband decomposition illustrated in Figure 6d in [59]. The methods denoted "J_{MS}" and "J_U" are FIR filters optimized for maximal energy separation, [60]. For the J_{MS} method the filters are designed to maximize the ratio between the extracted mean feature values,

$$J_{MS} = \frac{\mu_{v_1}}{\mu_{v_2}}. \tag{6.3}$$

where μ_{v_i} is the mean feature value for texture i. For the J_U method optimization was done with respect to the criterium

$$J_U = \frac{(\mu_{v_1} - \mu_{v_2})^2}{\mu_{v_1}\mu_{v_2}}. \tag{6.4}$$

The last two methods use co-occurrence (statistical) and autoregressive (model-based) features. For more details of the classification methods referred and results of more methods we recommend [59]. The results in Table 6.1 are directly comparable to the results of the proposed method, FTCM, that will be presented in the end of this chapter.

6.2 Frame texture classification method

The frame texture classification method (FTCM) is presented in Figure 6.4. The method can be used for supervised classification, i.e. example images of the possible textures are available and we know which textures are used in the test images. In this section we will describe the different parts of FTCM.

6.2.1 Preprocessing

The very first step in the FTCM is to decide the frame parameters. These parameters can be chosen quite freely, they are:

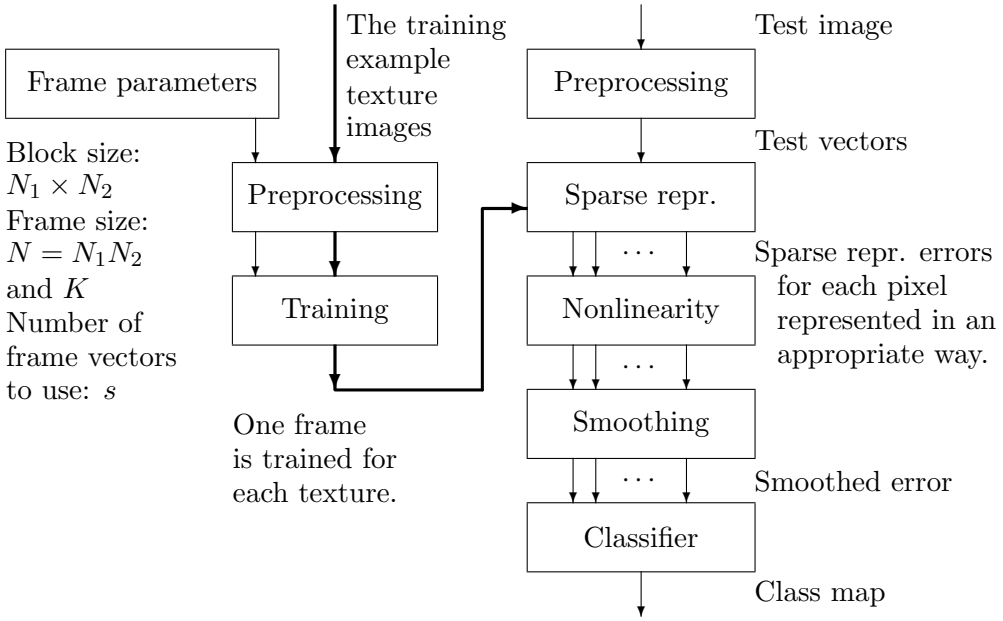


Figure 6.4: Experiment setup for the classification approach used in this chapter. Training is in the left part, and testing is in the right part of the figure.

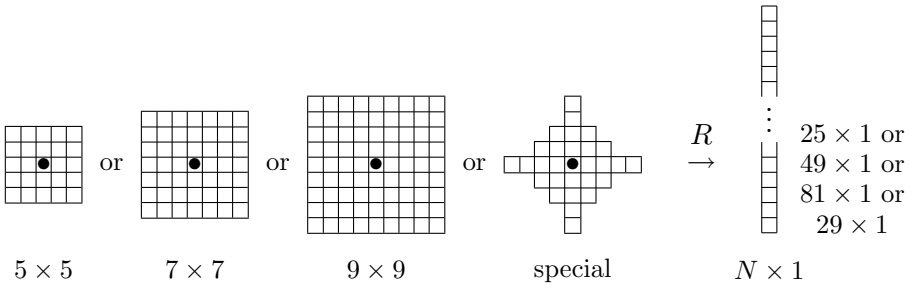


Figure 6.5: Different ways of making the training and test vectors from the image. Each small square indicate one pixel in the image. Around a central pixel, here marked with a dot, a block is made, and this block is reordered into a column vector. The figure shows four different ways of making this block around the central pixel. The most obvious ones are the square blocks, but also special kind of blocks may be used, one alternative is shown.

- The shape and size of the block made around each pixel, the properties of this block are used to classify the pixel. The block shape and size give the length of the frame vectors, N. In Figure 6.4 a rectangular block of size $N_1 \times N_2$ is indicated but, as shown in Figure 6.5, also special shapes may be used. We will use the square shapes with sizes as shown in the left three alternatives of Figure 6.5. For color textures the vectors could be made by concatenating the vectors from each of the color spaces, a 5×5 block and three colors will give a vector of length $N = 75$.

- The number of vectors in the frame, K, may be chosen quite freely. The L training vectors that represent the signal class are reduced to K frame vectors that also represent the class but in a more compact (and hopefully more general) form. To avoid overtraining[1] K should be much smaller than L. As a rule of thumb, found from the comprehensive experiments done during the work on this thesis, we may use $N \leq K \leq 5N \ll L$.

- The sparseness factor is defined as $S = s/N$ where s is the number of frame vectors that are used to approximate the signal vector. The lowercase s is the parameter used in the texture classification context, it can also be chosen quite freely. Since vector selection is more difficult, i.e. computationally demanding, the larger s is, a small value of s is usually preferred. But the main objective is to choose a value of s that provides a good discrimination of the different textures. It should be noted that the sparseness factor used during frame design does not necessarily need to be the same as the sparseness factor used during testing.

When the frame parameters are chosen the training of the frames for the different texture classes can start. Each texture class is given by one example image, and for each texture class a set of training vectors are generated. This task is denoted preprocessing in Figure 6.5, and it may be done in several different ways. The important thing is that the training vectors and the test vectors are generated in a similar way, to make the test vectors as similar to the frame vectors for the correct texture as possible. Thus the preprocessing step, which generates the vectors from an image, in the test part should be the same as the preprocessing step during training. Preprocessing may consist of one or several different steps, some of them are:

- The mean of (the pixel values of) the image may be subtracted. For an eight bit gray scale image (pixel values in range 0-255) an alternative is

[1] Overtraining: the training vectors are approximated better than other vectors belonging to the same signal class.

to subtract 128 from each pixel value. Some tests done on the effect of this step showed that the sparse representation error was reduced when the mean was subtracted, but the discrimination capabilities was usually best when this step was omitted.

- A low-pass version of the image may be subtracted, i.e. the image is high-pass filtered with a small cutoff frequency, typically the low-pass image is the (smoothed) image generated by the local mean of 16×16 or 32×32 blocks of the image. This is justified by the fact that the texture is a local property of the image. As with the step above this improved the representation quality, but (at least for the "uniform" texture examples here) it had small effect on classification.
- The center pixels of the L blocks used to make the training vectors are selected; this may be done in a random way or in a regular grid on the texture image. The blocks may be overlapping.
- Each of the training vectors may be individually scaled and/or translated, usually in such a way that the smallest value (of each training vector) is set to 0 or -1 and the largest value to 1.
- Each of the training vectors may be normalized, i.e. scaled such that each has norm one. This is often a reasonable last step of the process of preparing the training data set, and gives all the training vectors the same weight during training.

We note that the preprocessing steps can be grouped into two classes: processing done on the texture image, and processing done on each of the training vectors. For the experiments in this chapter training was done using the same 256×256 texture images as the ones used in [59][2]. For these textures preprocessing on the images, the two first steps in the list above, was not needed. The values of the pixels are in the range from 0 to 255. Square blocks were used to make the training vectors and the vectors were normalized. The effect is that they all are vectors (points) on the surface of the positive quadrant of the unit ball in $\mathbb{R}^N$. For each of the example textures the set of training vectors were collected into a $N \times L$ matrix $\mathbf{X}$.

6.2.2 Training

Training of the frame was done using the method below, the same as in Section 2.1 but described here in more detail. This is done to better illustrate

[2]The training images and the test images are available at `http://www.ux.his.no/~tranden/`.

the similarities to the Generalized Lloyd Algorithm (GLA) [28]. In a clustering context the GLA is sometimes known as the *k-means algorithm* and it is widely used and well analyzed [47] [9]. In GLA the frame is usually called the codebook, and in the clustering context the training vectors may be called feature vectors. The frame design procedure has the following steps

1. Setting the initial frame: If $s = 1$, we start by choosing K arbitrary vectors from the set of training vectors as the initial frame. If $s > 1$, the frame designed with $s = 1$ is used as the initial frame.

2. For each training vector, $\mathbf{x}_l$, we find a sparse representation, i.e. $\mathbf{w}_l$ with s non-zero values is found, typically applying an MP algorithm. If $s = 1$ this is the same as finding the frame vector closest to the training vector, i.e. nearest neighbor. The training vectors associated with a certain frame vector form a cluster. In standard GLA there is only one non-zero value in each column of $\mathbf{W}$, $\mathbf{w}_l$, and this value is 1.

3. The new frame is found by the equation
   $\mathbf{F} = \mathbf{X}\mathbf{W}^T(\mathbf{W}\mathbf{W}^T)^{-1}$.
   This $\mathbf{F}$ matrix is the frame that minimize the norm of the representation error, $\|\mathbf{X} - \mathbf{F}\mathbf{W}\|$, when $\mathbf{X}$ and $\mathbf{W}$ are fixed. If $s = 1$ this gives each column vector of $\mathbf{F}$ as the mean or centroid (weighted mean if not all non-zero values of $\mathbf{W}$ are 1) of the corresponding cluster.

4. The frame vectors are normalized, i.e. scaled such that each has norm one. The frame is then uniform. This step is not done in standard GLA.

5. Step 2 to 4 are repeated for a predefined number of times or until the method has converged, i.e. no change, or only a minimal change, in the frame since last iteration.

In step 1 we have to select the initial frame. Using a frame made by picking K of the L training vectors in a random way has, in the many experiments done, shown to work quite well in frame design. In the FTCM, for $s > 1$, convergence of the frame design method was a little bit better if the initial frame was the one that was designed for $s = 1$ rather than randomly picked training vectors. This means that the frame for $s = 1$ must be the first to be designed. We should also note the effect of step 4, and remember that the training vectors also have norm one, in this case they are in the positive quadrant of the unit ball in $\mathbb{R}^N$. Then the nearest neighbor of a training vector is the frame vector where the angle between the two vectors is minimum. That is cosine of the angle which is the inner product of the two vectors, is maximum. This is also

the frame vector that gives the best representation using only one frame vector. Negative weights are allowed for sparse representation, but for $s = 1$ this can not occur since all elements of $\mathbf{X}$ and $\mathbf{F}$ are non-negative, and this design method reduces to standard GLA with normalization of the frame vectors. For $s > 1$ the frame vectors may migrate outside of the positive quadrant during training.

6.2.3 Classification

Texture classification of a test image is the task of classifying each pixel of the test image to belong to a certain texture. To do this, a small block around each pixel to be tested must be made into a test vector. How this should be done was decided when the frames were trained, Section 6.2.1. The process for the Frame Texture Classification Method (FTCM) is illustrated in Figure 6.4, it can be compared to the process for the filter bank classification in Figure 6.1.

A test image (of size $L_1 \times L_2$) is used to generate $L = L_1 L_2$ test vectors, one for each pixel. For pixels near the edge of the image the reflection of the edge pixel is used, i.e. when $x(l)$ is defined for $l = 1, 2, \ldots, L_1$ then $x(1-l) = x(1+l)$ and $x(L_1 + l) = x(L_1 - l)$ for $l = 1, 2, 3, \ldots$. An alternative would be not to classify the pixels near the edge until the very end of the classification process, and then classify those pixels to the class of the nearest classified pixel. A test vector, $\mathbf{x}$, is represented in a sparse way using each of the different frames that were trained for the textures under consideration, $\mathbf{F}^{(i)}$ for texture class i and $i = 1, 2, \ldots, C$, C is the number of texture classes that are tested for this image. Each sparse representation gives a representation error, $\mathbf{r}^{(i)} = \mathbf{x} - \mathbf{F}^{(i)}\mathbf{w}^{(i)}$, and since $\|\mathbf{x}\| = 1$ the range of the errors will be $0 \leq \|\mathbf{r}^{(i)}\| \leq 1$. The norm squared of each error, $\mathbf{r}^{(i)T}\mathbf{r}^{(i)}$, is stored in a three dimensional matrix $\mathbf{R}$ of size $L_1 \times L_2 \times C$. This corresponds to the energy of the filter responses from the filtering step in Figure 6.1.

Direct classification based on the norm squared of the representation error for each pixel, i.e. based on the values stored in the $\mathbf{R}$ matrix, gives quite large classification errors, but the results can be substantially improved by smoothing, i.e. low-pass filtering each of the C error images (layers of $\mathbf{R}$). A nonlinearity may be applied before the low-pass smoothing filter is applied, as illustrated in Figure 6.4. For the filtering approach in Figure 6.1 the (first) nonlinearity was necessary, but here the nonlinearity should only be used if it improves the results. This may be the square root to get the magnitude of the error, or the inverse sine of the magnitude which gives the angle between the signal vector and its sparse approximation, or a logarithmic operation, then

$N \times K$	s	σ	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
49×98	3	none	23.7	49.6	64.2	67.5	59.5	58.9	77.8	64.9	78.8	60.5
49×98	3	4	3.0	18.8	23.6	26.6	22.7	27.6	38.5	33.1	42.0	26.2
49×98	3	8	4.1	13.5	12.0	12.0	11.5	19.2	21.1	22.9	25.8	15.8
49×98	3	12	6.2	11.5	10.0	10.3	10.4	17.1	19.2	20.4	21.6	14.1
49×98	3	16	8.5	11.7	10.8	11.2	12.5	17.3	20.1	18.9	21.7	14.7

Table 6.2: Classification errors, given as percentage wrongly classified pixels, for different smoothing filters (rows) and different test images (columns). Different values of σ for the Gaussian filter is presented here. The logarithmic nonlinearity is used. The frame parameters are $N = 49$, $K = 98$, and $s = 3$.

the error is represented like the common SNR measure, Equation 1.8. For smoothing Randen and Husøy concluded that the separable Gaussian low-pass filter, Equation 6.1, is the better choice, and this is also the filter used here.

The effect of filtering is illustrated in the Table 6.2 where the percentage wrongly classified pixels is shown (the experiments will be further discussed later). We see that the low-pass filtering reduce the errors considerably. The final classification is very simple in the FTCM. The frame, – remember the training resulted in one frame for each texture class, that gives the best approximation to the test vector gives the class of the test vector. This simple scheme makes the second nonlinearity, see Figure 6.1, redundant.

6.3 Classification results

The presentation of the results is divided in four parts. First, in Subsection 6.3.1, rather comprehensive experiments are done, the main purpose is to try different sets of frame parameters. This is done to get an overview of the possibilities of the FTCM. Next, in Subsection 6.3.2, the importance of the low-pass filtering and the preceding nonlinearity is further investigated. In Subsection 6.3.3 frames with different number of frame vectors are used, i.e. we let the parameter K vary and keep the other parameters fixed. Finally, in Subsection 6.3.4, the FTCM is compared to the "optimal" filtering methods proposed in [59] for the two-texture classification problem .

6.3.1 Different sets of frame parameters

The setup for the experiments is shown in Figure 6.4. In the first experiments we used 9 different sets of frame parameters. The block sizes of 5×5, 7×7

$N \times K$	s	σ	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
25×49	1	4	8.9	29.5	28.6	41.3	37.4	35.9	55.3	43.0	53.9	37.1
25×49	1	8	9.1	23.6	18.1	29.5	27.6	30.3	46.4	36.7	39.3	29.0
25×49	1	12	10.8	21.2	19.1	29.2	26.7	29.9	45.9	36.9	35.0	28.3
49×98	1	4	7.9	27.6	30.5	38.2	33.7	34.4	54.1	42.4	56.5	36.1
49×98	1	8	8.0	18.5	21.2	28.0	24.4	28.2	43.7	34.0	45.5	27.9
49×98	1	12	9.4	16.0	22.5	27.0	21.9	27.6	40.5	32.7	43.6	26.8
81×144	1	4	7.3	29.5	29.9	36.0	35.6	34.5	55.2	42.1	58.5	36.5
81×144	1	8	7.1	22.3	21.9	26.8	28.3	27.9	45.8	33.7	49.0	29.2
81×144	1	12	8.0	20.4	22.3	26.5	27.7	26.9	42.8	31.6	46.8	28.1
$N \times K$	s	σ	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
25×49	3	4	2.1	21.0	28.4	27.1	18.6	29.2	39.3	35.2	43.1	27.1
25×49	3	8	3.6	14.2	12.1	12.6	8.7	21.0	24.0	24.3	26.7	16.4
25×49	3	12	5.5	12.8	9.3	10.5	7.6	19.2	21.5	21.6	21.2	14.4
49×98	3	4	3.0	18.8	23.6	26.6	22.7	27.6	38.5	33.1	42.0	26.2
49×98	3	8	4.1	13.5	12.0	12.0	11.5	19.2	21.1	22.9	25.8	15.8
49×98	3	12	6.2	11.5	10.0	10.3	10.4	17.1	19.2	20.4	21.6	14.1
81×144	3	4	3.5	19.3	21.6	27.0	22.6	27.8	43.0	32.8	43.7	26.8
81×144	3	8	3.9	14.0	10.4	12.3	13.2	19.2	29.9	21.6	27.1	16.8
81×144	3	12	5.8	12.2	9.8	10.3	12.5	17.3	25.6	18.2	22.9	15.0
$N \times K$	s	σ	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
25×49	5	4	2.2	19.2	29.9	31.4	31.0	32.5	44.1	40.3	47.5	30.9
25×49	5	8	3.5	12.7	17.1	20.7	24.8	26.6	32.4	36.4	32.7	23.0
25×49	5	12	5.2	12.9	13.5	20.0	24.2	26.4	30.7	36.8	28.8	22.1
49×98	5	4	2.5	16.1	23.4	22.7	17.4	28.5	35.9	33.0	41.7	24.6
49×98	5	8	3.6	9.7	11.6	9.0	11.4	21.5	24.5	24.3	27.3	15.9
49×98	5	12	5.4	9.1	9.6	7.4	10.1	21.1	23.3	22.6	23.7	14.7
81×144	5	4	3.4	17.7	21.9	23.2	18.7	26.4	37.2	31.1	38.7	24.2
81×144	5	8	3.8	11.8	11.1	9.3	10.6	18.1	24.0	20.6	23.6	14.8
81×144	5	12	5.7	11.2	10.0	7.2	10.9	15.6	22.8	18.1	20.0	13.5

Table 6.3: Classification errors, given as percentage wrongly classified pixels, for different sets of frame parameters and smoothing filters (rows) and different test images (columns). The nonlinearity before smoothing was logarithmic (SNR). The first three columns are the frame parameters and σ used in the Gaussian smoothing filter. The next nine columns show the results for the test images, Figures 6.2 and 6.3. The last column is the mean for these test images.

and 9×9 were used, the three leftmost examples in Figure 6.5. The number of frame vectors in each frame was first selected so that K is approximately two times the value of N or a little bit smaller, $K = 49$ for $N = 25$, $K = 98$ for $N = 49$ and $K = 144$ for $N = 81$. Each of these frame sizes were combined with the values $s = 1$, $s = 3$ and $s = 5$, where s is the number of frame vectors to use in the sparse representation. For each parameter set a frame was designed for all the textures of interest. Note that the training vectors were made from separate example images of each texture, not from blocks of the test images. The number of different textures used in the 9 test images, Figure 6.2 and Figure 6.3, is 77, and the number of different parameter sets is 9, thus it was necessary to design 693 frames in this experiment. The design of all the frames needed many weeks of computer time, in average one hour for each frame, but this task must only be done once.

We tested these nine sets of frame parameters on all of the test images in Figure 6.2 and Figure 6.3. Test vectors were made also for pixels near the edge. The nonlinearity was logarithmic (SNR) and Gaussian low-pass filters were used for smoothing, the bandwidths used were $\sigma = 4$, $\sigma = 8$ and $\sigma = 12$.

The classification results are presented in Table 6.3. Let us start by looking at the case where $s = 1$. This is the same as vector quantizing classification, or nearest neighbor classification. These results are quite similar to the best results of the filtering methods in Table 6.1. The mean for the method "f16b" was 25.9 percent wrongly classified pixels, while the parameter setup 49×98 for $N \times K$ and $\sigma = 12$ gave 26.8 percent wrongly classified pixels. Even though the means are comparable, the results for the individual test images may be more different. For the test image (h) the result is 39.8 for the "f16b" filtering method, and 32.7 for the FTCM with parameters 49×98 and $\sigma = 12$, while for the test image (i) the results are 28.5 and 43.6 respectively. Generally, we note that the different filtering methods and the autoregressive method perform better on (i) than on (h), and that the co-occurrence method and the FTCM with $s = 1$ (VQ-method) perform better on (h) than on (i).

For the tests with the FTCM and $s > 1$ much better results are achieved. The number of wrongly classified pixels is almost divided by two, and this is very good. We do not have a good explanation for why the sparse representation model should be so much better than the VQ-method, but the many experiments are quite conclusive. One interesting thing is noteworthy: The number of vectors used in the representation, s, should be increased when the parameter N is increased. For $N = 25$ the frames where $s = 3$ perform better than the frames where $s = 5$, for $N = 49$ the two different values of s perform almost equal, and for $N = 81$ the frames where $s = 5$ is better that the frames where $s = 3$. This observation can be explained by the fact that

when N is larger the number of vectors to select must be larger to have the same sparseness factor, – remember $S = s/N$, or to have a reasonable good representation of the test vector. Best results were found using three or five frame vectors to represent each test vector, this indicate that "the optimal" number of vectors to use, s, probably is in the N-range explored: for $N = 25$ the best s may be 3 or 4, for $N = 49$ the best s may be 4 and for $N = 81$ the best s may be 5 or 6.

The block size should probably be larger for coarse textures than what is necessary for the more fine textures. The case where $N = 25$ is best for the fine textures in the test image (a), while the cases with larger values for N perform better for the test images (c) and (d) which also contain regions with coarser textures.

6.3.2 The nonlinearity and low-pass filtering

In Figure 6.6 the results of different nonlinearities and different values of σ in the Gaussian filter are shown. The results are not clear, but some conclusions seem likely. It seems best to choose the logarithmic nonlinearity as the preferred one. For many of the test images the choice of nonlinearity seems to be unimportant, but for three of the test images, (b), (e), and (f), the nonlinearity matters. For these three test images the logarithmic nonlinearity is slightly better than the magnitude nonlinearity and the energy nonlinearity is clearly the worst choice.

When it comes to the choice of σ in the Gaussian filter it seems like the values in the range from 10 to 14 generally perform best. One exception is for the fine textures in test image (a), here the number of wrongly classified pixels is low. Too much smoothing cause more errors to occur near the edges between different textures, this smoothing effect is illustrated in Figure 6.7. When $\sigma = 4$ smoothing on the test image (a) has correctly classified almost all pixels, the wrongly classified pixels are near the image edges and the borders between regions of different texture. Increasing σ cause only more smoothing and an increased probability of errors near the border between different texture regions, but the errors near the edges are corrected. This effect on the region borders also exists for test image (c) in Figure 6.8 and test image (g) in Figure 6.9, but here the effect is compensated for by the much better performance within the interior of the texture regions. The conclusion can be stated as: Generally a large value of σ is best, for example $\sigma = 12$, but if the error rate is small or if the texture regions are relatively small a smaller value of σ should be used.

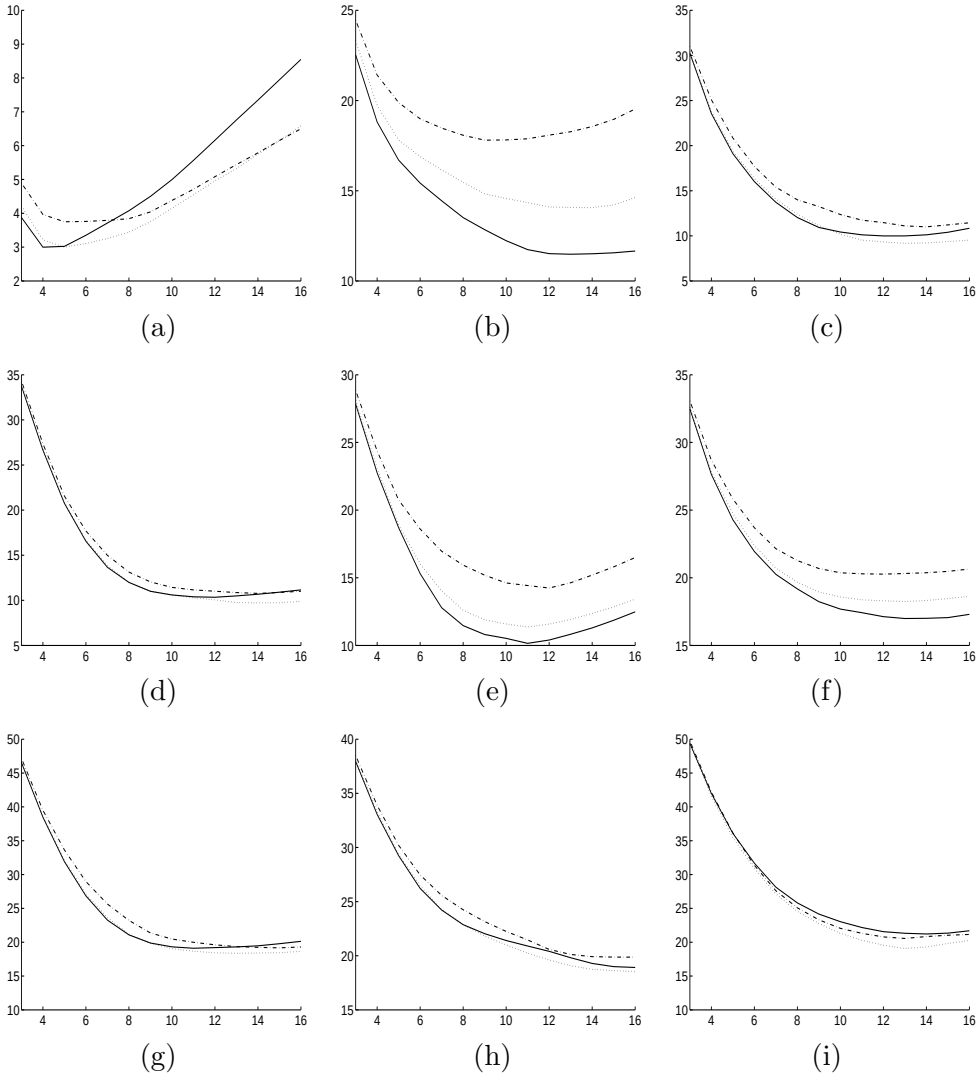


Figure 6.6: The percentage wrongly classified pixels along the y-axis for the nine test images, note that the scale varies. Along the x-axis is σ used in the Gaussian low-pass filtering. The nonlinearity are: logarithmic as solid lines, magnitude as dotted lines and energy (the output after sparse representation) as dash-dot lines. The frame parameters are $N = 49$, $K = 98$, and $s = 3$.

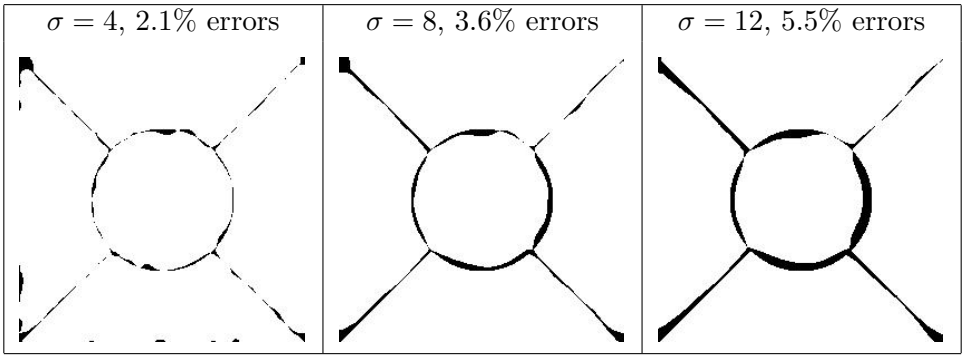


Figure 6.7: The wrongly classified pixels for the test image (a). The logarithmic nonlinearity is used. The frame parameters are $N = 25$, $K = 49$, and $s = 3$.

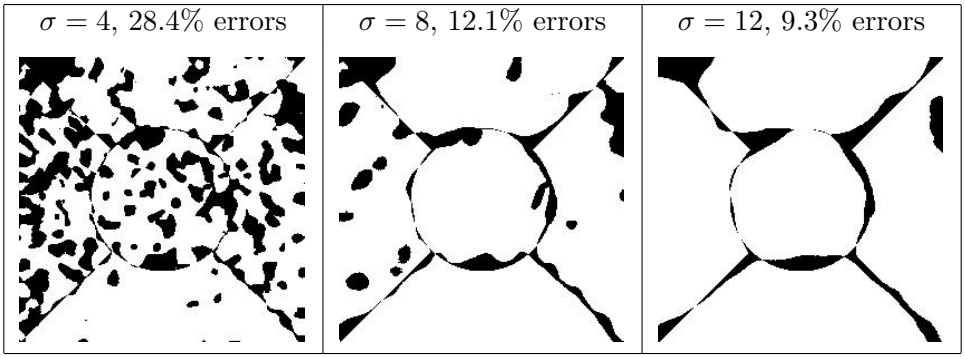


Figure 6.8: The wrongly classified pixels for the test image (c). The logarithmic nonlinearity is used. The frame parameters are $N = 25$, $K = 49$, and $s = 3$.

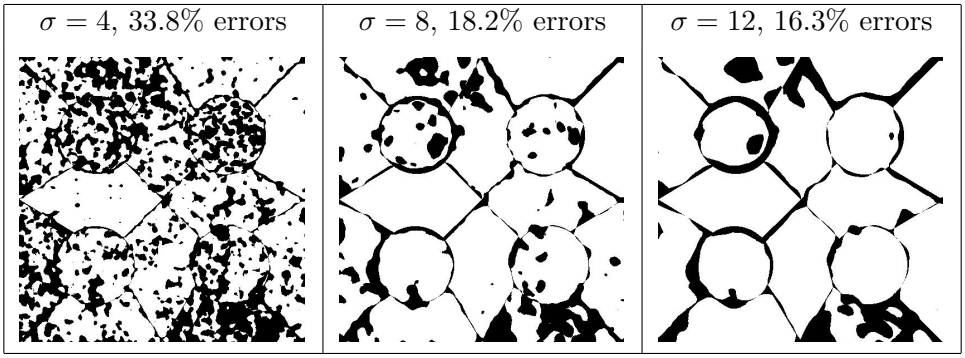


Figure 6.9: The wrongly classified pixels for the test image (g). The logarithmic nonlinearity is used. The frame parameters are $N = 25$, $K = 100$, and $s = 3$.

$N \times K$	s	σ	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
25×25	1	12	10.8	29.4	15.4	33.9	23.8	30.1	49.3	34.8	40.2	29.7
25×49	1	12	10.8	21.2	19.1	29.2	26.7	29.9	45.9	36.9	35.0	28.3
25×100	1	12	11.7	16.3	21.1	26.2	25.5	28.7	45.8	35.4	34.4	27.2

Table 6.4: Classification errors, given as percentage wrongly classified pixels, for different choices for the K parameter and different test images. The other frame parameters are $N = 25$ are $s = 1$, the nonlinearity is logarithmic (SNR) and $\sigma = 12$ in the Gaussian filter.

6.3.3 Choosing the frame size K

To test the effect of varying the number of vectors in the frame, the parameter denoted K, more frames were designed. For the block size 5×5, giving $N = 25$, frames were designed using $K = 25$, $K = 49$ and $K = 100$ vectors to best represent each of the textures. This was done for both $s = 1$ and $s = 3$. These frames were used in tests using the same test images as before.

Let us start by looking at the case where $s = 1$. This is the same as vector quantizing (VQ) classification, or nearest neighbor classification. It would be interesting to see if VQ classification methods are able to discriminate textures better as K increase. It is obvious that the representation capabilities of the frame (codebook in VQ context) gets better when K is increased, but this does not need to give better classification. In Table 6.4 we see that on the average only a small improvement in classification is achieved as K increase, but we also note that there are large differences for the test images. For test image (b) the classification result is much better for $K = 100$ than for $K = 25$, while for test image (c) the classification error is larger for $K = 100$ than for $K = 25$. The latter illustrate that better representation of the different textures does not necessarily give better discrimination between the textures. Another conclusion we may make is that the FTCM with $s > 1$ performs better than vector quantizing classification (or FTCM with $s = 1$) even if we let K be larger for the case where $s = 1$.

In Table 6.5 the classification result for $s = 3$ is shown. Here we have included results for four different low-pass Gaussian filters, but the best results in the average are achieved for $\sigma = 12$ for the different values of K. The results are generally much better than for $s = 1$. On the average a small improvement in classification is achieved as K increase, but as for the case when $s = 1$ there is large differences for the test images. For test image (b) the classification result is much better for $K = 100$ than for $K = 25$, while for test image (c) the best results are almost equal for the three values of K but best for $K = 49$. For

$N \times K$	s	σ	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	Mean
25×25	3	4	2.7	22.7	33.3	34.8	22.8	32.4	45.8	36.7	47.7	31.0
25×25	3	8	3.5	19.0	17.3	15.1	14.0	24.2	28.5	25.5	29.6	19.7
25×25	3	12	5.3	19.2	11.7	11.0	12.3	22.8	22.8	22.0	23.7	16.8
25×25	3	16	6.9	21.1	9.5	12.8	13.4	22.6	22.2	21.6	23.1	17.0
25×49	3	4	2.1	21.0	28.4	27.1	18.6	29.2	39.3	35.2	43.1	27.1
25×49	3	8	3.6	14.2	12.1	12.6	8.7	21.0	24.0	24.3	26.7	16.4
25×49	3	12	5.5	12.8	9.3	10.5	7.6	19.2	21.5	21.6	21.2	14.4
25×49	3	16	7.7	13.1	10.1	12.3	9.6	19.1	21.9	20.9	20.8	15.1
25×100	3	4	2.3	13.8	22.7	25.8	20.2	28.7	33.8	33.2	39.3	24.4
25×100	3	8	3.9	7.6	10.1	12.2	11.8	21.8	18.2	21.8	23.9	14.6
25×100	3	12	5.9	6.5	9.8	8.9	10.8	21.1	16.3	18.9	20.1	13.2
25×100	3	16	8.3	6.8	12.0	10.9	11.9	21.7	17.0	18.6	18.6	14.0

Table 6.5: Classification errors, given as percentage wrongly classified pixels, for different choices for the K parameter and different test images. The other frame parameters are $N = 25$ are $s = 3$, the nonlinearity is logarithmic (SNR) and $\sigma = 4, 8, 12, 16$ in the Gaussian filter.

test image (e) the result is better for $K = 49$ than for the other two cases. An important observation that can be made from Table 6.5 and Table 6.3 is that it is usually better to increase K than to increase N. The frame of size 25×100 is only about half the size of the 49×98 frame and considerable smaller than the large 81×144 frame, but still it is the frame that performs best on the average.

One important question is: What is the best set of frame parameters to use? The experiments done so far tell us that this is not an easy question to answer. The answer depends on the texture classification task at hand. For the mean of the test images used the best results were achieved for case where 5×5 ($N = 25$), $K = 100$ and $s = 3$, so if any combination should be especially recommended this is it.

6.3.4 Two-texture test image

It is also interesting to test the FTCM on two texture test images. Randen and Husøy designed filters that were optimized to discriminate between two different textures, [60]. It is interesting to see how the FTCM performs compared to these filters. They tried three different object function: J_{MS} as used by Mahalanobis and Singh [43] Equation 6.3, J_U originally suggested by Unser [69] Equation 6.4, and J_F [27] suggested by Fisher and defined as

$$J_F = \frac{(\mu_{v_1} - \mu_{v_1})^2}{\sigma_{v_1}^2 + \sigma_{v_2}^2}. \tag{6.5}$$

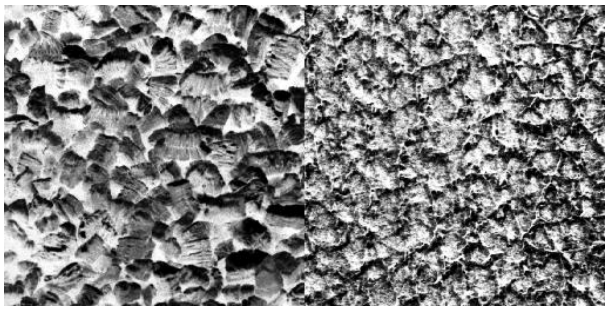


Figure 6.10: The 2-textures test image "D5D92".

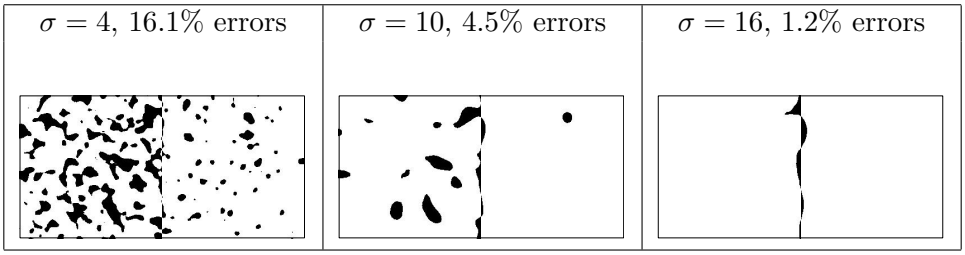


Figure 6.11: The wrongly classified pixels for the test image "D5D92" using FTCM. The logarithmic nonlinearity is used. The frame parameters are $N = 25$, $K = 100$, and $s = 3$.

σ	4	6	8	10	12	14	16
% errors	16.1	10.6	6.8	4.5	3.0	2.0	1.2
Method	J_{MS}	J_U	J_F	DCT	f16b(d)	f8a(d)	Daub4(d)
% errors	28.6	5.1	5.1	2.5	8.2	7.2	8.2
Method	co-occurrence	AR	eigenfilter		f16b(c)	f8a(c)	Daub4(c)
% errors	3.3	3.0	4.7		4.1	3.9	6.5

Table 6.6: The FTCM results for the test image "D5D92" with different choices for the σ parameter in the first two lines. The frame parameters are $N = 25$, $K = 100$ and $s = 3$. The last lines are results of different methods from the work of Randen and Husøy.

where $\sigma_{v_i}^2$ is the feature variance.

Here, we only use one test image, denoted "D5D92". It is shown in Figure 6.10 and consists of textures D5 and D92 from the Brodatz album [10]. Frames were designed for these textures and the classification test was done as described in this chapter. The classification results using the FTCM with different values of σ, and the classification results for some of the methods used in [59] including the three optimized filters J_{MS}, J_U, and J_F, are shown in Table 6.6. The methods denoted "f8a" and "f16b" use a tree structured bank of quadrature mirror filters (QMF), the filters are finite input response (FIR) filters of length 8 and 16, respectively. The method denoted "Daub-4" use the Daubechies filters [17] of length 4. Two different structures are referred, the filters marked (d) use a non-dyadic filter bank structure, while the ones marked (c) use a dyadic decomposition. The table also refers the results of the "DCT method", using the coefficients from a 3×3 discrete cosine transform as features, the "eigenfilter" method where the filters are derived from the eigenvalues of the autocorrelation functions of the textures, and the methods based on co-occurrence (statistical) and autoregressive (model-based) features. For more details on these methods we again refer to Randen and Husøy [59].

Also for the two-texture test image, as can be seen from Figure 6.11 and Table 6.6, the FTCM performs quite well. The smoothing filter used for the methods J_{MS}, J_U, and J_F had the parameter $\sigma = 8$. For FTCM, using the same value of σ, the results are comparable, but increasing σ improves classification considerable. Since smoothing has the same effect, as shown in Figure 6.11, for the filtering method as for FTCM, it is reasonable to assume that increasing σ will also improve classification for these methods. For this particular test image we note that using the J_{MS} criterium fails compared to the alternative criteria J_U and J_F, but better results were obtained in [59] using the methods "DCT", "co-occurrence", and "AR". The FTCM obtains better classification results only when the value of σ is allowed to grow, best for $\sigma = 16$ where the percentage wrongly classified pixels is as low as 1.2%.

6.4 Some comments

Comparing the results in Table 6.5, the lines with parameters $N \times K = 25 \times 100$, $s = 3$, and $\sigma = 8$ (perhaps the one most comparable to the values of σ used in [59]) or $\sigma = 12$ (which gives a little bit better results), to the results in Table 6.1, the conclusion is that FTCM seems like a very good method

for texture classification. It significantly reduces, compared to the methods tried in [59], the number of wrongly classified pixels for all the test images containing several textures. For the two-texture classification problem the results, Table 6.6, are not conclusive.

From the results in Table 6.3 we noted that FTCM and $s > 1$ gave much better results than FTCM and $s = 1$ (VQ classification). The two methods are both model-based methods where a signal block is approximated. At present we lack a full theoretic explanation why the models using $s > 1$ perform that much better than the models using $s = 1$ in classification. One explanation could be that the signal model should be able to capture most of the essential properties of the texture in the approximation, but at the same time exclude much of the other properties. In this context a property is represented by a frame vector, and a linear combination of some few of them makes up an approximation. The problem is of course that the different textures seem to have a lot of common properties, implying that catching only the most conspicuous properties will not make good discrimination possible. Catching too much of the signal energy will also make it difficult to discriminate, if $s = N$ there will be no representation error and no signal class discrimination is possible.

It is obvious that the frames for different textures must hold different properties to be able to discriminate. Which properties that are important in the texture classification context are yet not known. In Chapter 7 we consider many possible frame properties, but clear connections from these properties to the "being able to discriminate textures"-property are yet not found. Nevertheless, we hope that the work in Chapter 7 may be helpful in future work on these issues. A theoretic understanding of the FTCM will with no doubt be helpful when selecting the appropriate frame parameter set for a texture classification problem at hand.

An alternative to theoretic understanding is an understanding based on experience. This latter can be increased by doing more experiments: trying more parameter sets, training frames for more textures, and testing these on more test images and also different kinds of test images. Often the interaction between theoretical knowledge and practical knowledge will increase both, theoretical knowledge will help to define relevant experiments, and practical knowledge may give useful hints in the theory development.

Chapter 7

Some Considerations on Frame Properties

The objective of this chapter is to define some properties of the frame that measure the sparse representation capabilities of the frame, and relevant and practical ways to calculate these properties. We also hoped that the frame properties would help us to get a better understanding of how frames are able to discriminate between different textures, as shown in Chapter 6. For this latter purpose we have not yet found any obvious connection between the proposed frame properties and the texture discrimination capabilities, but there is still much work to in this area, we hope that the results presented here may be helpful in that work.

The proposed properties must be seen in the context of frame properties that are used in the frame theory before, and to do this some of the frame theory must be referred. This is addressed in Section 7.1. The limited class of frames that we consider is described in the last part of this section. In Section 7.2 we discuss some possible frame properties.

The mathematical details of the proposed properties are further discussed in Section 7.3 where the first subsections consider block-oriented frames. The singular value decomposition is discussed in Subsection 7.3.1. As for matrices in general, the singular values of the frames are important properties also for frames. The frame bounds are directly connected to these. In Subsection 7.3.2 we define some properties that are directly connected to the representation error, even though these properties measure the sparse representation capabilities of the frame directly they are not very practical because they are difficult to calculate, and they are also tied to a specific signal class. Properties based

on the angles between frame vectors do more indirectly measure the sparse representation capabilities of the frame, and they are discussed in Subsection 7.3.3. The norm of the weights is discussed in Subsection 7.3.4. Angles between frame vectors can also be a good way of measuring the difference between two frames, as shown in Subsection 7.3.5.

Finally properties of overlapping frames are discussed in Subsection 7.4. Some examples for the use of these properties are shown in the end of this chapter, Section 7.5.

7.1 Definitions and theory

The frame concept was first introduced in the early fifties by Duffin and Schaeffer [20], where much of the general theory was laid out. In the late eighties frames received renewed attention, mainly as a consequence of identified connections with the wavelet transform and time-frequency analysis [34] [16]. Since then much work has been done on the frame theory, especially on Gabor frames [25], i.e. frames made by translations and modulations of a single function, and Wavelet frames, i.e. frames made by translations and dilations of a single function. The oversampled filter bank, in this thesis usually called overlapping frame, is a frame if perfect reconstruction is possible [15] [8]. Pei and Yeh [57] present the special case of discrete finite frames. This class of frames includes the frames used for sparse signal representations. A tutorial on the art of frame theory was written by Casazza [11].

7.1.1 Bases and Frames

Bases and frames are defined in a separable Hilbert space, denoted H. A *basis* is a set of elements $\{e_n\}$ such that any element of H can be uniquely expressed as a linear combination of the basis elements. Since many important Hilbert spaces, for example the common function space $\mathbf{L}^2(\mathbb{R})$, are infinite dimensional, the basis can have an infinite (but countable) number of elements. In fact, the number of basis elements is equal to the dimensionality of the space. Infinite dimensional spaces make it necessary to use precise definitions for the concepts used, and depending on the definitions different bases can be defined. For a finite dimensional space this is much simpler, a basis is a set of linearly independent elements that span the space. In this thesis we mainly use the space $\mathbb{R}^N$, and the elaborate and precise mathematical definitions are not necessary, a basis is any set of N vectors that span the space. We should note that in this thesis the N-dimensional space is used to represent

N consecutive samples of a one-dimensional signal. An *orthonormal* basis is a basis where the elements are normalized and orthogonal to each other, i.e. the inner product of two different elements is zero and the inner product of an element with itself is one; $< e_n, e_m >= \delta_{n,m}$ where $< a, b >$ is the inner product of a and b. In $\mathbb{R}^N$ the basis vectors are often collected as columns in an invertible matrix of size $N \times N$.

Frames are a generalization of bases. Let us start with the formal definition of a frame. A family of elements $\{f_i\} \subseteq H$ is called a frame for the separable Hilbert space H if there exists constants $A, B > 0$ such that

$$A\|x\|^2 \leq \sum_i | < x, f_i > |^2 \leq B\|x\|^2, \qquad \text{for all } x \in H. \tag{7.1}$$

The indices i may be elements in any countable index set. The numbers A, B are called *frame bounds*. They are not unique, i.e. if A and B are frame bounds that satisfy Equation 7.1 then so do A/c and Bc where $c > 1$. The largest possible value for A and the smallest possible value for B are called the *optimal frame bounds* and these are the frame bounds most commonly used. In the rest of this chapter the optimal frame bounds will often only be referred to as the frame bounds. If we can choose $A = B$ the frame is called *tight*. The definition in Equation 7.1 implies that the set of elements $\{f_i\}$ must span the space H, if not we will get $A = 0$ since $< x, f_i >= 0$ when $x \in (H \setminus \text{span } \{f_i\})$. The frame is called *exact* if the frame ceases to be a frame of H when any element is removed.

An operator, S, is called the *frame operator* of the frame, it is defined by $Sx = \sum_i < x, f_i > f_i$, for all x in the space spanned by the frame elements. It is a bounded linear operator and it is invertible [34]. The *dual frame* of $\{f_i\}$ is defined as $S^{-1}\{f_i\}$ [34], it is used in the Method of Frames, MOF in Section 1.3, to find the weights in a signal expansion, the frame is used in the synthesis part and the dual frame in the analysis part of the signal expansion. The optimal (tightest possible) frame bounds A and B are given by the essential infimum and supremum, respectively, of the eigenvalues of the frame operator [34], [16]. The frame bounds for the dual frame are $1/B$ and $1/A$.

In $\mathbb{R}^N$ a frame can simply be defined as any set of K vectors that span the space[1]. It is represented by an $N \times K$ matrix $\mathbf{F}$ where $N \leq K$. The equation corresponding to Equation 7.1 is

[1] If the vectors do not span $\mathbb{R}^N$ they still form a frame, but then they form a frame in the subspace of $\mathbb{R}^N$ spanned by the frame vectors.

$$A\|\mathbf{x}\|^2 \leq \|\mathbf{F}^T\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{F}\mathbf{F}^T\mathbf{x} \leq B\|\mathbf{x}\|^2, \qquad \text{for all } \mathbf{x} \in \mathbb{R}^N. \tag{7.2}$$

The frame operator is represented as a matrix of size $N \times N$, $\mathbf{S} = \mathbf{F}\mathbf{F}^T$ [57]. The dual frame is $\tilde{\mathbf{F}} = \mathbf{S}^{-1}\mathbf{F} = (\mathbf{F}\mathbf{F}^T)^{-1}\mathbf{F}$, it is the transposed of the pseudoinverse of $\mathbf{F}$, $\mathbf{F}^\dagger = \mathbf{F}^T(\mathbf{F}\mathbf{F}^T)^{-1}$. The frame bounds are given as the smallest and largest eigenvalues of the frame operator matrix $\mathbf{S}$ [57].

7.1.2 Oversampled Filter Banks

The oversampled filter bank has been analyzed by frame-theoretic methods by Cvetković and Vetterli [15], Bölcskei et al. [8] and Stanhill [66]. The overlapping frame introduced in Section 1.1 is an oversampled filter bank.

A common way to represent a filter bank is to use the polyphase matrix, for example see Section 5.5 in [71]. The polyphase representation is also used in [8] and [15] for the oversampled filter bank. The matrix notation used in this work is closely connected to the polyphase notation. The overlapping frame in Equation 2.10 (expanded like in Equation 1.10 and $\mathbf{F}$ as in Equation 2.11) has the synthesis polyphase matrix

$$\mathbf{R}(z) = \sum_{p=0}^{P-1} \mathbf{F}_p z^{-p}. \tag{7.3}$$

The frame operator can also be represented as a polynomial matrix [8]

$$\mathbf{S}(z) = \mathbf{R}(z)\mathbf{R}^H(z) = \sum_{p=0}^{P-1}\sum_{i=0}^{P-1} \mathbf{F}_p\mathbf{F}_i^T z^{i-p}, \tag{7.4}$$

where $\mathbf{R}^H(z)$ is the complex conjungated and transposed of $\mathbf{R}(z)$. From the work of Bölcskei et al. [8], the frame bounds A and B are given by the essential infimum and supremum, respectively, of the eigenvalues $\lambda_n(\theta)$ of the frame operator matrix for z-values on the unit circle, i.e. $z = e^{j2\pi\theta}$ for $0 \leq \theta < 1$, $j = \sqrt{-1}$. Note that in [8] the frame is the analysis filter bank and the dual frame is the synthesis filter bank.

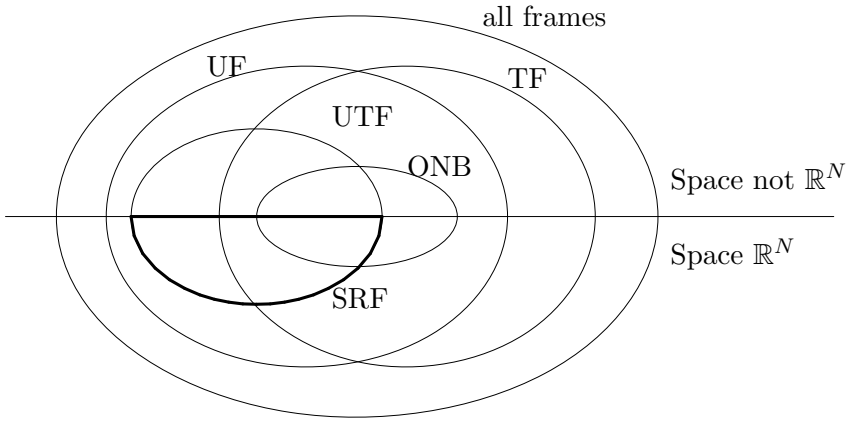


Figure 7.1: Different frame classes. The large ellipse represents all frames. Within this two ellipses are drawn, the left represents all uniform frames (UF), and the right all tight frames (TF). The intersection of these is the class of uniform tight frames (UTF), within this class the orthonormal bases (ONB) are represented by a small ellipse. The frames for sparse representation (SRF) are represented by an ellipse within the UF class. The horizontal line divides all these classes of frames into two, one part for frames in space $\mathbb{R}^N$ and one for the rest (infinite dimensional spaces and complex spaces). The frames of our interest are the frames for sparse representation in $\mathbb{R}^N$, this class is indicated by thick lines in the figure.

7.1.3 Frames for signal representation

The definition of a frame in Equation 7.2 is quite general, in fact any full rank matrix of size $N \times K, N \leq K$, represents a frame. We may put some restrictions onto the frame without restricting its signal representation properties.

1. In signal representation the length of the frame vectors does not matter, the value of the weight can compensate for this. We prefer to use uniform frames, for which the frame vectors have 2-norm one, $\|\mathbf{f}_k\| = 1$. The Frobenius/trace norm of the frame is then $\|\mathbf{F}\| = \sqrt{K}$ since $\|\mathbf{F}\|^2 = \sum_{k=1}^{K} \|\mathbf{f}_k\|^2 = K$.
2. The sign of a frame vector can also be compensated for by the weight. We select the sign such that $\sum_n f_k(n) \geq 0$, and if this sum is zero the sign is set so that the first non-zero element is positive.

3. The frame should contain no identical frame vectors, i.e. $\mathbf{f}_i \neq \mathbf{f}_j$ when $i \neq j$.

4. Permutations: The frame vectors may be in any order in the matrix $\mathbf{F}$. The frame $\mathbf{FP}$ where $\mathbf{P}$ is a $K \times K$ permutation matrix has the same representation capabilities as the frame $\mathbf{F}$. We do not have a preferred order of the frame vectors.

The different classes of frames are illustrated if Figure 7.1. The class of frames for signal representation (SRF) is a subclass of uniform frames (UF), restriction 1 above. The class is further reduced by point 2 and 3, and the fact that we here only consider the space $\mathbb{R}^N$. Note that point 2 cuts out a part of the orthogonal bases from SRF. Since we do not have any preferred order of the frame vectors (point 4), different permutations of the frame vectors are regarded as different frames, but their representation properties are the exactly the same. If a preferred order of the frame vectors were defined, the class of frames under consideration would be even smaller, but Figure 7.1 could be the same.

7.2 Alternative frame properties

What we want to do is to find some frame properties that can help us to reveal the sparse representation capabilities of the frame. The representation must clearly be assessed in the context of the signal or signal class that the frame is designed for. But also the general representation properties of the frame, not connected to any particular signal class, reveal some interesting information. The frame bounds are the most important frame properties for this purpose. They are well known in frame theory, and also useful for our purpose and they will be discussed in Subsection 7.3.1.

The fact that our main interest is in *sparse* representations, makes us look for other, and hopefully more informative, frame properties. The ultimate sparse representation is when only one vector is used to approximate the signal vector, and this case is equivalent to a shape gain vector quantizer where only the shape is quantized. It may also be regarded as a general vector quantizer (VQ) of the normalized (and possible change of sign as in point 2 in Subsection 7.1.3) signal vector. For the VQ case the frame is usually called the codebook. In a VQ context the overall performance of a codebook is often assessed using a suitable distortion measure. If the signal class is given by the N-dimensional probability density function (pdf) for the length N signal

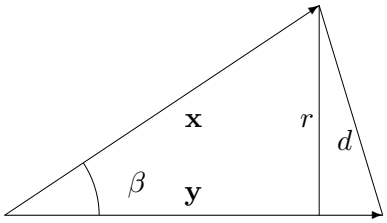


Figure 7.2: Figure illustrating different possible measures for distance between two vectors. The vectors $\mathbf{x}$ and $\mathbf{y}$ are normalized to unit length and $\beta \leq \pi/2$. Some possible measures are the angle, β, the distance between endpoints, d, the distance between one vector and its projection onto the other vector (the representation error), r, or this error squared, r^2.

vectors, the distortion for a codebook can be calculated, i.e. the pdf for the distortion can be calculated. If the signal class is given by many representative vectors the pdf for the distortion can be estimated. The pdf for the distortion is a (continuous) function and properties derived from this are more practical to assess the codebook, these derived properties can be the statistical average or the worst-case distortion (the highest value of the distortion for which the pdf is larger than zero).

The distortion between a vector $\mathbf{x}$ and its representation $\mathbf{y}$ can be measured in several ways, as illustrated in Figure 7.2, and listed in the following table

Measure	Definition
β	$\cos\beta = \mathbf{y}^T\mathbf{x}$
d	$d = \|\mathbf{x} - \mathbf{y}\| = \sqrt{2(1-\cos\beta)}$
r	$r = \|\mathbf{x} - \cos\beta\mathbf{y}\| = \sin\beta$
r^2	$r^2 = \sin^2\beta = 1 - (\mathbf{y}^T\mathbf{x})^2$

We should note that in the figure and the table both the signal $\mathbf{x}$ and its representation $\mathbf{y}$ are assumed to be normalized to unit length. If the signal is not normalized the distortion relative to the signal length will often be preferred measures, i.e. $d/\|\mathbf{x}\|$ or $r/\|\mathbf{x}\|$. For a frame, or shape-gain VQ codebook, the frame vectors are normalized (in Figure 7.2 $\mathbf{y}$ would be the frame vector closest to the signal vector), but in this case the representation (assuming no quantization of the weight or gain) will be the $\mathbf{x}$ projected onto $\mathbf{y}$, $\tilde{\mathbf{x}} = (\mathbf{y}^T\mathbf{x})\mathbf{y}$. In this case the distortion measure d is not that relevant. Note

that both r and d are scalars representing the norm of the vectors $\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}$ and $\mathbf{d} = \mathbf{x} - \mathbf{y}$ respectively.

The sparse representation performance of a frame could be assessed in a way similar to how a codebook is assessed in a VQ context. The properties used could be the ones derived from the estimated pdf for the chosen distortion measure (excluding d) in the table above, i.e. the average or the maximum. Using a frame for sparse representation, we are not restricted to use only one of the frame vectors, a linear combination of s vectors may be used to approximate a given signal vector. The same properties can be used also for this case. For a frame whit $s > 1$ these properties also depend on the vector selection algorithm used, but to not complicate too much we assume, at least for small values of s, that the optimal full search vector selection algorithm should be used. And, as mentioned above, these properties of course depend on the signal class, i.e. the pdf of the signal.

Totally, this gives a lot of different properties of a frame which are all connected to the sparse representation capabilities. We choose some few of these: The most relevant distortion measure is perhaps the representation error, $r = \|\mathbf{r}\| = \|\mathbf{x} - \tilde{\mathbf{x}}\|$. The derived properties may then be the average, $r_s^{(avg)} = \frac{1}{L}\sum_{l=1}^{L} r_l$, and the maximum, $r_s^{(max)} \max_l r_l$. The subscript s is the value of s used in the sparse representation. Since the objective when designing the frames is to minimize the sum of errors squared a property based on the r^2 measure is also included, we let $r_s^{(mse)}$ be the square root of the average of the errors squared, i.e. the mean is taken for the squared errors. This makes $r_s^{(mse)}$ approximately the same size as $r_s^{(avg)}$ (actually $r_s^{(avg)} \leq r_s^{(mse)}$). If the signal class is not explicitly given for the properties based on r, $r_s^{(max)}$, $r_s^{(avg)}$, and $r_s^{(mse)}$, a random white Gaussian signal should be assumed. All directions in the space $\mathbb{R}^N$ will then have the same probability and the signal distribution is uniform on the unit ball.

An obvious measure for the representation capabilities for the frame is the SNR, Equation 1.9, achieved during training, denoted SNR_t in Table 7.1. During training of the frame the sum of squared errors is wanted as small as possible, Equation 2.3, this imply that the SNR is maximized. Assuming some strict conditions on the set of training vectors, which are hardly ever met, the SNR_t property and the $r_s^{(mse)}$ property will measure the same property but with different units. If we use the set of training vectors to represent the signal class, and this set consists of normalized vectors and the mean is the zero vector, and if optimal vector selection was done during training, then we would have $SNR_t = -20\log_{10} r_s^{(mse)}[dB]$.

A	Lower frame bound.
B	Upper frame bound.
A_s	The frame property such that the norm of the weights for an optimal sparse representation (selecting s frame vectors) will be limited by $\|\mathbf{w}\| \leq (1/\sqrt{A_s})\|\mathbf{x}\|$.
SNR_t	The SNR, Equation 1.9, achieved during training.
	Some measures based on the relative representation error, $r = \|\mathbf{r}\|/\|\mathbf{x}\|$.
$r_s^{(max)}$	The maximal error, r.
$r_s^{(avg)}$	The mean (average) of r.
$r_s^{(mse)}$	The square root of the mean square error.
	Some measures based on the angle between frame vectors.
$\beta^{(min)}$	The angle between the two frame vectors closest to each other.
$\beta^{(avg)}$	The average for each of the frame vectors of the angle to its closest neighbor.
$\beta^{(mse)}$	The average of all the angles between frame vectors taken in the "mean square sense".
$\beta^{(avg2)}$	The average for each of the frame vectors of the angle to the center of the cluster formed by the frame vectors.
	Measures for difference between two frames.
$\theta^{(avg)}$	The average for the angles for all the frame vectors to its closest neighbor in the other frame.

Table 7.1: A summary of different frame properties. The subscript s tells the number of frame vectors that are used in the signal representation. The properties are further explained in the text.

For a codebook used for VQ there is no point in having two codebook vectors identical, or almost identical. For a good codebook the vectors are often distributed in a way similar to the distribution of the signal. Here we propose some frame properties that give some information about how the normalized frame vectors are distributed or clustered on the surface of the unit ball in $\mathbb{R}^N$. The distance measure used below could be any one of the ones illustrated in Figure 7.2, but the preferred one here is the angle, or more precise the "inside" angle, $0 \leq \beta \leq \pi/2$. The proposed properties are: $\beta^{(min)}$ – the smallest distance between any two frame vectors, $\beta^{(avg)}$ – the average (statistical mean) distance for all frame vectors to its closest neighbor, or $\beta^{(mse)}$ – the average, taken in the "mean square sense" which will be explained in Subsection 7.3.3, of the distance between all possible combinations of pairs of frame vectors. In Subsection 7.3.3 we will also show that $\beta^{(mse)}$ can be calculated from the eigenvalues of the frame operator.

The $\beta^{(mse)}$ property tells how "clustered" the frame vectors are. A perhaps more intuitive property that measure how "clustered" the frame vectors are, is the average (statistical mean) distance for all frame vectors to the cluster center, we denote this property $\beta^{(avg2)}$. The frame vectors (usually) form a cluster on the surface of the unit ball, the center of this cluster can be defined in two different ways: The sum of all frame vectors is a vector and when this vector is normalized it is the cluster center. This center will depend on the sign of the frame vectors to be appropriate in a representation context, the representation is independent of the sign of the frame vectors, the frame vectors should point in "approximately the same direction", see point 2 in Subsection 7.1.3. A better alternative to define the cluster center is to use the eigenvector corresponding to the largest eigenvalue of the frame operator, see Subsection 7.3.1. The vector for the cluster center will then be the unit length vector, $\mathbf{x}$, for which $\|\mathbf{F}^T\mathbf{x}\|^2$ has its maximum value, B, Equation 7.2. The latter definition of the cluster center should be used for the $\beta^{(avg2)}$ property. An advantage of the β-properties compared to the r-properties is that the β-properties are independent of the signal class and can be calculated from the frame alone.

Signal representation can be done in many ways using a frame, one of the benefits compared to a transform is that the many frame vectors give more flexibility when selecting the weights. Using the dual frame to find the weights gives an exact representation and minimum 2-norm of the weights where generally all the K frame vectors are used. Using the Basis Pursuit algorithm gives an exact representation and minimum 1-norm of the weights, only N frame vectors are used. For a sparse representation the primary demand for the weights is that only a limited number of the weights can be non-zero. Gen-

erally this demand will make it impossible to have no representation error, so the second objective for the weight selection is to have a small representation error. A third objective may be to have the norm of weights small, but this is often ignored. Ignoring the third objective and selecting the weights using a full search algorithm to minimize the representation error may sometimes give a very large norm for the weights. The greedy vector selection algorithms select the weights such that the value for the norm of the weights is often smaller than if the weights were selected by a full search algorithm which will find the optimal weights in the sense that the error is as small as possible given the sparseness constraint. When the norm of the optimal weights is large, the greedy vector selection algorithms (especially the BMP method) are less likely to find this optimal solution. So we may say that the greedy vector selection algorithms, in a hidden way, also consider the third objective. In sparse representation the primary goal for frame design is to design the frame such that the sum of squared errors is minimized, the norm of the weights is paid no attention, except possibly indirectly by the choice of vector selection algorithm.

For some applications, as when the weights are to be quantized, it is unfavorable that the weights can be very large. Since neither frame design nor vector selection, i.e. the optimal full search algorithm, put any restrictions on the norm of the weights, it is interesting to know if there is a property of the frame that can be used to find an upper limit for the norm of the weights. For the non-sparse minimum norm solution (MOF) such a limit exists and is given by the lower frame bound, $\|\mathbf{w}\| \leq (1/\sqrt{A})\|\mathbf{x}\|$, – since the upper frame bound of the dual frame, used to find the weights in MOF, is $1/A$. Frames where the lower frame bound, A, is small do more often get large norm for the weights also for sparse representation. Thus it is natural to conclude that if it is important to have reasonable sized weights, it is not good if the lower frame bound is very small. But it is also so that even if A is not small, an optimal sparse representation may give a very large norm of the weights. In a similar way as the lower frame bound gives the limit for the norm of the weights found by the MOF algorithm, a frame property that limits the norm of the weights for an optimal sparse representation (selecting s frame vectors) may be defined such that $\|\mathbf{w}\| \leq (1/\sqrt{A_s})\|\mathbf{x}\|$. For many frames and small values of s the A_s property can be calculated. In Subsection 7.3.4 we will give some examples and discuss this a little bit more, an algorithm to calculate the A_s frame property (actually, a lower limit for it) is proposed and the value is calculated for some example frames.

The frame design method in Section 2.1 starts with an initial frame, often some randomly selected vectors from the training set. A relevant question to

ask is if frames designed using the same set of training vectors but different initial values will be much different from each other, and what could be an appropriate way to measure this difference. Also frames designed with different target sparseness factors could be interesting to compare to each other, if they are quite equal it is probably not necessary to design frames for many different sparseness factors, – a frame can be used with another sparseness factor than the one used during design. One measure that could be used is the value of the object function minimized during frame design, but this measure is very dependent on the set of training vectors (like the r_s properties) and can of course only be used for frames designed with the same sparseness factor. An alternative could be to use a property derived from the angle between a frame vector and its closest neighbor in the other frame, denoted $\theta^{(avg)}$. This property is defined and discussed more in Subsection 7.3.5.

Properties of overlapping frames are a little bit more complicated. The frame bounds and properties derived from the angles between frame vectors can be extended from the block-oriented frames to the overlapping frames. Measures using s, the number of frame vectors to use in each block, can not directly be extended to the overlapping frame case. Replacing s by the sparseness factor S is one possibility, then the property $r_S^{(mse)}$ (note: uppercase S) is the representation mean square error for a given signal and a sparseness factor. Actually, $20 \log_{10} r_S^{(mse)}$ is the SNR, it is plotted as a function of S for some frames in Figure 5.7. We think that this is good enough to illustrate (measure) the representation capability of an overlapping frame on a signal class, and do not pursue this track any further. Neither do we see the need for an extension of the A_s property to the overlapping frame case. The rest of the properties in Table 7.1 applied on overlapping frame are discussed in Section 7.4.

7.3 Mathematical details for the frame properties

Before starting on this section it may be a good idea to refresh the concepts and notation introduced in Chapter 1, as they, together with the frame concepts introduced in Section 7.1, will be used extensively throughout this section.

7.3.1 Singular value decomposition

The singular value decomposition (SVD) gives the essential information about a matrix, and the SVD of the frame matrix gives useful insight on the frame properties. It is well known that the frame bounds are the largest and smallest

eigenvalues of the frame operator [34], [16]. Using linear algebra it is easy to show that the eigenvalues of the frame operator are the square of the singular values of the frame. The SVD factorization of the frame matrix is

$$\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U} \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \sigma_N & 0 & \cdots & 0 \end{bmatrix} \mathbf{V}^T. \tag{7.5}$$

The matrices $\mathbf{U}$ and $\mathbf{V}$ are unitary, i.e. orthonormal, and of size $N \times N$ and $K \times K$ respectively. The singular value matrix, $\mathbf{\Sigma}$, has size $N \times K$ and is zero everywhere except on the main diagonal where the singular values are. Only the first N column vectors of $\mathbf{V}$ (rows of $\mathbf{V}^T$) are needed, the "silent" rightmost column vectors are discarded as they correspond to the $(K - N)$ rightmost columns of $\mathbf{\Sigma}$ which are only zeros. It is assumed that the singular values, $\{\sigma_n\}_{n=1}^N$ are positive real numbers and in non-increasing order; that is, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_N \geq 0$. When $\sigma_N > 0$, the frame matrix $\mathbf{F}$ is of full rank. All matrices, also the complex ones, can be factorized in this way where the singular values are real and non-negative [67].

Using the SVD the frame operator can be written as

$$\mathbf{S} = \mathbf{F}\mathbf{F}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{7.6}$$

where $\mathbf{\Lambda} = \mathbf{\Sigma}\mathbf{\Sigma}^T$ is the $N \times N$ diagonal matrix containing the eigenvalues of the frame operator. This is easy to see by writing $\mathbf{S}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$. $\mathbf{\Lambda}$ has the eigenvalues on the main diagonal and zeros elsewhere. The eigenvalues of the frame operator are denoted $\{\lambda_n\}_{n=1}^N$, and the values of each is the square of the singular value, i.e. $\lambda_n = \sigma_n^2$. It is also clear that the matrix $\mathbf{U}$ in the SVD of $\mathbf{F}$ is the eigenvectors of the frame operator.

In the rest of this section we let the frame be uniform. The sum of the eigenvalues for a frame operator is

$$\sum_n \lambda_n = \sum_n \sigma_n^2 = \|\mathbf{\Sigma}\|^2 = \|\mathbf{F}\|^2 = \sum_{k=1}^K \|\mathbf{f}_k\|^2 = K. \tag{7.7}$$

Since the frame bounds are given by $A = \lambda_N$, the smallest eigenvalue, and $B = \lambda_1$, the largest eigenvalue, the ranges for the frame bounds are

$$0 < A \leq K/N \leq B < K. \tag{7.8}$$

For tight frames the frame bounds are equal and all the eigenvalues have the same value, $A = B = \lambda_n = K/N$.

The SVD also shows that the eigenvalues can be related to the angles between the frame vectors. A $K \times K$ matrix can be defined as $\mathbf{F}^T\mathbf{F} = \mathbf{V}\mathbf{\Lambda}_K\mathbf{V}^T$. $\mathbf{\Lambda}_K$ is the $K \times K$ diagonal matrix with $\mathbf{\Lambda}$ as the upper left part and zeros elsewhere. The angle between frame vector i and frame vector j is denoted $\beta_{ij} = \angle\mathbf{f}_i\mathbf{f}_j$. Since the frame is uniform $\cos(\beta_{ij}) = [\mathbf{F}^T\mathbf{F}]_{ij}$. This gives the following relation

$$\sum_{i,j} \cos^2(\beta_{ij}) = \|\mathbf{F}^T\mathbf{F}\|^2 = \|\mathbf{\Lambda}\|^2 = \|\mathbf{S}\|^2 = \sum_n \lambda_n^2. \tag{7.9}$$

Let $\mathbf{F}$ be a frame and $\mathbf{A}$ be a unitary $K \times K$ matrix. From the SVD of $\mathbf{F}$ we see that the frames $\mathbf{F}$ and $\mathbf{F}\mathbf{A}^T$ will have the same frame operator, and thus the same eigenvalues of the frame operator and the same frame bounds. The angles between the frame vectors will generally be changed. Now let $\mathbf{A}$ be a unitary $N \times N$ matrix. The frames $\mathbf{F}$ and $\mathbf{A}\mathbf{F}$ will not have the same frame operator but the eigenvalues and consequently the frame bounds will be the same, as well as the angles between the frame vectors. Left multiplying the frame by a unitary matrix is the same as rotating the coordinate system.

In Figure 7.2 we see that the difference between two normalized vectors can be expressed by several measures. Since we are mainly interested in representation properties of the frame the most natural measure is $r = \sin\beta$, or sometimes the error squared (r^2). Only inner angles, i.e. angles in the range $0 \leq \beta \leq \pi/2$ need to be considered, if $\beta > \pi/2$ the angle $(\pi - \beta)$ can be used instead.

The frame bounds are important properties of a frame. In a "loose" way we may say that the frame bounds tell how close the frame is to a tight frame, for a transform they tell how close the transform is to an orthogonal transform. Let us illustrate the frame bounds by the two simple examples in Figure 7.3. From Equation 7.2 we see that the norm of the expression $\mathbf{F}^T\mathbf{x}$ is bounded between the frame bounds. $\mathbf{F}^T\mathbf{x}$ is a vector of length K where each element is the inner product, $\mathbf{f}_k^T\mathbf{x}$, between frame vector k and the signal vector. When both the signal and the frame vector are normalized this is the cosine of the angle between the two vectors. It can also be seen as the length of the projection of one vector onto the other vector. Thus the expression that is

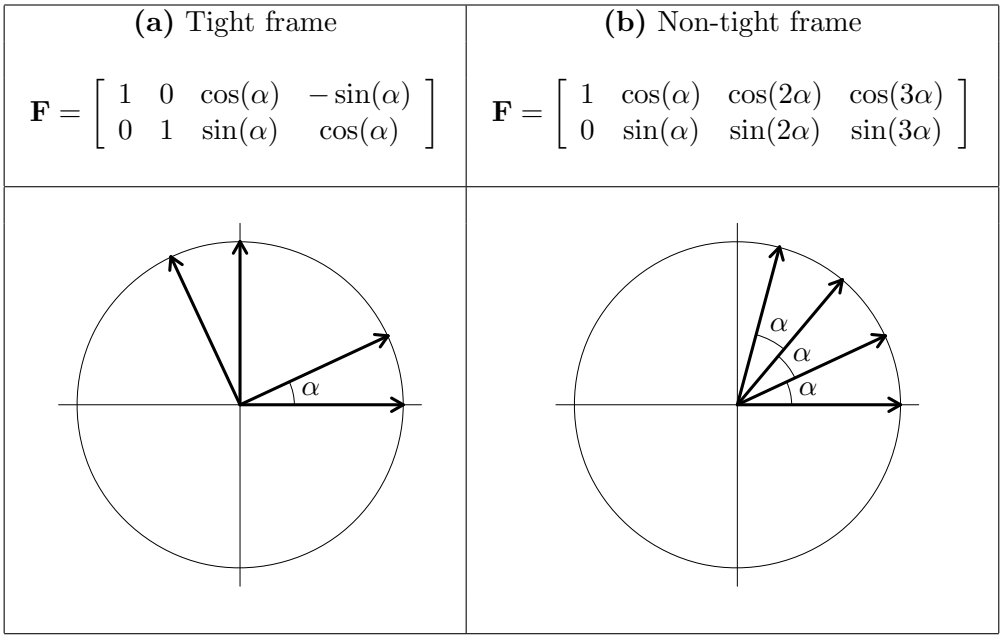


Figure 7.3: Two simple example frames in $\mathbb{R}^2$, These examples illustrate the difference between a tight frame and a non-tight frame as explained in the text. In (a) $0 < \alpha < \pi/2$ and in (b) $0 < \alpha \leq \pi/4$

limited (between the frame bounds) is the sum of the cosine squared of the angles between a signal vector and the frame vectors, $\sum_{k=1}^{K} \cos^2(\beta_k)$ where β_k is the angle between the signal vector and frame vector k. For tight frames, as the one in Figure 7.3 (a), the upper and lower frame bound is the same and the expression $\|\mathbf{F}^T\mathbf{x}\|^2$ is constant. The "energy" for the projections onto all frame vectors is constant for $\mathbf{x}$ in all possible directions, meaning that no direction in space is "preferred" by the frame and that the frame vectors are well spread in the space. Figure 7.3 (a) is the concatenation of two orthonormal bases. All frames that are the concatenation of one or more orthonormal bases are tight. But also many other kind of tight frames exist. In $\mathbb{R}^2$ it is easy to see that a $2 \times K$ frame, and the frame vectors evenly distributed between 0 and π, i.e. like Figure 7.3 (b) with $\alpha = \pi/K$, will be a tight frame. For all tight frames the frame bounds are $A = B = \lambda_n = K/N$ as can easily been seen from Equation 7.8, the frame in Figure 7.3 (a) has $A = B = 2$.

The non-tight frame in Figure 7.3 (b) is quite simple and it is possible to do symbolic calculation of its eigenvalues, i.e. find the eigenvalues without replacing α by a numeric value. The frame bounds are

$$\begin{aligned} A &= 2 - 2\,|\cos(\alpha)\cdot\cos(2\alpha)| = 2 - \,|\cos(3\alpha)+\cos(\alpha)| \\ B &= 2 + 2\,|\cos(\alpha)\cdot\cos(2\alpha)| = 2 + \,|\cos(3\alpha)+\cos(\alpha)|. \end{aligned} \tag{7.10}$$

We see that for small α the lower frame bound gets close to zero and the upper frame bound gets close to 4. Note that if $\alpha = 0$ this is not a frame since it does not span $\mathbb{R}^2$. The frame bounds are defined as the minimum and maximum of the expression $\|\mathbf{F}^T\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{F}\mathbf{F}^T\mathbf{x} = \sum_{k=1}^{K}\cos^2(\beta_k)$ for all $\|\mathbf{x}\| = 1$, Equation 7.2. From Figure 7.3 (b) we see (easiest to see for a small α) that this expression has maximum value when $\mathbf{x}$ has the angle $\frac{3}{2}\alpha$ with the x-axis, this is the vector "most parallel" to the frame vectors. It also happens to be the eigenvector corresponding to the largest eigenvalue (of the frame operator $\mathbf{S} = \mathbf{F}\mathbf{F}^T$). The minimum of $\|\mathbf{F}^T\mathbf{x}\|^2$ is for $\mathbf{x}$ with the angle $\frac{3}{2}\alpha + \pi/2$ with the x-axis, which is the eigenvector corresponding to the smallest eigenvalue. Generally, the eigenvector corresponding to the smallest eigenvalue of $\mathbf{S}$ minimize the expression $\mathbf{x}^T\mathbf{S}\mathbf{x}$ while the eigenvector corresponding to the largest eigenvalue maximize the expression. For a small α the frame bounds are close to the smallest and largest frame bounds possible for uniform frames, Equation 7.8, and the frame is far away from being a tight frame. As α gets larger the frame bounds become more equal to each other and the frame gets closer to a tight frame. For $\alpha = \pi/4$ $A = B$ and the frame is tight. This way we can say that the frame bounds tell how close the frame is to a tight frame.

Let us now consider signal representation. Many different weights can give perfect reconstruction, $\mathbf{x} = \tilde{\mathbf{x}} = \mathbf{F}\mathbf{w}$ for different choices of $\mathbf{w}$. The method of frames (MOF) is often used to find the weights, it applies the dual frame and it gives perfect reconstruction and the solution that minimize the norm of the weights. To have a small norm of the weights is often a desirable property of the representation. Even if MOF finds the minimum norm solution, the ratio $\|\mathbf{w}\|^2/\|\mathbf{x}\|^2$ can be large. Remember that the dual frame is used on the analysis side (find the weights) and the frame is used on the synthesis side (reconstruction).

The range for the weights will be

$$1/K \leq 1/B \leq \|\mathbf{w}\|^2/\|\mathbf{x}\|^2 \leq 1/A < \infty \tag{7.11}$$

where A and B are the frame bounds for the synthesis frame, Equation 7.2. The range of the frame bounds are given by Equation 7.8. The lower frame bound, A, is especially important since it gives the upper limit for this ratio. For tight frames A and B are equal and the ratio $\|\mathbf{w}\|^2/\|\mathbf{x}\|^2 = 1/A = N/K$ is constant, this is similar to the conservation of energy property of orthogonal transforms and filter banks. For non-tight frames the ratio $\|\mathbf{w}\|^2/\|\mathbf{x}\|^2$ will be limited by $1/A$, it is clear that small values of A may give large values for the norm of the weights. For non-tight frames the lower bound can be very close to zero, one example is Figure 7.3 (b) with small α. The upper bound will always be limited as long as the frame is uniform.

7.3.2 Representation error

In previous section we saw that using the MOF to find the weights gives the minimum norm solution. The representation is non-sparse and exact. Using a frame and an approximative *sparse* representation some error is accepted, but we want to keep the error limited and to have some control of it. The relative error for the representation of a signal vector is $r_l = \|\mathbf{x}_l - \tilde{\mathbf{x}}_l\|/\|\mathbf{x}_l\|$, where $\tilde{\mathbf{x}}_l$ is the sparse approximation using s of the frame vectors. Be careful not to confuse the scalar r_l with the residual vector $\mathbf{r} = \mathbf{x}_l - \tilde{\mathbf{x}}_l$. The properties $r_s^{(max)}$, $r_s^{(avg)}$, and $r_s^{(mse)}$ from Table 7.1 characterize the relative representation error.

The representation error depends on the signal and the frame, and the vector selection algorithm used, this makes it complicated to find a good frame property that tells how large we can expect the representation error to be. Suppose we know the distribution, i.e. the N-dimensional probability density function (pdf), for a signal class, or more likely have a large number L of representative training vectors (each of length N) from the signal class, then we could find (estimate) the pdf for the relative representation error for a given frame and a sparseness factor, using s of the frame vectors for each signal vector. The estimated pdf for the relative representation error is a continuous function that characterize the sparse representation capabilities of the frame, and not a simple frame property. The properties $r_s^{(max)}$ and $r_s^{(avg)}$ are directly derived from the pdf.

The proposed properties for the representation error $r_s^{(max)}$, $r_s^{(avg)}$, and $r_s^{(mse)}$ are signal dependent. Two classes of signals are of special interest. First, the signal class for which the frame was trained. A large representative set of vectors from this class may be the set of L vectors used during frame design, assuming the frame was designed as described in Chapter 2. Defining the relative error for the representation of each signal vector as $r_l = \|\mathbf{x}_l - \tilde{\mathbf{x}}_l\|/\|\mathbf{x}_l\|$,

where $\tilde{\mathbf{x}}_l$ is the sparse approximation using s of the frame vectors, gives the following definitions

$$\begin{aligned} r_s^{(max)} &= \max_l r_l \\ r_s^{(avg)} &= \frac{1}{L}\sum_{l=1}^{L} r_l \\ r_s^{(mse)} &= \sqrt{\frac{1}{L}\sum_{l=1}^{L} r_l^2} \end{aligned} \tag{7.12}$$

The second signal class is the signals generated by a *real white Gaussian random process*[2]. N subsequent samples from this process form a signal vector, non-overlapping vectors will be independent of each other and their pdf will be the N-dimensional real Gaussian distribution. Normalizing these signal vectors their distribution will be uniform on the unit ball (UB) in $\mathbb{R}^N$, the value of the pdf will be constant $f_{\mathbf{x}}(\mathbf{x}) = 1/S_N(1)$ where $S_N(1)$ is the surface of the unit ball. Since all possible unit length signal vectors are contained in this signal class the $r_s^{(max)}$ property for the Gaussian signal will be as large as possible for the frame, i.e. no other signal classes will have larger max than this. The definitions of the proposed frame properties for the random Gaussian signal class may be written

$$\begin{aligned} r_s^{(max)} &= \max_{\mathbf{x}\in UB} \|\mathbf{x} - \tilde{\mathbf{x}}\| \\ r_s^{(avg)} &= \frac{1}{S_N(1)} \int_{UB} \|\mathbf{x} - \tilde{\mathbf{x}}\| \, \mathrm{d}\mathbf{x} \\ r_s^{(mse)} &= \sqrt{\frac{1}{S_N(1)} \int_{UB} \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 \, \mathrm{d}\mathbf{x}} \end{aligned} \tag{7.13}$$

Let us look at some examples for the $r_s^{(max)}$ property as defined in Equation 7.13. Selecting only one vector, $s = 1$, this is like a shape-gain quantizer and the property has a simple geometric interpretation. A vector $\mathbf{x}$ is selected in the space $\mathbb{R}^N$ such the angle to the closest frame vector is maximum, then

[2]Each signal sample is normal distributed, $x(n) \sim N(0, 1)$, and independent of other signal samples.

(a) 3×6 tight frame

$$\begin{bmatrix} 1 & 0 & 0 & \frac{\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & \frac{-\sqrt{2}}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

(b) 4×8 tight frame

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

(c) 3×6 tight frame

$$\begin{bmatrix} c & c & 0 & 0 & s & -s \\ s & -s & c & c & 0 & 0 \\ 0 & 0 & s & -s & c & c \end{bmatrix}$$

(d) 4×8 tight frame

$$\begin{bmatrix} c & c & 0 & 0 & 0 & 0 & s & -s \\ s & -s & c & c & 0 & 0 & 0 & 0 \\ 0 & 0 & s & -s & c & c & 0 & 0 \\ 0 & 0 & 0 & 0 & s & -s & c & c \end{bmatrix}$$

Figure 7.4: Four tight frames. For (c) and (d) the constants are $c = \cos\alpha = \sqrt{0.5 + 0.1\sqrt{5}}$ and $s = \sin\alpha = \sqrt{0.5 - 0.1\sqrt{5}}$ where $2\alpha = \arctan 2$. The six frame vectors in (c) are six of the twelve vertices in an icosahedron, a regular polyhedron made up by twenty equilateral triangles where the length of each edge is $2s$.

$r_1^{(max)}$ is the sine of this angle, Figure 7.2. Even if this seems simple, it is hard to search the entire unit ball for this vector when the space has many dimensions, i.e. $N \geq 4$. But in $\mathbb{R}^2$ this is easy by visualizing the frame like in Figure 7.3. For frame (a) and $\alpha \leq \pi/4$ the largest error is when $\mathbf{x}$ has an angle $\beta = \pi/4 + \alpha/2$ to the x-axis, then $r_1^{(max)} = \sin(\pi/4 - \alpha/2)$. If α is selected to minimize the $r_1^{(max)}$ property, then we should select $\alpha = \pi/4$, and $r_1^{(max)} = \sin(\pi/8)$. For frame (b) the "worst" signal is when $\mathbf{x}$ has an angle $\beta = \pi/2 + 3\alpha/2$ to the x-axis, then $r_1^{(max)} = \sin(\beta)$, for small α this is close to 1. Now, if α is selected to minimize the $r_1^{(max)}$ property, then we should select $\alpha = \pi/4$, and this gives $r_1^{(max)} = \sin(\pi/8)$. For this value of α the frame in (b) is tight, and equal the frame in (a) with $\alpha = \pi/4$. In $\mathbb{R}^2$ the $2 \times K$ frame with minimum value for $r_1^{(max)}$ is a frame like in Figure 7.3 (b) and $\alpha = \pi/K$, this frame will be tight.

In $\mathbb{R}^N$ for $N > 2$ it is more difficult to find the $N \times K$ frames that minimize the $r_1^{(max)}$ property. The multi-dimensional spaces are difficult to visualize. Let us illustrate the increased complexity of the spaces $\mathbb{R}^3$ and $\mathbb{R}^4$ compared to $\mathbb{R}^2$ by trying to find the $N \times 2N$ frames that minimize the $r_1^{(max)}$ property. For the $N = 2$ case we found that the solution is the concatenation of two

orthogonal bases, the 2×2 identity matrix and the 2×2 Haar matrix. Now let us look at the case when $N = 4$ the frame is the concatenation of the 4×4 identity matrix and the 4×4 Haar matrix, Figure 7.4 (b). Each of the basis vectors of the Haar matrix has maximal value for the angle to its closest neighbor unit vector, in fact the angle is the same to all the unit vectors (basis vectors of the identity matrix). This will be valid for $N = 2^p$ and p is integer, the cases where the Haar bases are easily defined. From Figure 7.3 we saw that the 2×4 frame that minimize the $r_1^{(max)}$ value is the "identity+Haar" frame. A relevant question is if this will be so also for larger "identity+Haar" frames. The answer is no, as will be shown by comparing two frame, frame (b) and (d) in Figure 7.4. The four example frames in Figure 7.4 are all frames that were selected such that the value of the $r_1^{(max)}$ property should be small. By searching the unit ball we found the $\mathbf{x}$ vectors that maximized the representation errors, and $r_1^{(max)}$. The results were

Frame	$N \times K$	$\mathbf{x}^T$	$r_1^{(max)}$
(a)	3×6	$[-\sqrt{1-2a^2}, a, a]$	$\sqrt{1-a^2} \approx 0.7345$
(b)	4×8	$[\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}]$	$\frac{1}{2}\sqrt{3} \approx 0.8660$
(c)	3×6	$[1, 1, 1]/\sqrt{3}$	$\sqrt{\frac{2}{3} - \frac{2}{15}\sqrt{5}} \approx 0.6071$
(d)	4×8	$[\frac{1}{2}\sqrt{2}, 0, \frac{1}{2}\sqrt{2}, 0]$	$\frac{1}{2}\sqrt{3 - \frac{1}{5}\sqrt{5}} \approx 0.7989$

The constant used for frame (a) is $a = 0.67859834454585$. We see that frame (d) is better than frame (b) with regard to the $r_1^{(max)}$ property, but still better 4×8 frames may be found. The 3×6 frame in (c) that uses the same constants as the frame in (d) though is the best possible frame of its size. This will be shown in the end of the next subsection.

Some interesting problems can be defined for frames using the $r_1^{(max)}$ property. First the problem already looked at: Find the $N \times K$ frames that minimize the $r_1^{(max)}$ property. From Subsection 7.3.1 we know that the frame $\mathbf{QF}$ ($\mathbf{Q}$ a unitary $N \times N$ matrix) will be the frame $\mathbf{F}$ in a rotated coordinate system, and thus have the same $r_s^{(max)}$ property. Another problem may be to find the minimum number of frame vectors needed to have $r_s^{(max)}$ for the best frame below a given value. These problems belong to the mathematical field.

7.3.3 Angle between frame vectors

It is a reasonable assumption that the frame vectors cover the space best when they are well spread out over the space $\mathbb{R}^N$. When the frame vectors are well spread no frame vectors are close to each other. This motivates the use of the $\beta^{(min)}$ property in Table 7.1 as an alternative to the $r_1^{(max)}$ property found by using a signal class that cover the entire surface of the unit ball. The advantage of the $\beta^{(min)}$ property is that it can be calculated from the frame alone, it is not necessary to search the space for the $\mathbf{x}$ vector with largest angle to the frame vectors. Note that "natural signals" usually covers only a smaller part of the space $\mathbb{R}^N$, and consequently the frame vectors of a frame designed for such classes would not be well spread, in the contrary they will often be well clustered. But in both cases it is relevant with properties that measures how the frame vectors are spread, or clustered, in the space. For a uniform frame the $\beta^{(min)}$ property is defined as

$$\beta^{(min)} = \min_{j \neq i} \beta_{ij} \qquad \text{where} \quad \cos(\beta_{ij}) = [\mathbf{F}^T\mathbf{F}]_{ij}. \tag{7.14}$$

β_{ij} is the angle between frame vector i and frame vector j, $\mathbf{f}_i$ and $\mathbf{f}_j$. The $\beta^{(min)}$ and the $r_1^{(max)}$ properties are two totally different properties, but the following connection seems to be valid: when $\beta^{(min)}$ is large, i.e. close to the maximal possible value for a given frame size, the $r_1^{(max)}$ property will be small, i.e. close to the minimal possible value for a given frame size. There is also another reason to use the $\beta^{(min)}$ property. If $\beta^{(min)}$ is small then two of the frame vectors are almost identical. If two frame vectors are close to each other, and a vector $\mathbf{x}$ is in the plane spanned by these two frame vectors and almost perpendicular to them, then a perfect representation of $\mathbf{x}$ is possible using the two frame vectors. However, the weights will be large, and as pointed out previously, to have the ratio $\|\mathbf{w}\|/\|\mathbf{x}\|$ large may be problematic in some applications. The smaller the $\beta^{(min)}$ property gets the larger this ratio may be. This, and the connection to the $r_1^{(max)}$ property, speak for the $\beta^{(min)}$ property as a useful frame property.

The $\beta^{(min)}$ property only depends on the two frame vectors closest to each other. An alternative way to measure the spread of the frame vectors is to take the average for all frame vectors of the angle to its closest neighbor. The $\beta^{(avg)}$ property is

$$\beta^{(avg)} = \frac{1}{K} \sum_i \min_{j \neq i} \beta_{ij}. \tag{7.15}$$

For the frames in Figure 7.3 and a small value of α the frame vectors are better spread in the space for the tight frame in (a) than for the non-tight frame in (b), but both the $\beta^{(min)}$ and $\beta^{(avg)}$ properties will be the same for the two frames, $\beta^{(min)} = \beta^{(avg)} = \alpha$. The alternative property $\beta^{(avg2)}$, defined as the average for all frame vectors of the angle to the cluster center, more directly measure how clustered a frame is. A problem here is to define the cluster center in a good way. We think that the eigenvector corresponding to the largest eigenvalue of the frame operator, i.e. the upper frame bound B, is the better choice. For some frames this is not a unique vector, but for "well clustered" frames this vector will be unique. Denoting this vector $\mathbf{f}_B$ and its (inner) angle to frame vector k by $\angle \mathbf{f}_B \mathbf{f}_k$ give the following definition

$$\beta^{(avg2)} = \frac{1}{K} \sum_k \angle \mathbf{f}_B \mathbf{f}_k. \tag{7.16}$$

The value of the $\beta^{(avg2)}$ property of frame (a) in Figure 7.3 is $\pi/4$, note that this is independent of α and in fact also of the "cluster center" $\mathbf{f}_B$. For frame (b) in Figure 7.3 $\beta^{(avg2)} = \alpha$. We may conclude that the $\beta^{(avg2)}$ property is useful and has a clear geometric meaning when the frame vectors are well clustered and the cluster center is well defined.

We should think that these three "β-properties" were enough, but we propose yet another one which both has a clear geometric interpretation and a simple mathematical foundation as it can be calculated from the eigenvalues of the frame operator. Now we use the r^2 measure for the distance between two frame vectors instead of the angle β, see Figure 7.2. We take the average (arithmetic mean) of all the different possible pairs of frame vectors, this is like a "mean square error"-property even if no error is involved here. This gives the "average r^2"-property, but to be in line with the other "β-properties" we prefer to use the corresponding angle, which we denote $\beta^{(mse)}$. The "average r^2"-property can be written as $\sin^2(\beta^{(mse)})$. Its definition can be written in mathematical terms

$$
\begin{aligned}
\sin^2(\beta^{(mse)}) &= \frac{1}{K(K-1)}\sum_{i\neq j}\sin^2\beta_{ij} = \frac{1}{K(K-1)}\sum_{ij}\sin^2\beta_{ij} \qquad (7.17)\\
&= \frac{1}{K(K-1)}\sum_{ij}(1-\cos^2\beta_{ij}) = \frac{K}{K-1} - \frac{1}{K(K-1)}\sum_{ij}\cos^2\beta_{ij}\\
&= \frac{K}{K-1} - \frac{1}{K(K-1)}\sum_{ij}[\mathbf{F}^T\mathbf{F}]_{ij}^2 = \frac{K}{K-1} - \frac{1}{K(K-1)}\sum_{n}\lambda_n^2\\
&= \frac{1}{K(K-1)}\Big(K^2 - \sum_n \lambda_n^2\Big) = \frac{K}{K-1}\Big(1 - \frac{1}{K^2}\sum_n \lambda_n^2\Big)
\end{aligned}
$$

Note that the angle of a frame vector to itself, which of course is zero, does not count when calculating the average in this measure, but it is included, but not contributing, in the sum of the sine squared of the angles. For the tight frame (a) in Figure 7.3 $\beta^{(mse)}$ is independent of α, it is 54.7 degrees, as it will be for all tight 2×4 frames. The frame in (b) will be clustered for small values of α, and then the $\beta^{(mse)}$ property will also be small. A small table illustrates this (the angles are written in degrees)

α	1	2	5	10	15	20	25	30	45
$\beta^{(mse)}$	1.8	3.6	9.1	18.0	26.6	34.5	41.6	47.4	54.7

The derivation in Equation 7.17 assumes that the frame is uniform, this assumption is necessary for $\cos(\beta_{ij}) = [\mathbf{F}^T\mathbf{F}]_{ij}$ to be true. It is interesting to know the range of the expression $\frac{1}{K^2}\sum_n \lambda_n^2$ used in Equation 7.17, i.e. the smallest and the largest values this expression can have. The largest value is when one eigenvalue of the frame operator is large and the rest is almost zero, the largest eigenvalue must be smaller than the sum of all $\sum_n \lambda_n$. The smallest value is when all eigenvalues are equal, then the value of each eigenvalue is $\frac{1}{N}\sum_n \lambda_n$. This gives the following range for the expression as

$$
\frac{1}{N} \;\leq\; \frac{1}{K^2}\sum_n \lambda_n^2 \;=\; \frac{\sum_n \lambda_n^2}{(\sum_n \lambda_n)^2} \;<\; 1. \qquad (7.18)
$$

The alternative way to write the expression in Equation 7.18 is easily verified since $\sum_n \lambda_n = K$. Equations 7.17 and 7.18 show that the angle $\beta^{(mse)}$ is also a measure of the "spread" of the eigenvalues, a large angle $\beta^{(mse)}$ (the frame vectors are "spread") corresponds to the situation where $\sum_n \lambda_n^2$ is small (the eigenvalues are clustered), and a small angle $\beta^{(mse)}$ (the frame vectors are

"clustered") corresponds to the situation where $\sum_n \lambda_n^2$ is large (the eigenvalues are "spread", typically one is large, the upper frame bound B, the rest are small).

We will now look a little bit closer on the $\beta^{(mse)}$ property of tight frames. This case gives the maximal value for $\beta^{(mse)}$ for a $N \times K$ frame. For a tight frame all eigenvalues of the frame operator are equal, for a $N \times K$ frame $\lambda_n = K/N$, and the $\beta^{(mse)}$ property is simply

$$\begin{aligned} \sin^2(\beta^{(mse)}) &= \frac{K(N-1)}{N(K-1)} \\ \beta^{(mse)} &= \arcsin\sqrt{\frac{K(N-1)}{N(K-1)}}. \end{aligned} \tag{7.19}$$

An tiny remark in the end: If the average is taken over all possible pairs of frame vectors, including the frame vector to itself, the measure $\sin^2(\beta^{(mse)})$ should be multiplied by $K(K-1)$ (the number of pairs used) and divided by K^2 (the total number of pairs). For a tight frame this gives a result also independent of the number of frame vectors K

$$\frac{K(K-1)}{K^2}\sin^2(\beta^{(mse)}) = \frac{N-1}{N} \tag{7.20}$$

These properties based on angles between frame vectors were calculated for the tight frame in Figure 7.4 and the result is presented below (angles in degrees)

Frame	$N \times K$	$\beta^{(min)}$	$\beta^{(avg)}$	$\beta^{(mse)}$
(a)	3×6	45	45	$\arcsin\sqrt{4/5} \approx 63.43$
(b)	4×8	60	60	$\arcsin\sqrt{6/7} \approx 67.79$
(c)	3×6	63.43	63.43	$\arcsin\sqrt{4/5} \approx 63.43$
(d)	4×8	63.43	63.43	$\arcsin\sqrt{6/7} \approx 67.79$

In the end of this subsection we will prove the optimality of frame (c) in Figure 7.4, optimal in the sense that this frame is the 3×6 frame with the largest possible value for the $\beta^{(min)}$ property and the smallest possible value for the $r_1^{(max)}$ property. For the frame in (c) the minimum angle is equal to the average, $\beta^{(min)} = \beta^{(mse)} = \arcsin\sqrt{4/5} = \arctan 2$. The fact that the average is taken in "mean square sense" does not matter here. Since the frame is tight and $\beta^{(min)} = \beta^{(mse)}$ all (inside) angles between any two frame vectors are the same, $\beta_{ij} = \beta^{(min)}$. The $\beta^{(mse)}$ is given by the frame size for a tight frame, and we see from Equation 7.17 that $\beta^{(mse)}$ is always larger for a tight frame than for a non-tight frame of the same size. This shows that $\beta^{(min)}$ for the frame in (c) has the largest possible value for all 3×6 frames. The maximum error, $r_1^{(max)}$, will be for a vector $\mathbf{x}$ that is in the middle of three frame vectors, sign of the frame vectors should be selected such that the angle between frame vectors are smaller or equal to $\pi/2$, then the distance from $\mathbf{x}$ to each of these three frame vectors is the same. Since all (inside) angles are the same for this frame we may chose any three frame vectors and the $\mathbf{x}$ in the middle of these will have maximum error. Since the angles between the frame vectors are maximum, this error must be the smallest possible $r_1^{(max)}$ property of all 3×6 frames. For the frame in (c) we can use $\mathbf{f}_1$, $\mathbf{f}_3$ and $\mathbf{f}_5$, then $\mathbf{f}_1 + \mathbf{f}_3 + \mathbf{f}_5 = [c+s, c+s, c+s]^T$ and $\mathbf{x} = [1,1,1]^T/\sqrt{3}$. When β is the angle between $\mathbf{x}$ and $\mathbf{f}_1$ we have $\cos\beta = \mathbf{x}^T\mathbf{f}_1 = (c+s)/\sqrt{3}$ and $r_1^{(max)} = \sin\beta = \sqrt{1-((c+s)/\sqrt{3})^2} = \sqrt{\frac{2}{3} - \frac{2}{3}cs} = \sqrt{\frac{2}{3} - \frac{2}{15}\sqrt{5}}$. The constants c and s are as defined in Figure 7.4. The twelve points in three dimensional space given by the columns of $\mathbf{F}$ and $-\mathbf{F}$, $\mathbf{F}$ as frame (c) in Figure 7.4, are the twelve vertices of an icosahedron, one of the five regular polyhedra, which are also known as the Platonic solids and have been known since the time of the ancient Greeks.

7.3.4 Norm of the weights

Let us start this subsection by two examples. They illustrate that the norm of the weights may be very large for an optimal sparse representation. We have previously mentioned that for some applications it may be a problem if the ratio $\|\mathbf{w}\|/\|\mathbf{x}\|$ is large, for the example frames in Figure 7.5 this may happen. First note that the lower frame bounds for these frames are not very small so the ratio $\|\mathbf{w}\|/\|\mathbf{x}\|$ will not be very large if the weights are found by the MOF, – remember that for the MOF the norm of the weights is limited by $\|\mathbf{w}\| \leq (1/\sqrt{A})\|\mathbf{x}\|$.

For the 3×4 frame in (a) let the vector to be approximated be $\mathbf{x} = [0,1,0]^T$. If $s = 1$ a good approximation is found using column 2 of $\mathbf{F}$, $\mathbf{f}_2$. If two

(a) 3×4 non-tight frame

$$\begin{bmatrix} 1 & \sqrt{\frac{a}{2}(2-a)} & 0 & \sqrt{1-a^2} \\ 0 & 1-a & 0 & a \\ 0 & \sqrt{\frac{a}{2}(2-a)} & 1 & 0 \end{bmatrix}$$

Frame properties

a	A	B	$\beta^{(min)}$
0.1	0.5948	2.2141	5.74
0.01	0.8899	2.0216	0.57
0.001	0.9674	2.0021	0.06

(b) 4×5 non-tight frame

$$\begin{bmatrix} 1 & 0 & \sqrt{\frac{a}{3}(2-a)} & 0 & \sqrt{\frac{1}{2}(1-a^2)} \\ 0 & 1 & -\sqrt{\frac{a}{3}(2-a)} & 0 & \sqrt{\frac{1}{2}(1-a^2)} \\ 0 & 0 & 1-a & 0 & a \\ 0 & 0 & \sqrt{\frac{a}{3}(2-a)} & 1 & 0 \end{bmatrix}$$

Frame properties

a	A	B	$\beta^{(min)}$
0.1	0.5663	2	45.29
0.01	0.8589	2	45.00
0.001	0.9553	2	45.00

Figure 7.5: Two non-tight frames. The constant a is assumed to be a small positive number. The first three (a) or four (b) frame vectors are close to the basis of unit vectors in $\mathbb{R}^3$ or $\mathbb{R}^4$ respectively. The $\beta^{(min)}$ property is given in degrees.

frame vectors are allowed in the sparse representation, an exact solution can be found using $\mathbf{f}_1$ and $\mathbf{f}_4$. The weight for the $\mathbf{f}_4$ frame vector will be $1/a$ and the weight for the $\mathbf{f}_1$ frame vector will also be large in magnitude, the norm of the weights will be $\|\mathbf{w}\| = \frac{1}{a}\sqrt{2-a^2}$. We see that when a gets close to zero $\|\mathbf{w}\|$ gets large. Evidently this is connected to the fact that $\mathbf{x}$ is in the space spanned by two frame vectors very close to each other and $\mathbf{x}$ is almost orthogonal to both these vectors. The small value of the $\beta^{(min)}$ property of this frame indicate that these problems may occur, for a small a for example $a = 0.01$, $\beta^{(min)} = \arcsin a \approx a$ (radians).

The frame (b) in Figure 7.5 is an example of a frame where $\|\mathbf{w}\|/\|\mathbf{x}\|$ can be large even when neither A nor $\beta^{(min)}$ are small. Using this frame and letting the vector to represent be $\mathbf{x} = [0, 0, 1, 0]^T$ a solution with a small representation error can be found by selecting only one vector ($\mathbf{f}_3$). An exact solution can be found using three vectors ($\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_5$). For a small value of a the exact solution using three frame vectors will give a large value for $\|\mathbf{w}\|/\|\mathbf{x}\|$. The observation here is that the weights can get large when one of the frame vectors is close to (but not a member within) the space spanned by some few $(s-1)$ of the other frame vectors.

The question is: Is it possible to find a frame property that gives the limit for the ratio $\|\mathbf{w}\|/\|\mathbf{x}\|$ when the weights are selected by the full search algorithm? Let s be the allowed number of frame vectors to use, and let us assume that such a frame property exists and is similar to the lower frame limit A as

used in Equation 7.11, $\|\mathbf{w}\|/\|\mathbf{x}\| \leq 1/\sqrt{A_s}$. Here we propose an algorithm to find the value of A_s for a given frame. The algorithm examines all possible combinations of s frame vectors. It is $I = \binom{K}{s}$ different ways (possible combinations) to select s frame vectors of the K available ones, let us number these by $i = 1, 2, \ldots, I$. For combination i we build a $N \times s$ matrix where the s selected frame vectors are the columns, this matrix is denoted $\mathbf{F}_s^{(i)}$. This matrix is a frame in the space spanned by the s column vectors, not in the space $\mathbb{R}^N$, and considering it as a frame in this subspace the non-zero frame bounds exist. We are interested in the lower frame bound which we denote $A^{(i)}$. $A^{(i)}$ is the square of the smallest non-zero singular value of $\mathbf{F}_s^{(i)}$. Representing a vector in the space spanned by the selected frame vectors, the columns of $\mathbf{F}_s^{(i)}$, the MOF can be used to find the weights with minimum norm. The norm of the weights will be limited by $\|\mathbf{w}\|^2/\|\mathbf{x}\|^2 \leq 1/A^{(i)}$, Equation 7.11. The optimal sparse approximation must be represented by one of the I frames, $\{\mathbf{F}_s^{(i)}\}_{i=1}^{I}$, and the largest possible norm of the weights in this representation must be limited by

$$\|\mathbf{w}\|/\|\mathbf{x}\| \leq 1/\sqrt{A_s} \qquad \text{where} \quad A_s = \min_i A^{(i)} \tag{7.21}$$

It is only possible to calculate the A_s property for small values of s. The trivial case, $A_1 = 1$, only tells that the norm of the weights is smaller than, or equal to, the norm of the signal when only one vector is selected.

In the following table some properties and the norm of the weights for the frames in Figure 7.5 are displayed. The weights are for the optimal $s = 2$ solution for $\mathbf{x} = [0, 1, 0]^T$ for frame (a) and the optimal $s = 3$ solution for $\mathbf{x} = [0, 0, 1, 0]^T$ for frame (b).

frame	parameter a	$\|\mathbf{w}\|$	$1/\sqrt{A}$	$1/\sqrt{A_2}$	$1/\sqrt{A_3}$
(a)	0.1	14.11	1.2967	14.1244	42.9936
(a)	0.01	141.4178	1.06	141.4196	1410
(a)	0.001	1414.2132	1.0167	1414.2134	44709
(b)	0.1	14.1067	1.3289	1.8367	14.1244
(b)	0.01	141.4178	1.079	1.8476	141.4196
(b)	0.001	1414.2132	1.0231	1.8478	1414.2134

These results confirm that the $1/\sqrt{A_s}$ property gives the limit for the norm of the weights. The value of $1/\sqrt{A_2}$ is slightly larger than $\|\mathbf{w}\|$ for frame (a). The example vector is the "worst case" vector for the limit when a goes

towards zero, for the case when $a = 0.1$ another $\mathbf{x}$ vector would give a little bit larger norm for the weights, but always limited by Equation 7.21. The same observation is made for $1/\sqrt{A_3}$ and the norm of the weights for frame (b). The very high values for $1/\sqrt{A_3}$ for frame (a) illustrate another fact about the A_s property. The four different ways of selecting three out of four frame vectors give four different $\mathbf{F}_3^{(i)}$ frames, these four frames span the same space, they are all bases in the space $\mathbb{R}^3$. The A_s property uses the smallest lower frame bound, but a representation can be done by one of the other $\mathbf{F}_3^{(i)}$ frames and then the norm of the weights will be limited by the lower frame bound for this frame. Generally this may happen if the space of one (or more) of the $\mathbf{F}_s^{(i)}$ frames is the same as (or a subspace of) the space spanned by another combination of the frame vectors of $\mathbf{F}$, i.e. $\mathbf{F}_s^{(i)} \subseteq \mathbf{F}_s^{(j)}$ for $i \neq j$.

7.3.5 Difference between frames

The common measure "norm of difference", $\|\mathbf{F}^{(1)} - \mathbf{F}^{(2)}\|$, can often be used to measure the difference between two frames. But when the frames are used for representation permutation of the frame vectors is possible, and the norm of the difference is not appropriate. Here we suggest the average of the angles between each frame vector and the closest frame vector in the other frame. For two frames, $\mathbf{F}^{(1)}$ of size $N \times K_1$ and $\mathbf{F}^{(2)}$ of size $N \times K_2$, θ_{ij} is the angle between frame vector i in $\mathbf{F}^{(1)}$ and frame vector j in $\mathbf{F}^{(2)}$, and $\cos\theta_{ij} = [\mathbf{F}^{(1)T}\mathbf{F}^{(2)}]_{ij}$. The $\theta^{(avg)}$ measure is

$$\theta^{(avg)} = \frac{1}{K_1 + K_2}\Big(\sum_{i=1}^{K_1} \min_j \theta_{ij} + \sum_{j=1}^{K_2} \min_i \theta_{ij}\Big). \tag{7.22}$$

Since $\theta^{(avg)}$ measure uses the average in "both directions" the difference between $\mathbf{F}^{(1)}$ and $\mathbf{F}^{(2)}$ is the same as the difference between $\mathbf{F}^{(2)}$ and $\mathbf{F}^{(1)}$. This would not have been the case if only one of the sums were used in Equation 7.22.

7.4 Properties of overlapping frames

So far in this chapter only block-oriented frames have been considered. For the overlapping frames the situation is more complicated. We will here discuss the properties used for block-oriented frame in the context of overlapping frames,

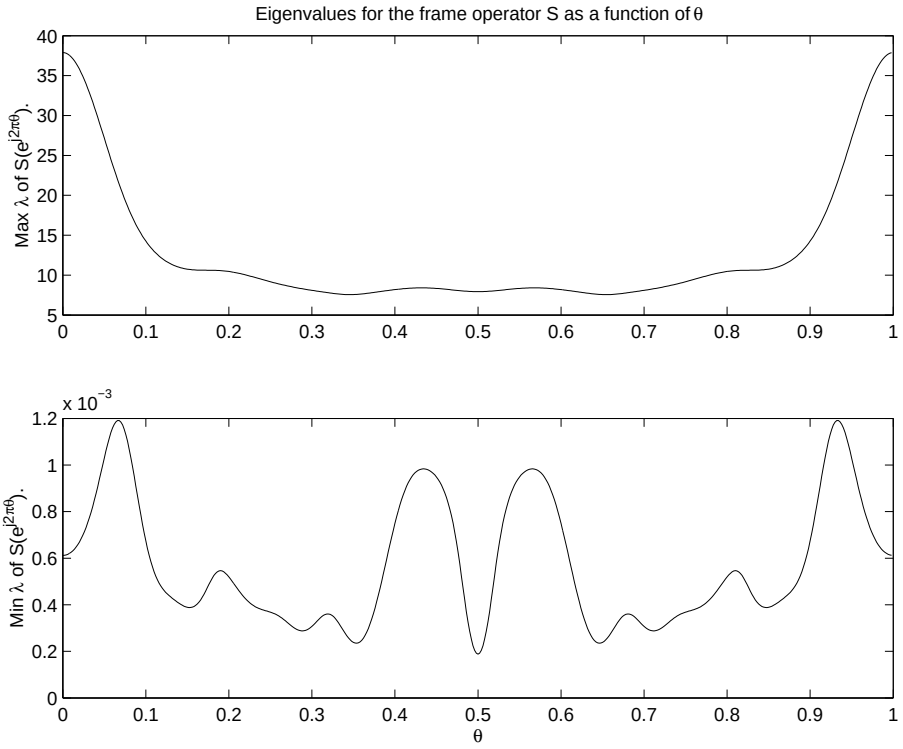


Figure 7.6: The largest and smallest eigenvalues of $\mathbf{S}(e^{j2\pi\theta})$ as a function of θ. Here the values for frame (e) in Figure 5.1 and Figure 5.2 are presented.

but we will focus on those properties where the extension to overlapping frames is quite simple.

For the block-oriented frames the singular value decomposition was useful and revealed many of the frame properties. For an overlapping frame the SVD analysis is not straightforward. An SVD of the large matrix $\mathcal{F}$ could be done, but this is computationally demanding since the order of SVD algorithms is cubic in the size of the matrix. One reason for doing the SVD of the frame was to find the singular values, and through them the eigenvalues of the frame operator. For an overlapping frame the frame operator and its eigenvalues can be found using the theory for oversampled filter banks [15] [8]. Thus, we find the eigenvalues of the frame operator rather than the SVD, including the singular values, for the overlapping frame.

For a block-oriented frame the frame operator can be expressed as a matrix, $\mathbf{S} = \mathbf{F}\mathbf{F}^T$. For the overlapping frame the frame operator is expressed using the polyphase matrix representation, Equation 7.4. The frame bounds are closely

connected to the eigenvalue decomposition of the frame operator, the frame bounds A and B are given by the essential infimum and supremum, respectively, of the eigenvalues $\lambda_n(\theta)$ of the polyphase matrix representation of the frame operator [8]. For values of θ between zero and one, the frame operator, $\mathbf{S}_\theta$, is found and eigenvalue decomposition is done, and the largest and smallest eigenvalues are found. This makes it possible to plot these eigenvalues as functions of θ, and then find/estimate the frame bounds. For the overlapping frame (e), with target sparseness factor $S = \frac{4}{32}$, in Section 5.1 these values are plotted in Figure 7.6. From this we can find that the lower frame bound is $A = 0.000188$ and the upper frame bound is $B = 37.89$.

For a uniform block-oriented frame we found that $\sum_n \lambda_n = \|\mathbf{F}\|^2 = K$, Equation 7.7. For the example frame in Figure 7.6 it is easily seen that $\sum_n \lambda_n(\theta) \neq K$, for small values of θ the largest eigenvalue alone is much larger than K. We will here show that an expression similar to Equation 7.7 exists for uniform overlapping frames. The expression is

$$\int_0^1 \sum_n \lambda_n(\theta)\, \mathrm{d}\theta = \|\mathbf{F}\|^2 = K, \tag{7.23}$$

and the proof is as follows: The polyphase representation of an overlapping frame is $\mathbf{R}(z) = \sum_{p=0}^{P-1} \mathbf{F}_p z^{-p}$, $\mathbf{R}(z)$ is a $N \times K$ matrix of polynomials in z, in the proof only z-values on the unit circle are used, i.e. $z = e^{j2\pi\theta}$ for $0 \leq \theta < 1$, $j = \sqrt{-1}$. For a certain value of θ the matrix is denoted $\mathbf{R}_\theta$. It is a complex valued matrix and a singular value decomposition of this matrix is possible. The polyphase matrix and the frame operator, Equation 7.4, can be written

$$\begin{aligned} \mathbf{R}_\theta &= \mathbf{U}_\theta \mathbf{\Sigma}_\theta \mathbf{V}_\theta^H \qquad \text{and} \\ \mathbf{S}_\theta &= \mathbf{R}_\theta \mathbf{R}_\theta^H = \mathbf{U}_\theta \mathbf{\Lambda}_\theta \mathbf{U}_\theta^H \end{aligned} \tag{7.24}$$

where $\mathbf{U}_\theta$ and $\mathbf{V}_\theta$ are unitary $N \times N$ matrices, and $\mathbf{\Sigma}_\theta$ is a matrix with real non-negative singular values on the main diagonal and zeros elsewhere. The singular values are denoted $\sigma_n(\theta)$ and the eigenvalues of the frame operator $\lambda_n(\theta)$, where $\lambda_n(\theta) = \sigma_n^2(\theta)$. Now we can write

$$\begin{aligned}
\int_0^1 \sum_n \lambda_n(\theta)\,\mathrm{d}\theta &= \int_0^1 \sum_n \sigma_n^2(\theta)\,\mathrm{d}\theta = \int_0^1 \|\mathbf{\Sigma}_\theta\|^2\,\mathrm{d}\theta = \int_0^1 \|\mathbf{R}_\theta\|^2\,\mathrm{d}\theta \\
&= \int_0^1 \|\sum_{p=0}^{P-1} \mathbf{F}_p e^{-j2\pi p\theta}\|^2\,\mathrm{d}\theta \\
&= \int_0^1 \sum_{n=1}^{N}\sum_{k=1}^{K} \quad |\sum_{p=0}^{P-1} [\mathbf{F}_p]_{nk} e^{-j2\pi p\theta}|^2 \quad \mathrm{d}\theta \\
&= \int_0^1 \sum_{n=1}^{N}\sum_{k=1}^{K} \Big(\sum_{p=0}^{P-1} [\mathbf{F}_p]_{nk} e^{-j2\pi p\theta}\Big)\Big(\sum_{i=0}^{P-1} [\mathbf{F}_i]_{nk} e^{j2\pi i\theta}\Big)\,\mathrm{d}\theta \\
&= \int_0^1 \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{p=0}^{P-1}\sum_{i=0}^{P-1} [\mathbf{F}_p]_{nk}[\mathbf{F}_i]_{nk} e^{j2\pi(i-p)\theta}\,\mathrm{d}\theta \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{p=0}^{P-1}\sum_{i=0}^{P-1} [\mathbf{F}_p]_{nk}[\mathbf{F}_i]_{nk} \int_0^1 e^{j2\pi(i-p)\theta}\,\mathrm{d}\theta \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{p=0}^{P-1} [\mathbf{F}_p]_{nk}^2 = \sum_{p=0}^{P-1} \|\mathbf{F}_p\|^2 = \|\mathbf{F}\|^2 = K \qquad (7.25)
\end{aligned}$$

Equation 7.23 is the same as Equation 23 in [39].

Some properties to measure for the representation capabilities for a block-oriented frame was found in Subsection 7.3.2. It was difficult to find a simple frame property to use, and the proposed measures are signal dependent. For the overlapping frames we will get an additional problem limiting the (part of) frame to consider and finding (and using) the number of non-zero weights. This justifies that it is reasonable not to use the approach of Subsection 7.3.2 for the overlapping frame.

Frame properties derived from the angles between frame vectors are relevant measures also for an overlapping frame. As for a block-oriented frame these properties tell how clustered the frame vectors are. Both the $\beta^{(min)}$, the smallest angle between any two frame vectors, and the $\beta^{(avg)}$, the average of the angles for all the frame vectors to its closest neighbor, properties can be calculated by looking on a small part of the large band-diagonal matrix $\mathcal{F}$. Let the example where $P = 3$ illustrate this, the relevant part of $\mathcal{F}$ is

$$\tilde{\mathcal{F}} = \begin{bmatrix} \mathbf{F}_2 & \mathbf{F}_1 & \mathbf{F}_0 & & \\ & \mathbf{F}_2 & \mathbf{F}_1 & \mathbf{F}_0 & \\ & & \mathbf{F}_2 & \mathbf{F}_1 & \mathbf{F}_0 \end{bmatrix}. \qquad (7.26)$$

The $NP \times K(2P-1)$ matrix $\tilde{\mathcal{F}}$ consists of NP rows from the large matrix $\mathcal{F}$. The central K columns are the K synthesis vectors, the matrix $\mathbf{F}$ in Equation 2.11. An entry of the $K \times K(2P-1)$ matrix $(\mathbf{F}^T\tilde{\mathcal{F}})$ is cosine of the angle between one frame vector and one of the frame vectors that overlap, i.e. the column vectors of $\tilde{\mathcal{F}}$. For an overlapping frame the $\beta^{(min)}$ property can be defined as

$$\beta^{(min)} = \min_{i,j \neq (P-1)K+i} \beta_{ij} \qquad \text{where} \quad \cos(\beta_{ij}) = [\mathbf{F}^T\tilde{\mathcal{F}}]_{ij}. \tag{7.27}$$

The entries of $([\mathbf{F}^T\tilde{\mathcal{F}}])$, i.e. $\cos\beta_{ij}$, that are used include all angles between different frame vectors except the angle of one frame vector to itself (which of course is zero), this is what the subscript, $i, j \neq (P-1)K+i$, below the min function indicate. The $\beta^{(avg)}$ property is a little bit more difficult, it is the average for all the frame vectors to its closest neighbor. Here we interpret "all the frame vectors" as the K column vectors of $\mathbf{F}$ which are the central K column vectors of $\tilde{\mathcal{F}}$, and it closest neighbor is one of the other columns in $\tilde{\mathcal{F}}$, including translations of the vector itself. With this the average is defined similar to Equation 7.15

$$\beta^{(avg)} = \frac{1}{K}\sum_i \min_{j \neq (P-1)K+i} \beta_{ij}. \tag{7.28}$$

The geometric interpretation of the $\beta^{(mse)}$ property is more hazy for overlapping frames than for block-oriented frames. Nevertheless, the definition in Equation 7.17 shows that it is possible to calculate the $\beta^{(mse)}$ property only using the eigenvalues of the frame operator (and the size of the frame), and this should be possible also for an overlapping frame. A problem is that for the overlapping frame the eigenvalues depends on the value θ. For any given value of θ a range for the sum of the eigenvalues squared exists, it is like Equation 7.18

$$\frac{1}{N} \;\leq\; \frac{\sum_n \lambda_n^2(\theta)}{\left(\sum_n \lambda_n(\theta)\right)^2} \;<\; 1. \tag{7.29}$$

Note that the term $\sum_n \lambda_n$ can be replaced by K in Equation 7.18, but this can not be done for the overlapping frame. A fixed value for the sum of the

eigenvalues exist only if the integral on the unit circle is taken, Equation 7.23. With this information it is natural to suggest the following definition

$$\sin^2(\beta^{(mse)}) = \frac{K}{K-1}\Big(1 - \int_0^1 \frac{\sum_n \lambda_n^2(\theta)}{\big(\sum_n \lambda_n(\theta)\big)^2}\,\mathrm{d}\theta\Big) \tag{7.30}$$

For the frame used in Figure 7.6 this definition gives $\beta^{(mse)} = 36.8$ degrees. This is only marginally larger than the average for the frame vectors to their closest neighbor for the same frame, $\beta^{(avg)} = 36.3$. For some frames the $\beta^{(mse)}$ property may even be smaller than the $\beta^{(avg)}$ property. The geometric interpretation of the $\beta^{(mse)}$ property can not be done for the overlapping frames like it was done for the block-oriented frames. The $\beta^{(mse)}$ property should be regarded as a property derived from the eigenvalues of the frame operator, like the frame bounds are, and in addition a geometric interpretation is possible for the block-oriented frame.

The difference between two overlapping frames, $\mathbf{F}^{(1)}$ and $\mathbf{F}^{(2)}$, can be measured like for block-oriented frames in Equation 7.22. The two frames should have the same values for N and P, but K can be different. Matrix $\mathbf{C}^{(1)}$ and $\mathbf{C}^{(2)}$ are defined as cosine for angles between frame vectors in different frames: $\mathbf{C}^{(1)} = \mathbf{F}^{(1)T}\tilde{\mathcal{F}}^{(2)}$ and $\mathbf{C}^{(2)} = \mathbf{F}^{(2)T}\tilde{\mathcal{F}}^{(1)}$ where $\tilde{\mathcal{F}}^{(i)}$ is like in Equation 7.26. The angles corresponding to $\mathbf{C}^{(1)}$ and $\mathbf{C}^{(2)}$ are denoted $\theta^{(1)}$ and $\theta^{(2)}$. The definition is then

$$\theta^{(avg)} = \frac{1}{K_1 + K_2}\Big(\sum_{i=1}^{K_1} \min_j \theta_{ij}^{(1)} + \sum_{i=1}^{K_2} \min_j \theta_{ij}^{(2)}\Big) \tag{7.31}$$

This is a more complicated definition that if we had just used the $\mathbf{F}$ matrices. The reason for this more complicated measure is that two frames where the difference is just a shift (of N positions) with the measure of Equation 7.31 are equal to each other, and as far as representation capabilities are considered these two frames are equal. Examples of two such frames are

$$\mathbf{F}^{(1)} = \begin{bmatrix} \mathbf{F}_0 \\ \vdots \\ \mathbf{F}_{P-1} \\ \mathbf{0} \end{bmatrix} \qquad \mathbf{F}^{(2)} = \begin{bmatrix} \mathbf{0} \\ \mathbf{F}_0 \\ \vdots \\ \mathbf{F}_{P-1} \end{bmatrix} \tag{7.32}$$

The $r_s^{(max)}$, $r_s^{(avg)}$, $r_s^{(mse)}$ and the A_s properties in Table 7.1 are not relevant for overlapping frames.

7.5 Some examples

In this section we will give a few examples of the use of the introduced properties. The intention is to illustrate what these frame properties tell about the frame. In Subsection 7.5.1 the geometry of the space $\mathbb{R}^N$ is used to argue that what seems like moderately sized values for the $\beta^{(min)}$ and $\beta^{(avg)}$ properties and the $\theta^{(avg)}$ difference measure are actually small, in the meaning that larger values are more likely (for randomly generated frames). Some examples, including some of the frames used for texture classification in Chapter 6, are presented in the following subsections. Even if the frame properties do not help to explain how texture classification works, the properties give some information on the frames.

7.5.1 Geometry of the space $\mathbb{R}^N$

It is difficult to visualize the space $\mathbb{R}^N$. We will not use much space on this subject here, but recommend an interested reader to search more on this topic in the mathematical literature. We will mention only a few facts here, these may be helpful when assessing the numerical value of some of the frame properties. First, some formulas for the surface and volume of the unit ball in $\mathbb{R}^N$ are presented. Then these are used to show that two randomly selected vectors are unlikely to have a small angle between themselves.

A volume in $\mathbb{R}^N$ is an N-dimensional set of elements, the unit volume is the elements within an N-dimensional cube where all sides have length 1. A surface in $\mathbb{R}^N$ is an $(N-1)$-dimensional set of elements, for example the unit cube in $\mathbb{R}^N$ is bounded by $2N$ unit cubes in $\mathbb{R}^{N-1}$ and thus has a surface with "area" $2N$. A ball in $\mathbb{R}^N$ consists of all points/vectors $\mathbf{x} \in \mathbb{R}^N$ such that $\|\mathbf{x}\| \leq r$, where r is the radius of the ball. The unit ball has radius one. The volume of a ball in $\mathbb{R}^N$ is denoted $V_N(r)$, and its surface is $S_N(r)$. Using calculus it can be shown that[3]

[3] In mathematical literature the following formulas, and variants of them, are usually only referred to as "well known". On the web I found one article where the derivation of the formulas is done [5].

$$\begin{aligned} S_N(r) &= \int_0^\pi S_{N-1}(r\sin\theta)\, r\ \mathrm{d}\theta \\ &= S_{N-1}(r)\cdot r\cdot \int_0^\pi \sin^{N-2}\theta\ \mathrm{d}\theta \\ V_N(r) &= \int_0^\pi V_{N-1}(r\sin\theta)\, r\ \sin\theta\ \mathrm{d}\theta \\ &= V_{N-1}(r)\cdot r\cdot \int_0^\pi \sin^{N}\theta\ \mathrm{d}\theta \end{aligned} \tag{7.33}$$

These recursive formulas are valid for $N \geq 2$ and the initial values to use are $S_1(r) = 2$ and $V_1(r) = 2r$. From these formulas it can be derived that

$$S_N(r) = 2\pi r V_{N-2}(r). \tag{7.34}$$

A more direct formula is

$$V_N(r) = \frac{(r\sqrt{\pi})^N}{\Gamma(N/2+1)}, \tag{7.35}$$

where $\Gamma(\cdot)$ is the gamma function. Both the volume and the surface of the unit ball goes towards zero as the dimensionality increase.

Let us look at the probability density function (pdf) for the inner angle between a random vector $\mathbf{x}$ uniformly distributed on the unit ball and another vector. Since $\mathbf{x}$ is uniformly distributed on the unit ball this other vector does not matter, it could be for example another random vector, a frame vector or a unit vector. The pdf, which we can denote $f(\beta)$, can be found for an inner angle β as the part of the surface of the unit ball in $\mathbb{R}^N$ that has an angle β (or $\pi - \beta$) to the $\mathbf{e}_1 = [1, 0, 0, \dots]^T$ unit vector divided by the total surface of the unit ball in $\mathbb{R}^N$:

$$f(\beta) = \frac{2S_{N-1}(\sin\beta)}{S_N(1)} \ \propto\ \sin^{N-2}\beta, \quad 0 \leq \beta \leq \pi/2. \tag{7.36}$$

The pdf $f(\beta)$ is proportional to $\sin^{N-2}\beta$ since $S_N(1)$ is a constant and $S_N(r)$ is proportional to r^{N-1}. From Equation 7.36 we can see that for large N two "random" vectors are unlikely to have a small angle between themselves, in

Size	Signal	$r_1^{(max)}$	$r_1^{(avg)}$	$r_1^{(mse)}$	$r_2^{(max)}$	$r_2^{(avg)}$	$r_2^{(mse)}$
2×4	Gaussian	0.3827	0.1932	0.2228	0.0000	0.0000	0.0000
4×8	Gaussian	0.8660	0.5005	0.5202	0.3437	0.1629	0.1784
8×16	Gaussian	0.8869	0.6937	0.7000	0.6397	0.4461	0.4535
16×32	Gaussian	0.9682	0.8148	0.8167	0.9128	0.6627	0.6657

Size	A	B	A_2	A_3	$\beta^{(min)}$	$\beta^{(avg)}$	$\beta^{(avg2)}$	$\beta^{(mse)}$
2×4	2	2	0.2929	-	45.00	45.00	45.00	54.74
4×8	2	2	0.5000	0.2929	60.00	60.00	62.42	67.79
8×16	2	2	0.6464	0.5000	69.30	69.30	72.22	75.04
16×32	2	2	0.7500	0.6464	75.52	75.52	78.21	79.65

Table 7.2: Properties of tight frames that are the concatenation of two orthonormal bases, the identity matrix and the Haar matrix. The angles are given in degrees.

fact as N goes towards infinity the probability that the angle between two random vectors is smaller than β goes towards zero for all $\beta < \pi/2$. It is not that easy to see how this influence the $\beta^{(min)}$ property, but as will be seen in Subsection 7.5.3, for random frames the $\beta^{(min)}$ property is unlikely to be small. In the example shown in Table 7.3 200 random 16×32 frames were used and none of them had $\beta^{(min)}$ less than 30 degrees.

7.5.2 Tight frames

The first frames which properties we will examine are tight frames that are the concatenation of two orthonormal bases, the identity matrix and the Haar matrix. These frames will have size $N \times 2N$. The properties of different values of N are presented in Table 7.2, it shows how the properties change with increasing value of N. The frame with $N = 2$ is the frame in Figure 7.3 (a) with $\alpha = \pi/4$ and the frame with $N = 4$ is the frame in Figure 7.4 (b).

One interesting thing to note is that for $N = 4^p$ where p is an integer we will have $r_1^{(max)} = \sqrt{1 - 1/N}$, i.e. sine of $\beta^{(min)}$. This means that in these particular cases we are able to find a vector $\mathbf{x}$ such that the angle between this one and the closest frame vector is as large as the angle between a unit vector and a Haar basis vector. From the table we see that the $r_1^{(max)}$ only increase a little bit from the 4×8 frame to the 8×16 frame, the increase is larger from the 8×16 frame to the 16×32 frame.

Since the frames are tight $A = B = K/N = 2$ and with $K = 2N$ we get $\sin \beta^{(mse)} = \sqrt{(N-1)/(N-0.5)}$ according to Equation 7.19. Since the

Property	Signal	$r_1^{(max)}$	$r_1^{(avg)}$	$r_1^{(mse)}$	$r_2^{(max)}$	$r_2^{(avg)}$	$r_2^{(mse)}$
Mean	Gaussian	0.9881	0.8240	0.8262	0.9663	0.6840	0.6881
Max	Gaussian	0.9962	0.8274	0.8298	0.9894	0.6914	0.6957
Min	Gaussian	0.9813	0.8211	0.8232	0.8872	0.6787	0.6827
Std	Gaussian	0.0028	0.0012	0.0012	0.0098	0.0022	0.0022

Property	A	B	A_2	A_3	$\beta^{(min)}$	$\beta^{(avg)}$	$\beta^{(avg2)}$	$\beta^{(mse)}$
Mean	0.2656	4.8908	0.2885	0.1460	44.51	56.29	71.45	75.54
Max	0.4664	6.0005	0.3983	0.2082	53.01	59.28	76.68	76.44
Min	0.0908	4.0488	0.1556	0.0802	32.39	52.28	66.52	74.19
Std	0.0633	0.3722	0.0470	0.0285	3.92	1.33	1.81	0.43

Table 7.3: 200 uniform frames of size 16×32 were randomly generated and the frame properties were calculated for each of these frames. The tables present the mean, maximum value, minimum value, and the standard deviation for each ensemble of frame properties. The angles are given in degrees.

frames are the concatenation of two tight frames the $\beta^{(min)} = \beta^{(avg)}$. The frame vectors will not be clustered so the $\beta^{(avg2)}$ property is not very relevant. As N increase all these β-properties will go towards 90 degrees. For an orthonormal basis the four β-properties will all be 90 degrees.

7.5.3 Randomly generated uniform frames

In Table 7.3 some properties of 16×32 frames are shown. 200 uniform frames were randomly generated, the frame vectors were drawn from a set of normalized random Gaussian vectors, which is a uniform distribution on the surface of a ball in $\mathbb{R}^{16}$. These trials are used as an ensemble of frames. For each frame the frame properties are calculated generating an ensemble for each frame property. The table present the average (statistical mean), the minimum, the maximum and the standard deviation for each of these ensembles. The mean is perhaps the most relevant value as it represent a 'typical' value for the frame property of a random frame.

7.5.4 The 16×32 frame designed for an ECG signal

In Table 7.4 some properties of the 16×32 frame designed for an ECG signal are presented. This frame is shown in Figure 5.2 (a). The frame vectors are quite clustered as can be seen from the relative large value of the upper frame bound, and the relative small values for the $\beta^{(avg)}$, $\beta^{(avg2)}$ and $\beta^{(mse)}$

Size	Signal	$r_1^{(max)}$	$r_1^{(avg)}$	$r_1^{(mse)}$	$r_2^{(max)}$	$r_2^{(avg)}$	$r_2^{(mse)}$
16×32	Gaussian	1.0000	0.8913	0.8938	0.9999	0.7869	0.7925
16×32	ECG	0.9609	0.2583	0.3228	0.8041	0.1471	0.1890

Size	A	B	A_2	A_3	$\beta^{(min)}$	$\beta^{(avg)}$	$\beta^{(avg2)}$	$\beta^{(mse)}$
16×32	0.0000	23.4452	0.0035	0.0002	4.81	17.19	27.92	42.33

Table 7.4: Properties of the 16×32 frame designed for an ECG signal. Note that for the used precision $a = 0.0000$ means $-0.00005 \leq a < 0.00005$ and $a = 1.0000$ means $0.99995 \leq a < 1.00005$.

properties, for example compared to the random frames. We also note that the representation capabilities are of course much better for the ECG signal than for a random signal, but the maximum error can be quite large also for the ECG signal. The small values for A, A_2, and A_3 indicate that the norm of the (optimal) weights may get quite large.

7.5.5 Frames designed for texture classification

Table 7.5 shows some properties of the 25×49 frames designed for the textures in test image (e) in Figure 6.2. The last column, labelled ϕ, in the table shows the angle between the cluster center, the cluster is all the 49 frame vectors and its center is the eigenvector corresponding to the largest eigenvalue of the frame operator, and the vector $\mathbf{v} = [1/5, 1/5, \ldots, 1/5]$. We note that the "center" texture stand out, it is very clustered as can be seen both from its large value of B and its small values of $\beta^{(mse)}$ and $\beta^{(avg2)}$. Also the cluster center is very close to the vector $\mathbf{v} = [1/5, 1/5, \ldots, 1/5]$. It may look like the clusters for the different frames cover very much the same space of the unit ball, and thus it is hard to explain the texture discrimination capabilities.

For some of the textures it may look like the frames designed with $s = 1$ and $s = 5$ are more equal to each other than to the frame designed with $s = 3$. The reason for this is that the frames with $s = 1$ and $s = 5$ were designed using exactly the same training set, while the frames with $s = 3$ were designed using another training set from the same texture. Another observation is that the frames designed for the same texture are more naturally grouped together than the frames designed with the same sparseness factor.

7.5.6 Differences between frames

The differences between two frames can be measured by the $\theta^{(avg)}$ property as defined in Section 7.3.5. We include a table showing the $\theta^{(avg)}$ measure

Texture	s	SNR_t	A	B	$\beta^{(min)}$	$\beta^{(avg)}$	$\beta^{(mse)}$	$\beta^{(avg2)}$	ϕ
left	1	11.67	0.000027	40.29	10.57	16.24	34.54	24.01	4.20
left	3	16.52	0.000014	38.73	11.01	20.69	37.81	26.50	4.99
left	5	19.45	0.000012	39.90	13.32	19.17	35.49	24.73	4.68
upper	1	10.39	0.000044	41.42	11.97	17.09	32.30	22.47	2.88
upper	3	14.38	0.000020	40.96	11.20	20.36	33.46	23.44	3.80
upper	5	17.41	0.000030	41.55	13.57	19.82	32.17	22.36	1.97
lower	1	13.10	0.000011	41.59	8.67	14.47	31.86	21.87	3.64
lower	3	18.24	0.000011	40.43	7.77	18.06	34.49	23.90	4.80
lower	5	21.39	0.000005	41.60	7.97	16.16	31.96	21.71	3.76
right	1	14.12	0.000022	38.78	5.75	15.08	37.38	25.77	6.46
right	3	19.07	0.000045	31.67	7.66	24.55	49.32	35.57	8.41
right	5	21.64	0.000060	38.17	5.72	16.15	38.53	26.55	5.21
center	1	17.26	0.000012	47.11	4.20	6.46	16.04	9.93	1.33
center	3	20.51	0.000010	47.61	3.15	7.47	13.82	8.91	1.12
center	5	22.99	0.000015	47.55	4.61	7.20	14.06	8.79	1.01

Table 7.5: Some properties of the 25×49 frames designed for the textures in test image (e) in Figure 6.2.

Sparseness factor	1/16	2/16	3/16	4/16
1/16	-	10.11	12.31	12.66
2/16	10.11	-	5.31	6.10
3/16	12.31	5.31	-	3.85
4/16	12.66	6.10	3.85	-

Table 7.6: The $\theta^{(avg)}$ difference measure for some 16×32 frames designed for an ECG signal but with varying target sparseness factors. The angles are given in degrees.

for some 16×32 frames designed for an ECG signal but with varying target sparseness factors. The frame with $S = 2/16$ is the frame where the properties are shown in Table 7.4 and the frame in Figure 5.2 (a). From the table we see that the frames are actually quite equal to each other, especially the frame designed for a sparseness factor of 3/16 and 4/16. This indicate that it is probably not necessary to design frames for many different sparseness factors, just a few frames will be enough so that "close to optimal", i.e. almost as good as if the frame was trained for the wanted sparseness factor, sparse representations can be found for all sparseness factors of interest.

Chapter 8

Conclusions and Summary

The aim of this thesis was to investigate the use of frames for sparse signal representations. We wanted to obtain a better understanding of frames through an exhaustive analysis of the frame design method, and by using frames for sparse representation of some signals. During this work the frame design method was further developed and extended. We also searched for applications where this sparse representation will be useful, texture classification was the one focussed on.

The frame design method was formulated using the compact notation of linear algebra. This shed new light on the sparse representation problem using a frame: The representation is linear in representing the reconstructed signal as a linear combination of the frame vectors. And the free variables of the frame are calculated during frame design as a linear combination of the signal samples, with some special cases as exceptions. Also the compact expressions describing the problem made the extensions from block-oriented frames to overlapping frames and from one-dimensional frames to two-dimensional and multi-dimensional frames possible. A third advantage of the compact notation is that it is helpful in the process of developing good algorithms and implementing effective programs. During this work Matlab programs, for frame design and the use of frames in sparse signal representations, were made. The fact that optimal vector selection is an NP-hard problem can not be avoided though, and also the practical sub-optimal algorithms are computationally expensive.

We have presented two sparse representation experiments in this thesis: The first using a one-dimensional ECG signal and the second using images. In both experiments different kind of frames were used and the results were compared, and of course we also compared these results to the results achieved

by thresholding coefficients from commonly used transforms and filter banks. For an ECG signal the frame structures made possible by the new design method increase the sparse representation capabilities of the frames. Using the sparseness factor $S = 1/8$ as an example, we saw that thresholding of the DCT coefficients gave SNR at 23.8 dB, thresholding of the ELT coefficients gave SNR at 24.6 dB, while using a simple block-oriented frame gave SNR at 27.7 dB and a general frame with structure, frame (d) in Section 5.1, gave SNR at 29.2 dB. Increasing the degree of overcompleteness, from $K/N = 2$ to $K/N = 4$, further improved the SNR to 30.4 dB. At lower sparseness factors the performance gap increased, while for larger sparseness factors the performance of the different frame structures was closer to each other. In the sparse representation of an image the results were similar, the new frame structures performed best, but the differences between the frame structures were not as large as for the ECG signal.

The FTCM presented in Chapter 6 is a promising method for texture classification. The main idea is that a frame trained for making sparse representations of a certain class of signals is a model for this signal class. Frames are trained for several textures, one frame for each texture class. A pixel of an image is classified by processing a block around the pixel, the block size is the same as the one used in the training set. Many sparse representations of this test block are found, using each of the frames trained for the texture classes under consideration. Since the frames were trained to minimize the representation error, the tested pixel is assumed to belong to the texture for which the corresponding frame has the smallest representation error.

In this work we tested the FTCM on many test images, with excellent overall performance. The simple, but computationally demanding, classification scheme worked well on all the test images, the number of wrongly classified pixels was from 20% to 60% lower than for the state of the art classification methods presented in [59] for the same 9 test images. In the experiments we also tried many different sets of frame parameters. Which parameter set that will be the best depends on the texture classification task at hand. For the mean of the test images used, the best results were achieved for case where the frame size was $N = 25$ and $K = 100$, and the sparseness factor was $S = s/N$ where $s = 3$, so if any parameter set should be especially recommended this is it.

The frame properties discussed in Chapter 7 are helpful tools for characterizing frames, especially the well known frame bounds are important frame properties. The new suggested properties are helpful for some purposes, for example to measure how "clustered" the frame vectors are. This preliminary work may be a useful base for more extended frame analysis in the future.

8.1 Directions for future research

Based on experience gained during this work, we would suggest some possible directions for future research.

- *Design of tree structured filter bank*
  A tree structured filter bank can be represented as a frame where the parameters of the synthesis system are not linear in the free variables of the frame. This will be a non-linear frame similar to the separable frame in Section 3.6. The frame design method could be extended to also include the possibility to design such frames that can be implemented as tree structured filter banks.

- *Convergence issues*
  Using the hybrid vector selection algorithm in Subsection 4.1.3 the frame design method is guaranteed to converge to a local minimum of the object function. How to find the global minimum is still an open question, perhaps this is impossible to find even for quite small frames.

- *Deciding an appropriate set of frame parameters*
  A large number of choices must be made before starting the frame design method, i.e. optimizing the free variables of the frame to match a certain set of training vectors. The frame size, including the overlap factor, the frame structure which also gives the number of free variables in the frame, the degree of overcompleteness and the target sparseness factor must be decided. Also the initial frame has to be set somehow, it is not in any way obvious that the use of vectors from the training set is the better choice. We have no theory, only some limited experience is gained, that helps us making these choices.

- *Preprocessing of the signal*
  In Section 4.4 it was shown that applying an ELT filter bank on the image before doing a sparse representation using a block-oriented frame gave better results. Also for the FTCM preprocessing of the image blocks were done before a sparse representation was done. This has shown that a suitable preprocessing of the signal often will simplify or improve the following sparse representation. When preprocessing is needed, and what kind of preprocessing that will be best, is still an unanswered question. We think the answer will depend on the application.

- *Extending GMP to ORMP and overlapping frames*
  The GMP algorithm described in Subsection 4.2.3 does global matching

pursuit based on the BMP algorithm. The current version of the GMP algorithm uses block-oriented frames, it could be extended to use overlapping frames as well. It should also be possible to base this algorithm on the OMP or ORMP algorithms instead of the MP algorithm.

- *Obtain a better theoretical understanding of the TFCM*
  It is obvious that the frames for different textures must hold different properties to be able to discriminate. Which properties that are important in the texture classification context are yet not known. The work in Chapter 7 gave only moderate new insight, but we think that this approach may be useful in frame analysis in the future.

- *Obtain more practical experience with the TFCM*
  A supplement to theoretic understanding is an understanding based on experience. This understanding can be increased by doing more experiments: trying more parameter sets, training frames for more textures, and testing these on more test images and also different kinds of test images. Often the interaction between theoretical knowledge and practical knowledge will increase both, theoretical knowledge will help to define relevant experiments, and practical knowledge may give useful hints in the theory development.

- *Two texture classification problem*
  For the two texture classification problem it should be possible to design frames for texture discrimination that represented blocks from one texture class in a good way, and at the same time blocks from the other texture class in a bad way.

- *Explore the use of frames in other applications*
  Frames have been used for signal compression, one example is the multi frame compression scheme in [22]. The overlapping frames has not yet been tried in compression. The fact that the sparse representation capabilities are better for overlapping frames than for block-oriented frames indicate that they may perform better also in a compression application, but this is still not shown. Another application that we only just have started to look into is noise reduction.

Bibliography

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

[2] S. O. Aase, K. Skretting, J. H. Husøy, and K. Engan. Design of signal expansions for sparse representation. In *Proc. ICASSP 2000*, pages 105–108, Istanbul, Turkey, June 2000.

[3] O. K. Al-Shaykh, E. Miloslavsky, T. Nomura, R. Neff, and A. Zakhor. Video compression using matching pursuit. *IEEE Trans. Circuits, Syst. for Video Tech.*, 9(1):123–143, February 1999.

[4] M. R. Banham and J. C. Brailean. A selective update approach to matching pursuits video coding. *IEEE Trans. Circuits, Syst. for Video Tech.*, 7(1):119–129, February 1997.

[5] A. Ben-Israel. The matrix volume, surface integrals and probability distributions, 1998. Available at `http://rutcor.rutgers.edu/pub/rrr/-reports98/22body.ps`.

[6] A.P. Berg and W.B. Mikhael. An efficient structure and algorithm for image representation using nonorthogonal basis images. *IEEE Trans. Circuits, Syst. II: Analog and Digital Signal Processing*, 44(10):818–828, October 1997.

[7] F. Bergeaud and S. Mallat. Matching pursuit of images. In *Proc. IEEE Int. Conf. on Image Proc., ICIP-95*, volume 1, pages 53–56, 1995.

[8] H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger. Frame-theoretic analysis of oversampled filter banks. *IEEE Trans. Signal Processing*, 46(12):3256–3268, December 1998.

[9] L. Bottou and Y. Bengio. Convergence properties of the K-means algorithms. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 585–592. The MIT Press, 1995.

[10] Phil Brodatz. *Textures: A Photographic Album for Artists and Designers.* Dover, NY, 1966.

[11] P. G. Casazza. The art of frame theory. *Taiwanese Journal of Mathematics*, 4(2):129–201, June 2000.

[12] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal of Scientific Computing*, 20(1):33–61, 1998.

[13] Y-T. Chou, W-L. Hwang, and C-L. Huang. Matching pursuit low-bit rate video coding with dictionary optimized by shape-gain vector quantizer, 1999. Available at `http://citeseer.nj.nec.com/394375.html`.

[14] S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado. Forward sequential algorithms for best basis selection. *IEE Proc. Vis. Image Signal Process*, 146(5):235–244, October 1999.

[15] Z. Cvetković and M. Vetterli. Oversampled filter banks. *IEEE Trans. Signal Processing*, 46(5):1245–1255, May 1998.

[16] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Inform. Theory*, 36(5):961–1005, September 1990.

[17] I. Daubechies. *Ten Lectures on Wavelets.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

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

[19] G. Davis, S. Mallat, and M. Avellaneda. Adaptive nonlinear approximations, 1994. Similar to Davis' Ph.D. thesis.

[20] R. J. Duffin and R. C. Schaeffer. A class of nonharmonic Fourier series. *Transactions of the American Mathematical Society*, 72:341–366, 1952.

[21] P. J. Durka, P. J. Ircha, and K. J. Blinowska. Stochastic time-frequency dictionaries for matching pursuit. *IEEE Trans. Signal Processing*, 49(3):507–510, March 2001.

[22] K. Engan. *Frame Based Signal Representation and Compression*. PhD thesis, Norges teknisk-naturvitenskapelige universitet (NTNU)/Høgskolen i Stavanger, September 2000. Available at `http://www.ux.his.no/~kjersti/`.

[23] 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.

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

[25] H. G. Feichtinger and T. Strohmer. *Gabor Analysis and Algorithms*. Birkhäuser, Boston, USA, 1998.

[26] P. A. Freeborough and N. C. Fox. MR image texture analysis applied to the diagnosis and tracking of Alzheimer's disease. *IEEE Transactions on Medical Imaging*, 17(3):475–478, June 1998.

[27] K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition, 1990.

[28] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Norwell, Mass., USA, 1992.

[29] M. Gharavi-Alkhansari. A model for entropy coding in matching pursuit. In *Proc. IEEE Int. Conf. on Image Proc., ICIP-98*, volume 1, pages 778–782, 1998.

[30] M. Gharavi-Alkhansari. A fast globally optimal algorithm for template matching using low-resolution pruning. *IEEE Trans. Image Processing*, 10(4):526–533, April 2001.

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

[32] 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(3):600–616, March 1997.

[33] R. Gribonval. Fast matching pursuit with a multiscale dictionary of Gaussian chirps. *IEEE Trans. Signal Processing*, 49(5):994–1001, May 2001.

[34] C. E. Heil and D. F. Walnut. Continuous and discrete wavelet transforms. *SIAM Review*, 31(4):628–666, December 1989.

[35] C-L. Huang and S-H. Hsu. Road sign interpretation using matching pursuit method. In *Proc. 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 202–206, 2000.

[36] J. H. Husøy, S. O. Aase, K. Skretting, and K. Engan. Design of general block oriented expansions for efficient signal representation. In *Proc. ISCAS'99*, pages III:9–12, Orlando, USA, June 1999.

[37] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[38] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78(9):1464–1480, September 1990.

[39] J. Kovačević, P. L. Dragotti, and V. K. Goyal. Filter bank frame expansions with erasures. *IEEE Trans. Inform. Theory*, 48(6):1439–1450, June 2002.

[40] V. A. Kovalev, F. Kruggel, H. J. Gertz, and D. Y. Cramon. Three-dimensional texture analysis of MRI brain datasets. *IEEE Transactions on Medical Imaging*, 20(5):424–433, May 2001.

[41] H. Li and I. Wollf. Multiscale matching pursuit for image coding. In *Proc. ISSPA 1999*, pages 805–808, Brisbane, Australia, August 1999.

[42] S. M. Luthi. Textural segmentation of digital rock images into bedding units using texture energy and cluster labels. *Math. Geology*, 26(2):181–196, 1994.

[43] A. Mahalanobis and H. Singh. Application of correlation filters for texture recognition. *Applied Optics*, 33(11):2173–2179, 1994.

[44] S. G. Mallat and Z. Zhang. Matching pursuit with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41(12):3397–3415, December 1993.

[45] H. S. Malvar. Extended lapped transforms: Fast algorithms and applications. In *Proc. ICASSP '91*, pages 1797–1800, Toronto, Canada, May 1991.

[46] H. S. Malvar and D. H. Staelin. The LOT: Transform coding without blocking effects. *IEEE Trans. Acoust., Speech, Signal Processing*, 37(4):553–559, April 1989.

[47] J. L. Marroquin and F. Girosi. Some extensions of the K-means algorithm for image segmentation and pattern classification. Technical Report AIM-1390, Massachusetts Institute Of Technology Artificial Intelligence Laboratory, 1993.

[48] G. F. McLean. Vector quantization for texture classification. *IEEE Trans. Systems, Man, Cybernetics*, 23(3):637–649, May/June 1993.

[49] N. R. Mudigonda, R. M. Rangayyan, and J. E. Leo Desautels. Detection of breast masses in mammograms by density slicing and texture flow-field analysis. *IEEE Transactions on Medical Imaging*, 20(12):1215–1227, December 2001.

[50] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24:227–234, April 1995.

[51] R. Neff and A. Zakhor. Very low bit-rate video coding based on matching pursuit. *IEEE Trans. Circuits, Syst. for Video Tech.*, 7(1):158–171, February 1997.

[52] R. Neff and A. Zakhor. Modulus quantization for matching-pursuit video coding. *IEEE Trans. Circuits, Syst. for Video Tech.*, 10(6):895–912, September 2000.

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

[54] C. J. Oliver. Rain forest classification based on SAR texture. *IEEE Trans. on Geoscience and Remote Sensing*, 38(2):1095–1104, March 2000.

[55] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, November 1993. Proc. of Asilomar Conference on Signals Systems and Computers.

[56] S. Pavlopoulos, E. Kyriacou, D. Koutsouris, K. Blekas, A. Stafylopatis, and P. Zoumpoulis. Fuzzy neural network-based texture analysis of ultrasonic images. *IEEE Engineering in Medicine and Biology Magazine*, 19(1):39–47, Jan.-Feb. 2000.

[57] S-C Pei and M-H Yeh. An introduction to discrete finite frames. *IEEE Signal Processing Magazine*, pages 84–96, November 1997.

[58] T. A. Ramstad, S. O. Aase, and J. H. Husøy. *Subband Compression of Images – Principles and Examples*. ELSEVIER Science Publishers BV, North Holland, 1995.

[59] T. Randen and J. H. Husøy. Filtering for texture classification: A comparative study. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(4):291–310, April 1999.

[60] T. Randen and J. H. Husøy. Texture segmentation using filters with optimized energy separation. *IEEE Trans. Image Processing*, 8(4):571–582, April 1999.

[61] B. D. Rao. Signal processing with the sparseness constraint. In *Proc. ICASSP '98*, pages 1861–1864, Seattle, USA, May 1998.

[62] B. D. Rao and K. Kreutz-Delgado. Sparse solutions to linear inverse problems with multiple measrement vectors. In *Proc. DSP Workshop*, Bryce Canyon, Utah, USA, August 1998.

[63] K. Skretting, K. Engan, J. H. Husøy, and S. O. Aase. Sparse representation of images using overlapping frames. In *Proc. 12th Scandinavian Conference on Image Analysis, SCIA 2001*, pages 613–620, Bergen, Norway, June 2001. available at `http://www.ux.his.no/~karlsk/`.

[64] K. Skretting, J. H. Husøy, and S. O. Aase. General design algorithm for sparse frame expansions. Submitted for publication, available at `http://www.ux.his.no/~karlsk/`.

[65] K. Skretting, J. H. Husøy, and S. O. Aase. A simple design of sparse signal representations using overlapping frames. In *Proc. 2nd Int. Symp. on Image and Signal Processing and Analysis, ISPA01*, pages 424–428, Pula, Croatia, June 2001. available at `http://www.ux.his.no/~karlsk/`.

[66] D. Stanhill and Y. Y. Zeevi. Frame analysis of wavelet-type filter banks. *Signal Processing*, 67:125–139, 1998.

[67] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Siam, Philadelphia, PA, USA, 1997.

[68] M. Tuceryan and A. K. Jain. Texture analysis. In C. H. Chen, L. F. Pau, and P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, chapter 2.1, pages 207–248. World Scientific Publishing Co, Singapore, 1998.

[69] M. Unser. Local linear transforms for texture measurements. *Signal Processing*, 11(1):61–79, 1986.

[70] M. Unser and M. Eden. Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering. *IEEE Trans. Systems, Man, Cybernetics*, 20:804–815, 1990.

[71] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[72] M. Vetterli and T. Kalker. Matching pursuit for compression and application to motion compensated video coding. In *Proc. IEEE Int. Conf. on Image Proc., ICIP-94*, volume 1, pages 725–729, 1994.

[73] C. De Vleeschouwer and B. Macq. Subband dictionaries for low-cost matching pursuits of video residues. *IEEE Trans. Circuits, Syst. for Video Tech.*, 9(7):984–993, October 1999.

[74] D. Wu and J. Linders. A new texture approach to discrimination of forest clearcut, canopy, and burned area using airborne C-band SAR. *IEEE Trans. on Geoscience and Remote Sensing*, 37(1):555–563, January 1999.