TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Susanna Minasyan

Parametric Transforms based Adaptive Methods in Signal Processing and Logic Design

Julkaisu 895 • Publication 895

Tampere 2010

Tampereen teknillinen yliopisto. Julkaisu 895
Tampere University of Technology. Publication 895

Susanna Minasyan

Parametric Transforms based Adaptive Methods in Signal Processing and Logic Design

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB222, at Tampere University of Technology, on the 11th of June 2010, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2010

ISBN 978-952-15-2369-4 (printed)
ISBN 978-952-15-2404-2 (PDF)
ISSN 1459-2045

Abstract

The thesis is devoted to adaptive parametric transforms based methods in digital signal processing and logic design. It is organized as an introduction to the topic followed by six original publications presenting the main scientific results.

The introductory part begins with an overview of unified approaches to fast Discrete Orthogonal Transforms (DOTs) such as the well-known Fast Fourier Transform (FFT), Fast Haar Transform (FHT), and Fast Reed-Muller Transform (FRMT). Then, a new unified approach is presented based on introduction of new families of parametric transforms with fast algorithms having a unified structure. This proposed parametric transforms allow not only to generalize many well-known fast DOTs but also to synthesize an infinite number of new ones that can be adapted to given application and given input signal by selecting or defining a proper set of parameters. In particular, one of the proposed fast parametric transforms is the Parametric Haar-like Transform (PHT), which was studied in applications to important practical areas, such as image compression and signal/image denoising. Other examples are parametric fast logic transforms such as the Binary Reed-Muller Haar-like (BRMH), Hybrid Reed-Muller Haar (HybRMH), Ternary Haar-like Transforms (THT), which were successfully studied in applications to logic design.

The introductory part of the thesis continues with a short discussion on image compression techniques. Afterwords, our research aiming to reveal the potential of the proposed class of Haar-like parametric transforms in improving the performance of fixed block transforms in image compression is presented. This has resulted in two new algorithms. The nature of the proposed schemes is such that their performance is at least as good as that of conventional Discrete Cosine Transform (DCT) based schemes.

Thereafter, the application of parametric transforms in signal/image denoising is presented. After a short overview of transform based image denoising methods, two new PHT based algorithms are described. In these algorithms, post-processing steps using PHT are applied in combination with the well-known wavelet thresholding based denoising. The performance of the proposed methods shown a significant improvement in noise reduction compared to the wavelet based techniques.

Finally, the application of parametric transforms in digital logic design is explored. Many problems in analysis, synthesis, testing of digital devices have simpler solutions in spectral than in original domain. Examples of classical spectral transforms widely used in digital logic design and communications are Walsh, Haar, Reed-Muller, arithmetic transforms that gained popularity due to their simple and efficient in terms of space and time fast implementations important for practical applications. In the thesis new binary, ternary as well as hybrid parametric transforms combining some known transforms were introduced. The proposed spectral transforms are signal adaptive in the sense that they are designed as a result of analysis of the signal to be processed. New spectral methods utilizing these transforms were developed in order to improve their performance in compact representation of binary and ternary logic functions. The proposed methods were analyzed experimentally showing good results compared to classical transforms. The proposed transforms are of interest in logic design due to their flexible features such as signal adaptivity, fast implementation and low-complexity.

Acknowledgements

The work presented in this thesis has been carried out at the Department of Signal Processing, Tampere University of Technology, during the years 2004-2009.

First, I would like to express my deepest gratitude to Prof. Jaakko Astola, my thesis supervisor, for the possibility to work in an international environment at the Department of Signal Processing, for encouraging me to work on the topics in this thesis, for his expertise, valuable guidance and support during these years.

I express my gratitude to Prof. Karen Egiazarian, the leader of Transforms and Spectral Techniques research group, for his guidance and valuable advices. It was both pleasure and honor to work in that group. My special thanks to Dr. David Guevorkian for his inspirational guidance, continuous patience and support during this research. I would like to express my appreciation also to Prof. Radomir Stankovic for his advices, collaboration and inspiration. My great gratitude goes also to Prof. Hakob Sarukhanyan from the Institute of Informatics and Automation Problems (IIAP), National Academy of Sciences of Armenia.

I express my acknowledgments to Prof. David Akopian and Prof. Ing. Francescomaria Marino, reviewers of this thesis, for their recommendations and constructive comments.

I am thankful to Pirkko Ruotsalainen, Virve Larmila and Elina Orava for their assistance in practical matters.

My sincere thanks to all my colleagues and friends at the university, especially, to Andriy Bazhyna, German Gomez Herrero, Ekaterina Pogossova, who have shared the office with me, to Robert Bregovic for helping me preparing the thesis for publication, as well as to Dmytro Rusanovskyy, Alessandro Foi, Andrey Norkin, Artem Migukin, Atanas Boev, Stanislav Stankovic and others for making a warm, friendly and nice working atmosphere.

My great gratitude goes to my parents, Silva and Georgy, for everything they did for me. I am thankful to my lovely sisters, Anna and Lilit, for their love, support and continuous belief in me. I express my great gratitude also to my uncle Samvel and my aunt Zhanna for introducing me to the world of science and for their unlimited love I am always surrounded with.

Tampere, May 2010
Susanna Minasyan

Contents

Abbreviations

AWGN	Additive White Gaussian Noise
BF	Boolean Function
BPHT	Binary Parametric Haar-like Transform
CDMA	Code Division Multiple Access
CTF	Customized Thresholding Function
1D/2D	One/Two Dimensional
DD	Decision Diagram
DFT	Discrete Fourier Transform
DOT	Discrete Orthogonal Transform
DCT	Discrete Cosine Transform
DST	Discrete Slant Transform
DHT	Discrete Haar Transform
DWT	Discrete Wavelet Transform
DWHT	Discrete Walsh-Hadamard Transform
DPCM	Differential Pulse Code Modulation
ECG	Electrocardiogram
EZW	Embedded Zerotree Wavelet Encoding
EBCOT	Embedded Block Coding with Optimized Truncation
FDCT	Fast DCT
FFT	Fast Fourier Transform
FFT-LF	Fast Fourier Transform-ordered – L Filter
FHT	Fast Haar Transform
FPRM	Fixed Polarity Reed-Muller
FPRMH	Fixed Polarity Reed-Muller Haar
FWHT	Fast Walsh-Hadamard Transform
GF	Galois Field
GRM	Generalized Reed-Muller
HVS	Human Visual System
HybRMH	Hybrid Reed-Muller Haar
IDFT	Inverse Discrete Fourier Transform

IICS	Iterative Image Compression Scheme
JPEG	Joint Photographic Expert Group
KLT	Karhunen-Loeve transform
LT	Lapped Transform
LTD	Local Transform based Denoising
MBR	Multiple Bases Representation
MSE	Mean Square Error
MTIC	Multiple Transform Image Compression
MVL	Multiple Valued Logic
NPRM	Negative Polarity Reed-Muller
PHT	Parametric Haar-like Transform
PPRM	Positive Polarity Reed-Muller
RM	Reed-Muller
RMH	Reed-Muller Haar
SOP	Sum-of-Products
SPIHT	Set Partitioning in Hierarchical Trees
ST	Slant Transform
THT	Ternary Haar-like Transform
TI	Translation Invariant
VLSI	Very Large Scale Integration
WHT	Walsh-Hadamard Transform

List of Publications

The thesis consists of introductory part and the collection of the following publications, which are referred to as [P1]-[P6].

[P1] S. Minasyan, R. Stankovic, K. Egiazarian, J. Astola, ”Hybrid Reed-Muller Haar Representations of Logic Functions”, *Journal of Multiple-Valued Logic and Soft Computing,* vol.15, pp. 341-359, 2009.

[P2] S. Minasyan, J. Astola, D. Guevorkian, ”On a Class of Parametric Transforms and its Application to Image Compression,” *EURASIP Journal on Advances in Signal Processing*, vol. 2007, 14 pages.

[P3] S. Minasyan, R. Stankovic, J. Astola, “Ternary Haar-like Transform and its Application in Reduction of Spectral Representation of Ternary-valued functions,” *Eurocast 2009 Workshop on Simulation and Formal Methods in Systems Design and Engineering & LNCS*, 2009, pp. 518-529.

[P4] S. Minasyan, J. Astola, R. Stankovic, D. Guevorkian, ”Reduction of the number of coefficients in Reed Muller Haar-like spectrum”, *Int. Workshop on Spectral Methods and Multirate Signal Processing, SMMSP 2007,* Moscow, Russia, Sept., 2007, pp. 69-74.

[P5] S. Minasyan, K. Egiazarian, J. Astola, D. Guevorkian, “Signal Denoising based on Parametric Haar-like Transforms”, *Int. Conf. on Signal Processing and Multimedia Applications, SIGMAP 2006,* Setubal, Portugal, August, 2006, pp. 134-139.

[P6] S. Minasyan, J. Astola, K. Egiazarian, D. Guevorkian, “Parametric Haar-like Transforms in Image Denoising”, *Int. Conf. on Image Processing, ICIP 2006,* Atlanta, USA, October, 2006, pp. 2629-2632.

Chapter 1

Introduction

During the last two decades there were serious advances in the theory and practice of digital signal and image processing resulting in many familiar systems and devices such as digital television, mobile phones, computers, digital cameras, navigation devices, *etc.* In this sense, digital signal and image processing have become an integral part of human daily life. The more they are integrated in our daily life, the tougher are the requirements on them and the more efficient methods for designing such systems and devices need to be developed.

For example, one of the most important digital image processing applications is image compression the aim of which is to represent digital images with as little information (in terms of number of bits) as possible and at the same time to preserve the visual quality of images as high as possible. Analyzing this application, one can notice that in early digital cameras one or two megapixel resolution images were considered to produce enough quality, whereas nowadays even ten megapixel resolution images are considered to be of only moderate quality. At the same time, the number of images taken and processed by people has dramatically increased. All these images need to be stored and/or sent via communication links, often wireless. Even though the storage capacities as well as the communication bandwidths have also significantly increased, it is clear that for efficient handling of the enormous amount of large images, there is a great demand for image compression techniques that would reduce the amount of the information representing images at higher and higher rates.

Similarly, analyzing another important signal and image processing application, the noise removal (or denoising), one can notice that more and more advanced techniques are needed with the growing user experience resulting in higher user expectations from one side and the growing range of signal and image processing devices and systems introducing different kinds and different levels of noise from another side.

One of the most important toolsets widely used in signal and image processing applications, in particular, in signal and image compression and denoising, is based on the classical mathematical theory of spectral techniques. In these techniques, the actual processing of a signal or image is implemented in the spectral domain, to which the original signal or image is transferred using a spectral transformation. This is useful in many cases since some features of the signal or image are better represented in the spectral domain than in the original time or spatial domain. For example, in signal and image denoising, transferring to the spectral domain helps separating a white noise from the signal or image since the signal is highly correlated and is concentrated in low frequency components of the spectrum while the noise being uncorrelated is concentrated mainly in the high frequency components. Similarly, in image compression, transferring to the spectral domain helps separating image details from the principal content. In the cases where the detail can be ignored or weakly presented this means a possibility of presenting the principal content of the image with less components in the frequency domain compared to the original spatial domain representation.

However, to make a spectral technique efficient in one or another application the correct choice of the spectral transformation is very important. In signal/image compression and denoising applications, mainly Discrete Orthogonal Transforms (DOTs) (see, *e.g.* [1], [10], [49]) and Discrete Wavelet Transforms (DWTs) (see, *e.g.* [26],[28],[37],[55]) are used. Among the DOTs, the most popular are the Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Walsh-Hadamard Transform (DWHT), Discrete Haar Transform (DHT), Discrete Slant transform (DST). It should be mentioned that for each input signal or image there is the most optimal DOT. However, using the optimal DOT for each input in practical applications is often impossible. On the other hand, signals or images can be combined in groups represented by stochastic processes. In that case, instead of considering the optimal DOT for each input, a single transform for a group of inputs could be considered. For example, it is known that natural images can be modeled as first order Markov processes with high correlation coefficient ρ, $\rho \in (0.9, 0.95)$, and the optimal DOT, both in image compression and denoising applications, is the well-known Karhunen-Loeve Transform (KLT), which is formed from the eigenvectors of the covariance matrix of this process [10],[49]. Though, in some cases this may be useful, in practice the use of KLT is rather restricted due to computational complexity reasons. Instead, suboptimal transforms such as DCT or WHT are used which can be computed with much lower complexity using well-known fast transform algorithms [68]. These transforms (especially DCT) have long been successfully used in the above mentioned applications. However, a further improvement of the performance of signal and image compression as well as denoising applications is rather difficult to expect using any fixed transform since it has become clear that none of them can be suitable to the large variety of different signals and images processed nowadays. Therefore, recently adaptive spectral methods have gained popularity. This theory is still in the beginning of development but had already given some results. For example, in [65], very efficient image compression method was developed by varying only the size of the DCT transform.

Another important field of application of spectral techniques is digital logic design, which is very valuable in, *e.g.* Boolean function (BF) analysis for classification as well as finding fictive variables, in digital circuit synthesis, *etc.* [6],[34],[44]-[46],[50]-[52],[66],[69],[100]. In this field, the spectral transforms are used also with the aim of reducing the number of nonzero spectral coefficients for input function representation. Most often used transforms are the

Reed-Muller transform and (unnormalized) Haar transform and their different generalizations/extensions [51],[52],[79],[84]. Similarly, to the case of signal and image processing, also in this application one can expect a significant performance improvement by utilizing adaptive spectral methods since none of the fixed transforms can match the characteristics of all kinds of logical functions. The large area of applications of spectral techniques in terms of theory of logic design may be found in [45], [52],[89].

In this thesis, parametric transforms based adaptive spectral methods are investigated in applications to signal/image compression and denoising, as well as logic design. Parametric transforms are described with matrices involving set of parameters. In this way, a parametric transform represents a family of transforms from which the desirable one can be chosen using an appropriate set of parameters. This gives a possibility to adapt the transform features dynamically during processing with the aim to best match up the transform to the given task and the given input characteristics. Another advantage of parametric transforms is the possibility to implement large families of transforms with a unified software/hardware that is efficient for each representative of the family and may be tuned to the desired transform. Parametric transform based adaptive spectral techniques were not studied sufficiently, though they may offer a significant new breakthrough in improving efficiency of many applications.

The aim of the present research is to improve the efficiency and enlarge the scope of different applications of signal processing and logic design by using fast adaptive parametric transforms. With this aim, new parametric transforms and new methods utilizing these transforms in different applications, such as signal/image processing and circuit design were developed. In particular, in this thesis, a fast parametric Haar-like transform that is input adapted and can be designed according to the certain parameters is proposed. For this transform we introduced a methodology to define parameters in such a way that the matrix of the resulting transform involves predefined rows.

The proposed Haar-like transform was efficiently applied in signal/image compression and denoising applications. With respect to logic function processing methods, three approaches were proposed for reduction of the spectral representations of logic functions. The first approach is based on the proposed fast binary Haar-like transform. The second approach is based on the proposed fast hybrid Reed-Muller Haar transform defined for different algebraic structures, in particular, Galois fields of orders 2 and 3. For these transforms fast calculation algorithms based on classical theory of FFT-like algorithms have also been developed. The third approach is based on the synthesis of a ternary transform for processing of ternary logic functions. Transforms and methods that were developed for processing of logic functions can also be useful in solving other problems of logic design and communications.

Chapter 2

Discrete Orthogonal Transforms: Background

In this chapter a short overview on the fast discrete orthogonal transform theory is presented. After a general background on discrete orthogonal transforms presented in Section 2.1, definitions of most widely used transforms with their main relevant properties are given in Sections 2.2 and 2.3. This is followed by a short overview of different unified approaches to fast transform algorithms and new parametric transform synthesis methods presented in Section 2.4.

2.1 BASIC CONCEPTS OF DISCRETE ORTHOGONAL TRANSFORMS

Let $\mathbf{H}_N$ be a unitary $(N\text{x}N)$-matrix, i.e. $\mathbf{H}_N \cdot \mathbf{H}_N^* = \mathbf{H}_N^* \cdot \mathbf{H}_N = \mathbf{I}_N$, where $\mathbf{H}_N^*$ is a conjugate transpose of the matrix $\mathbf{H}_N$. For an $(N\text{x}1)$ input vector $\mathbf{x} = [x_0, x_1, ..., x_{N-1}]^T$ (subscript T denotes the transposition operator), the $(N\text{x}1)$ vector $\mathbf{y} = [y_0, y_1, ..., y_{N-1}]^T$

$$\mathbf{y} = \mathbf{H}_N \cdot \mathbf{x} \tag{2.1.1}$$

is called a Discrete Orthogonal Transform (DOT) of $\mathbf{x}$. The vector $\mathbf{y}$ is also referred to as the spectrum of $\mathbf{x}$. Equation (2.1.1) is, in essence, a representation of an arbitrary vector $\mathbf{y}$ as a linear combination of columns of matrix $\mathbf{H}_N$, which form a set of *basis functions* in a new coordinate system.

The inverse to (2.1.1) DOT is defined as

$$\mathbf{x} = \mathbf{H}_N^{-1} \cdot \mathbf{y} = \mathbf{H}_N^* \cdot \mathbf{y}. \tag{2.1.2}$$

In (2.1.2) the input vector $\mathbf{x}$ is represented as a linear combination of the columns of the matrix $\mathbf{H}_N^*$ with the coefficients being the components of vector $\mathbf{y}$.

Equations (2.1.1) and (2.1.2) corresponding to the one-dimensional (1D) transform can be extended to the two-dimensional (2D) transform. In practice, separable 2D transforms are mostly used where first a 1D transform is applied to each column of the input matrix $\mathbf{X}$ and then the same or another 1D transform is applied to each row of the resulting matrix. Thus, a separable 2-D transform over an $(N \times M)$ input matrix $\mathbf{X}$ is defined as follows:

$$\mathbf{Y} = \mathbf{H}_N \mathbf{X} \mathbf{Q}_M , \tag{2.1.3}$$

where $\mathbf{H}_N$ and $\mathbf{Q}_M$ are unitary matrices of sizes $(N \times N)$ and $(M \times M)$, respectively. In many practical applications, the case where $N = M$ and $\mathbf{Q}_M = \mathbf{H}_N^T$ is used.

The inverse separable 2D transform over an $(N \times M)$ matrix $\mathbf{Y}$ is then given as follows:

$$\mathbf{X} = \mathbf{H}_N^* \mathbf{Y} \mathbf{Q}_M^* . \tag{2.1.4}$$

Let us note that equations (2.1.3) and (2.1.4) essentially mean the representation of the matrix on the left side ($\mathbf{Y}$ and $\mathbf{X}$, respectively) as a linear combination of so called basis images of the corresponding transform, which are formed as products of columns of the left matrix ($\mathbf{H}_N$ and $\mathbf{H}_N^*$) with the rows of the right matrix ($\mathbf{Q}_M$ and $\mathbf{Q}_M^*$, respectively) [49].

Few examples of the most often used (classical) transforms are:

– Discrete Fourier Transform (DFT)
– Discrete Cosine Transform (DCT) of Type-II
– Discrete Walsh-Hadamard Transform (DWHT)
– Discrete Haar Transform (DHT)
– Slant transform (ST)
– Wavelet transforms (WTs)
– Lapped transforms (LTs).

DOTs constitute the base of spectral methods widely used in many applications of digital signal and image processing such as compression, filtering, pattern recognition, communications, *etc.* [49],[53],[90].

In a spectral method, the input signal is first transferred to the spectral domain using a transform. Most often a DOT is performed, which may also be considered as a transferring the input signal to a new coordinate system. This is a universal idea and can be applied to solve a large variety of theoretical and practical problems [1] (such as compression, denoising). Then, a computational procedure is applied in the spectral domain. After that, the inverse transform may be performed to convert the result back into the temporal or spatial domain if it is required by the application.

The DOTs are linear, easy invertible, and possess the *energy conservation* property which means that the energy in the original and spectral domains are the same. This means that for an input signal **x** and its spectrum **y** obtained as the result of a DOT, the following expression known, as the Parseval's relation, is satisfied:

$$\sum_{i=1}^{n} |x_i|^2 = \sum_{i=1}^{n} |y_i|^2 . \tag{2.1.5}$$

Many of the DOTs have also a good *energy compaction* property. This implies that in the transform domain they concentrate most of the energy of an input signal into few transform coefficients. It should also be mentioned that a number of efficient algorithms for fast implementation of many DOTs have been developed. These are some of the reasons why DOTs are attractive to many researchers and why they are widely used.

When applying a spectral method to solve a problem, an immediate question of selecting the most appropriate transform naturally evolves. Each transform has its specific area of applications, which is determined by several factors: the quality of processing, computational complexity, memory demands, *etc*. For a given class of signals or applications a certain transform is optimal. Therefore, the choice of the transform depends on the application and on the class of inputs to be processed or analyzed using the spectral method. For example, in image compression, a typical digital image contains a high degree of redundant data, implying the presence of some correlation between neighboring pixels. Therefore, the main benefit of any transform used in image compression is the removing of redundancy by decorrelating the data in the transform domain, that is, a *compactness* of a transform. In many applications, the theoretically optimal transform is the Karhunen-Loeve transform (KLT) [49]. However, the KLT is an input dependent and computationally very demanding transform as it does not possess a fast algorithm. In practice, signal independent suboptimal transforms allowing fast implementations are used instead of the optimal but input dependent and computationally demanding one like KLT.

Generally, two types of transforms may be considered:

1. *Fixed transforms* with matrices having constant entries.
2. *Parametric transforms* with matrices described in a unified form involving a set of parameters.

Among the large variety of fixed transforms, the so called classical transforms such as DFT, DCT, DWHT and DHT and ST are used most often. One common feature of the listed transforms, making them useful, is that there exist fast algorithms for computation of each of them. Analysis of these fast algorithms shows that all of them can be presented in a unified factorization form which describes a family of transforms, among which the desired one can be chosen using the appropriate values of parameters. In this sense, parametric transforms take an intermediate place between the fixed transforms and KLT. In some applications or in some methods, parameters describing the transform can be fixed once thus making the transform input independent. On the other hand, in some other applications or methods, using parametric transforms offers a possibility to make the transform input-dependent by varying the set of parameters describing the transform. This gives a great opportunity to improve the performance of fixed transform based spectral methods by making use of adaptive spectral methods where the transform can be changed even during processing a single input. Potentially, even KLT may thus be outperformed because KLT is a fixed transform for a fixed input signal. At the same time the computational complexity of a parametric transform will approximately be the same as that of a fixed transform.

2.2 KARHUNEN-LOEVE TRANSFORM

The Karhunen-Loeve Transform (KLT) [49] is an orthogonal transform which produces uncorrelated coefficients from a correlated signal by using the information on signal statistics.

Let $\mathbf{u}$ be an (Nx1) real-valued random vector. The basis vectors of the KLT are given by the normalized eigenvectors of its covariance matrix $\mathbf{R}_u$, such that,

$$\mathbf{R}_u \boldsymbol{\varphi}_k = \lambda_k \boldsymbol{\varphi}_k, \quad 0 \le k \le N-1, \tag{2.2.1}$$

where $\{\lambda_k\}$ and $\{\boldsymbol{\varphi}_k\}$ are the eigenvalues and eigenvectors of $\mathbf{R}_u$, respectively.

The KLT of vector $\mathbf{u}$ is defined as

$$\mathbf{v} = \boldsymbol{\Phi}^T \cdot \mathbf{u} \tag{2.2.2}$$

and the inverse KLT is defined as

$$\mathbf{u} = \boldsymbol{\Phi} \cdot \mathbf{v} = \sum_{k=0}^{N-1} v(k) \boldsymbol{\varphi}_k, \tag{2.2.3}$$

where $\boldsymbol{\varphi}_k$ is the kth column of matrix $\boldsymbol{\Phi}$. The unitary matrix $\boldsymbol{\Phi}^T$ is known as the KLT matrix.

The matrix $\boldsymbol{\Phi}$ reduces $\mathbf{R}_u$ to its diagonal form, that is,

$$\boldsymbol{\Phi}^T \mathbf{R}_u \boldsymbol{\Phi} = \mathbf{D} = diag\{\lambda_k\}, \tag{2.2.4}$$

where $\mathbf{D}$ is the diagonal matrix with eigenvalues located on the main diagonal in decreasing order. All these eigenvalues are always nonnegative and correspond to variances of the transform coefficients. The diagonalization of covariance matrix of the transform coefficients means the decorrelation of input data.

In many cases, signal statistics is unknown. Hence, it is approximated with different statistical models. For example, the statistics of small (say, 8x8) blocks of natural images can be described by a first order Markov process with a high correlation coefficient. The covariance matrix of a first order stationary Markov sequence is given by

$$\mathbf{R} = \begin{bmatrix} 1 & \rho & \rho^2 & \dots & \rho^{N-1} \\ \rho & 1 & \rho & \dots & \rho^{N-2} \\ . & \dots & \dots & & \\ . & & & & \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \dots & 1 \end{bmatrix}, \ |\rho < 1|. \tag{2.2.5}$$

For natural images ρ is between 0.9 and 0.95 meaning that neighboring pixels are on average similar to each other.

Properties of KLT:

1. KLT is a signal-dependent transform designed from the input signal at hand (it needs a priori information of signal statistics which may not always be available).
2. KLT has the highest energy compaction compared to any fixed transform (DCT, DFT, WHT, *etc.*). In this sense, it is the optimal transform.
3. KLT matrix computation needs $O(N^3)$ and KLT itself needs $O(N^2)$ operations. Due to the computational complexity, it is not always feasible to use KLT even if the necessary a priori information is available.

The KLT plays the fundamental role in many digital signal processing applications such as face recognition, feature extraction, ECG signal processing.

2.3 FAST ORTHOGONAL TRANSFORMS

When utilizing a particular spectral transform in a signal processing application, one should take into account the transform's computational complexity including the number of arithmetical operations, that is, additions and multiplications. The need of reduction of computational requirements brought to development of fast algorithms derived for different DOTs. The following subsections describe most widely used DOTs and their fast computational algorithms.

2.3.1 Discrete Fourier transform and fast algorithm

The discrete Fourier transform (DFT) is a complex-valued transform. The direct DFT and the inverse DFT (IDFT) of an N-point complex-valued vector $\mathbf{x} = \{x_0, x_1, \dots, x_{N-1}\}$ are defined by

$$y_m = \sum_{n=0}^{N-1} x_n W_N^{mn}, \quad m = 0, \dots, N-1 \tag{2.3.1}$$

$$x_n = \sum_{m=0}^{N-1} y_m W_N^{-mn}, \quad n = 0, \dots, N-1,$$

where $W_N = e^{-j2\pi/N}$, the N-th root of unity, is called a *twiddle factor*.

The expressions (2.3.1) may be rewritten in a matrix-vector product form as follows

$$\begin{aligned} \mathbf{y} &= \mathbf{F}_N \mathbf{x} \\ \mathbf{x} &= \mathbf{F}_N^{-1} \mathbf{y}, \end{aligned} \tag{2.3.2}$$

where the entries of $(N \times N)$ matrices $\mathbf{F}_N$ and $\mathbf{F}_N^{-1}$ are given by

$$[\mathbf{F}_N]_{nm} = W_N^{nm},$$

$$[\mathbf{F}_N]^{-1} = \frac{1}{N} W_N^{-nm}, \quad n, m = 0, 1, ..., N-1 \tag{2.3.3}$$

with n and m denoting the indices for entries of the matrices $\mathbf{F}_N$, $\mathbf{F}_N^{-1}$.

The DFT is the base of many signal processing and telecommunications algorithms related to spectral analysis, convolution, filtering, signal reconstruction, compression, communications *etc.*[49], [100].

Fast Fourier Transform

The computation of direct transforms is inefficient due to large number of computations required. The direct computation of N-point DFT (2.3.1) requires $O(N^2)$ operations. In practice, the fast Fourier transform (FFT) is used. The first fast algorithm for DFT of order N was developed by Cooley and Tukey in 1965 [29]. The main idea of FFT consists of splitting the calculation of an N-point DFT into DFTs of smaller sizes. That is, the $N = 2^n$-point DFT is decomposed into two (N/2)-point DFTs followed by the multiplications with twiddle factors, and then (N/2) 2-point DFTs. Similar algorithm is applied recursively until the entire DFT is obtained for 2-point DFTs. This brings to significant reduction of computation complexity.

In principle, the FFT decomposition by Cooley and Tukey may be implemented for the input sequence of size N, which is a composite number $N = pq$, by decomposing the sequence into p sequences of size q, then computing q-point DFT for each sequence and multiplying the resulting sequences by *twiddle factors*. The sequences are then reordered and q parallel p-point DFTs are computed.

In general, FFT may also be derived by using matrix factorization. For example, the matrix factorization corresponding to one of the most known FFTs, the Decimation In Time (DIT) FFT of order $N = 2^n$ without normalization factors is presented as (see [31])

$$\mathbf{F}_{2^n} = \left(\prod_{i=0}^{n-1} \left(\mathbf{I}_{2^i} \otimes \mathbf{F}_2 \otimes \mathbf{I}_{2^{n-i-1}} \right) \left(\mathbf{I}_{2^i} \otimes \mathbf{T}_{2^{n-i}} \right) \right) \mathbf{Q}, \tag{2.3.4}$$

where

$$\mathbf{F}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$\mathbf{T}_r = diag\left\{\begin{bmatrix} \mathbf{I}_{r/2} & \mathbf{O}_{r/2} \\ \mathbf{O}_{r/2} & \exp\left(-\frac{2\pi j}{r}k\right) \end{bmatrix}\right\}, \quad 0 \le k < \frac{r}{2}, \; j = \sqrt{-1},$$

$$\mathbf{Q} = \prod_{i=0}^{n-2}\left(\mathbf{I}_{2^{n-2-i}} \otimes \mathbf{P}_{2^{i+2},2}\right),$$

where '$\otimes$' stands for the Kronecker product[1]; $\mathbf{I}_N, \mathbf{O}_N$ are the identity and zero matrix of order N, respectively; $\mathbf{P}_{N,2}$ is the permutation matrix that reorders the input vector into a vector with even and odd indices, that is,

$$\mathbf{y} = \mathbf{P}_{N,2} \cdot (x_0, x_1, ..., x_{N-1})^T = (x_0, x_2, ..., x_{N-2}, x_1, x_3, ..., x_{N-1})^T.$$

With the representation (2.3.4) of matrix $\mathbf{F}_{2^n}$ in the form of the product of sparse matrices $\mathbf{F}^{(i)} = \left(\mathbf{I}_{2^i} \otimes \mathbf{F}_2 \otimes \mathbf{I}_{2^{n-i-1}}\right)\left(\mathbf{I}_{2^i} \otimes \mathbf{T}_{2^{n-i}}\right)$, the DFT

$$\mathbf{y} = \mathbf{F}_{2^n}\mathbf{x}$$

may be computed in n stages where at each stage $i = 1, ..., n$ a computationally simple transform with the sparse matrix $\mathbf{F}^{(n-i)}$ is implemented:

$$\mathbf{x}_0 = \mathbf{Q}\mathbf{x};$$

$$\mathbf{x}_i = \mathbf{F}^{(n-i)} \cdot \mathbf{x}_{i-1}, \quad i = 1, ..., n; \tag{2.3.5}$$

$$\mathbf{y} = \mathbf{x}_n.$$

It can be seen that the structure of matrices $\mathbf{F}^{(n-i)}$ is such that the multiplication of them to an input vector means the implementation of $N/2$ basic operations over pairs of input components $[a, b]^T = \left[x_k^{i-1}, x_{k+2^{i-1}-1}^{i-1}\right]^T$ of the vector $\mathbf{x}_{i-1} = \left[x_0^{i-1}, ..., x_{N-1}^{i-1}\right]^T$ being the input to stage $i = 1, ..., n$. Each basic operation consists of pre-multiplying the component b to a twid-

[1] The Kronecker product of two matrices $\mathbf{A} = [a(m,n)]$ and $\mathbf{B} = [b(k,l)]$ is the block matrix $\mathbf{C} = \mathbf{A} \otimes \mathbf{B} = [a(m,n)\mathbf{B}]$.

dle factor $e^{-j2\pi k/2^i}$ according to the matrix $\left(\mathbf{I}_{2^{n-i}} \otimes \mathbf{T}_{2^i}\right)$ and then implementing a 2-point DFT to the resulting pair according to $\left(\mathbf{I}_{2^{n-i}} \otimes \mathbf{F}_2 \otimes \mathbf{I}_{2^{i-1}}\right)$. That is, the basic operation consists of forming a new pair $[c,d]^T$ such that $c = a + e^{-j2\pi k/2^i} b$ and $d = a - e^{-j2\pi k/2^i} b$. Thus at each stage about $N/2$ multiplications and N addition/subtractions are implemented. Since there are $n = \log N$ stages, the total complexity of the fast transform algorithm (2.3.5) is estimated as $O(N \log N)$ operations.

The FFT algorithm corresponding to the representation (2.3.4) can be mapped into the flowgraph as shown in Figure 2.1 for the case $N = 8$. As in (2.3.5), the Figure 2.1 consists of several stages, each corresponding to multiplication by one sparse matrix in the matrix representation meaning a permutation of input components in order to form corresponding pairs $[a,b]^T = \left[x_k^{i-1}, x_{k+2^{i-1}-1}^{i-1}\right]^T$ followed by $N/2$ basic operations.

It should be noted that there are many versions of FFT algorithms with different computational complexities and structures [38],[71], [85], [94], [96]. Of our particular interest is the so called constant geometry algorithm, which as was shown in [2] is based on a DFT matrix factorization in the form of the product of block-diagonal matrices (presented as direct sums of kernels) and special permutation matrices that do not change from stage to stage. This representation corresponds to a unified representation presented in Section 2.4 as the base for synthesizing new large class of parametric transforms.

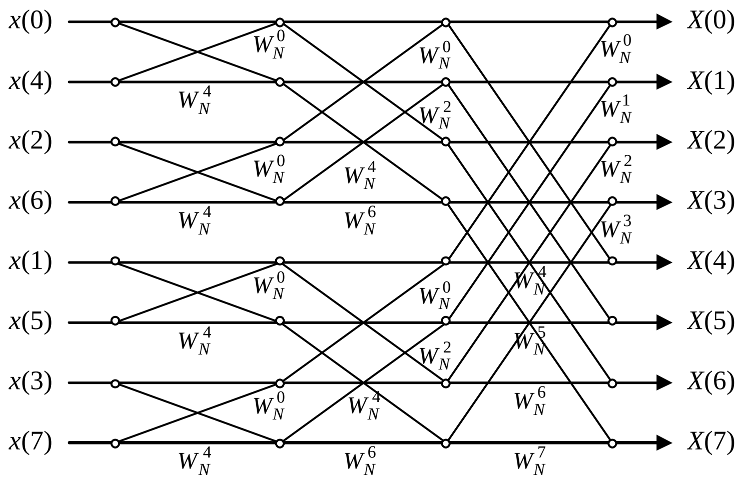


Figure 2.1 FFT for N=8.

The basic building block used in FFT is the operation called *butterfly*, given in Figure 2.2, where $c = a + bW_N^r$ and $d = a - bW_N^r$, $r = 0,\ldots,N-1$.

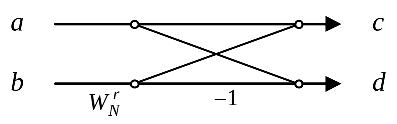


Figure 2.2 Butterfly diagram for FFT.

Properties of FFT (Cooley-Tukey FFT of order $N = 2^n$):

1. $n = \log_2 N$ stages, $N/2$ butterflies per stage
2. $(N/2)\log_2 N$ complex multiplications; $N\log_2 N$ complex additions; totally $O(N\log_2 N)$ operations.

2.3.2 Discrete cosine transform and fast algorithm

The discrete cosine transform (DCT) was first introduced in 1974 by Ahmed et al [9]. Four types of DCT have been defined [47], [72]. Here, we consider the Type II DCT since this is the one used both in digital image compression and in signal/image denoising. In what follows we refer to Type II DCT as DCT. It was shown that DCT transform is very close to the KLT derived from covariance matrix of first order Markov process with high correlation degree ($\rho > 0.9$). The DCT has excellent decorrelation and energy compaction properties [10].

The 1D DCT is an orthogonal transform

$$\mathbf{y} = \mathbf{C}_N \mathbf{x}$$

with the matrix $\mathbf{C}_N$ having the following real-valued entries:

$$c(k,n) = a(n)\frac{2}{\sqrt{N}}\cos\frac{\pi k(2n+1)}{2N}, \quad k,n = 0,...,N-1, \tag{2.3.6}$$

where N is the size of the transform, $\mathbf{x}$ and $\mathbf{y}$ are the $N \times 1$ input and output vectors, respectively, and

$$a(n) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } k = 0 \\ 1, & \text{otherwise.} \end{cases}$$

The DCT matrix of order $N = 8$ is the following

$$\mathbf{C}_8 = \begin{bmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.4619 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}.$$

The 2D direct and inverse DCT for a matrix $\mathbf{X}$ of size $N \times N$ with elements $[\mathbf{X}]_{m,n} = x[m,n]$ is defined as follows:

$$\begin{aligned} y[r,s] &= p(r)p(s)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n]\cos\left[\frac{\pi r(2m+1)}{2M}\right]\cos\left[\frac{\pi s(2n+1)}{2N}\right] \\ x[m,n] &= \sum_{r=0}^{N-1}\sum_{s=0}^{N-1} p(r)p(s)y[r,s]\cos\left[\frac{\pi r(2m+1)}{2M}\right]\cos\left[\frac{\pi s(2n+1)}{2N}\right], \end{aligned} \tag{2.3.7}$$

where $\mathbf{Y}$ is the matrix of the DCT spectral coefficients of matrix $\mathbf{X}$, with elements $[\mathbf{Y}]_{r,s} = y[r,s]$; and $p(r)$ and $p(s)$, $0 \le r,s \le N-1$, are defined as follows:

$$p(r) = \begin{cases} \sqrt{\dfrac{2}{N}}, & 1 \le r \le N-1 \\ \sqrt{\dfrac{1}{N}}, & r = 0. \end{cases}$$

The 2D DCT of $(N \times N)$ matrix $\mathbf{X}$ is a separable transform that may be expressed as:

$$\mathbf{Y} = \mathbf{C}_N \mathbf{X} \mathbf{C}_N^T. \tag{2.3.8}$$

The 64 basis images of size (8x8) for the 2D DCT are illustrated in Figure 2.3 a). Essentially, when implementing a transformation of an image with the DCT, the image is presented as a linear combination of these images.

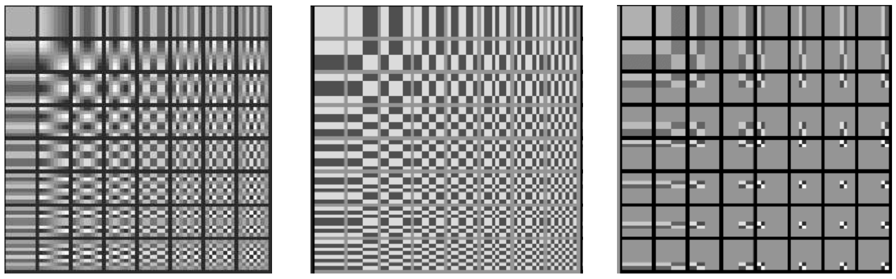


Figure 2.3 Basis images of 2D transforms for N=8: a) DCT, b) WHT, c) DHT.

The DCT is quite useful in many applications, for instance, in compression, denoising, adaptive filtering [93], [97], [99], [100].

Fast DCT

Many fast DCT (FDCT) algorithms are presented in literature [47]. In particular, the FDCT by Chen *et al* [25] is one of the most known ones. It is based on the following matrix representation (without normalization factor):

$$\mathbf{C}_N = \overline{\mathbf{P}_N} \begin{pmatrix} \mathbf{C}_{N/2} & \mathbf{0} \\ \mathbf{0} & \overline{\overline{\mathbf{R}}}_{N/2} \end{pmatrix} \mathbf{B}_N, \qquad (2.3.9)$$

where $\overline{\mathbf{P}_N}$ is a permutation matrix which permutes the even rows in decreasing order in the lower half; $\overline{\overline{\mathbf{R}}}_{N/2}$ is derived from the matrix $\mathbf{R}_N$ by reversing the orders of both the rows and columns of $\mathbf{R}_N$. The entry $r_{i,k}$ $(i,k=1,...,N)$ of the matrix $\mathbf{R}_N$ is

$$r_{i,k} = \cos\frac{(2i+1)(2k+1)\pi}{4N};$$

$\mathbf{B}_N$ is the following butterfly matrix:

$$\mathbf{B}_N = \begin{pmatrix} \mathbf{I}_{N/2} & \bar{\mathbf{I}}_{N/2} \\ \bar{\mathbf{I}}_{N/2} & -\mathbf{I}_{N/2} \end{pmatrix}, \qquad (2.3.10)$$

where $\mathbf{I}_{N/2}$ and $\bar{\mathbf{I}}_{N/2}$ are the identity and counter identity matrices of order $N/2$, respectively.

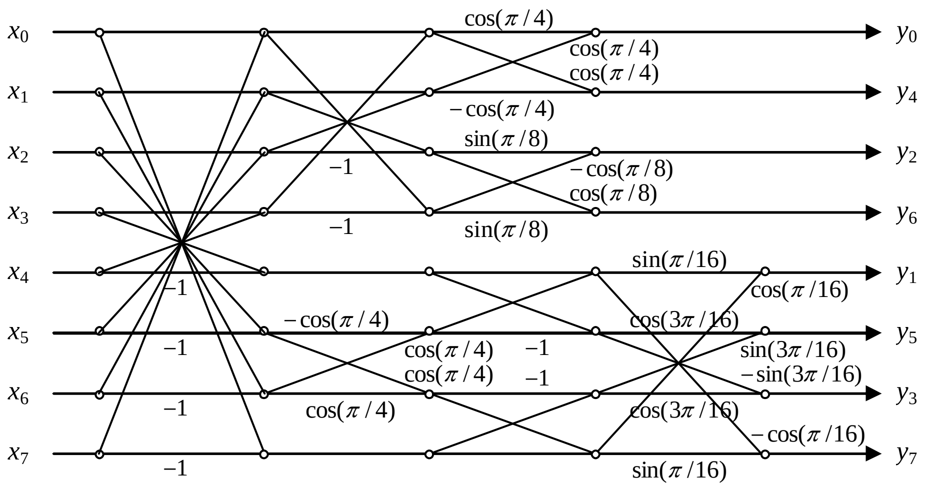


Figure 2.4 Forward FDCT algorithm for *N*=8 by Chen.

The flowgraph of FDCT of order $N = 8$, given in Figure 2.4, is the mapping of FDCT matrix decomposition (2.3.9). The decomposition (2.3.9) may be reduced to the product of block-diagonal and permutation matrices, the flowgraph of which has a regular constant-geometry structure [14], [87].

Properties of FDCT

FDCT flowgraph in Figure 2.4 has the FFT-like structure. The complexity of FDCT is $O(N \log_2 N)$, where N is the power of two [10], [47], [72]. For $N = 8$ the FDCT in Figure 2.4 has 20 multiplications and 26 additions.

2.3.3 Discrete Walsh-Hadamard transform and fast algorithm

The Discrete Walsh-Hadamard Transform (WHT) is a real, symmetric and orthogonal transform of order $N = 2^n, n = 1, 2, \ldots$. The basis functions of the WHT are rectangular and contain only two non-zero values {+1,-1} normalized according to transform size.

The WHT matrix is generated recursively from the core matrix

$$\mathbf{W}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

as

$$\mathbf{W}_1 = [1], \quad \mathbf{W}_N = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{W}_{N/2} & \mathbf{W}_{N/2} \\ \mathbf{W}_{N/2} & -\mathbf{W}_{N/2} \end{bmatrix}. \tag{2.3.11}$$

Another definition of WHT is: $\mathbf{W}_N = \mathbf{W}_2 \otimes \mathbf{W}_{N/2} = \mathbf{W}_2^{\otimes n}$.

The WHT matrix of order $N = 8$ is the following:

$$\mathbf{W}_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{bmatrix}.$$

Depending on the ordering method of Walsh functions, there are different versions of Walsh transform, *e.g.* Walsh-Paley, Rademacher-Walsh transforms [30],[45]. The WHT consists of the complete set of orthogonal Walsh functions arranged in Hadamard ordering [45]. Only the WHT matrix (2.3.11) has a recursive Kronecker product structure. Since the WHT matrix is symmetric, the direct and inverse transforms are identical up to a normalization factor. The basis images of the 2D WHT for $N = 8$ are shown in Figure 2.3 b).

Another definition of WHT matrix $\mathbf{W}_N$ can be derived as follows. For $i, j \in \{0,1,...,N-1\}$, the value of each element $\mathbf{W}_N(i,j)$ of $\mathbf{W}_N$ is defined based on the binary expansions $< i_0, i_1, ..., i_{n-1} >$ and $< j_0, j_1, ..., j_{n-1} >$ of its indices i and j, respectively. This results in

$$\mathbf{W}_N(i,j) = (-1)^{\sum_{m=0}^{n-1} i_m j_m} .$$

The WHT is useful in many different applications of signal/image processing, such as power spectrum analysis, filtering, image coding and enhancement, processing of medical signals (e.g. ECG), communication technologies (CDMA), logical design and analysis [23], [45], [51], [52].

Fast WHT

In practice, fast WHT (FWHT) with a structure similar to that of FFT-like algorithm is utilized. There exist many FWHTs. The matrix representation of a FWHT of order $N = 2^n$ is

$$\mathbf{W}_N = \prod_{i=1}^{n} \left(\mathbf{I}_{2^{i-1}} \otimes \mathbf{W}_2 \otimes \mathbf{I}_{2^{n-i}} \right), \tag{2.3.12}$$

where $\mathbf{I}_r$ is the identity matrix of order r.

The FWHT matrix decomposition (2.3.11) for order $N = 8,$ mapped into a flowgraph is given in Figure 2.5, a). Similarly to the case of FFT, the expression (2.3.12) can be reduced to a decomposition form including block-diagonal and permutation matrices at each iteration [2]. Such representation is important since it is the base of a unified representation for synthesizing a wide class of new parametric transforms (see Section 2.4).

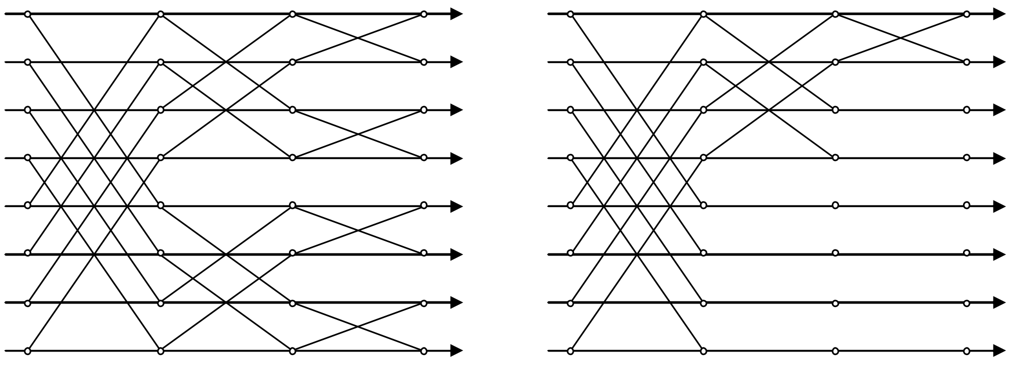

a) b)

Figure 2.5 Fast algorithms of order $N = 8$: a) FWHT, b) FHT.

Properties of FWHT

The FWHT has an FFT-like structure similar to the one in Figure 2.1. The only difference between FFT and FWHT is that the latter consists of only additions. The FWHT of order $N = 2^n$, may be implemented with $n = \log_2 N$ iterations. The computational complexity of FWHT is *O*(*NlogN*). Particularly, for $N = 8$ the FWHT given in Figure 2.5 a) consists of 3 iterations, at each of which the 4 butterfly operations of order 2 are performed.

2.3.4 Discrete Haar transform and fast algorithm

The Haar functions [10] constitute an orthogonal rectangular basis similar to the Walsh functions. The set of Haar basis functions was first introduced by the Hungarian mathematician Alfred Haar. The continuous Haar functions are defined on the time interval $x \in [0,1)$, as follows:

$$h_{0,0}(x) = \frac{1}{\sqrt{N}}, x \in [0,1),$$
$$h_{p,q}(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2}, \frac{q-1}{2^p} \le x \le \frac{q-1/2}{2^p} \\ -2^{p/2}, \frac{q-1/2}{2^p} \le x < \frac{q}{2^p} \\ 0, \ otherwise\ for\ x \in [0,1], \end{cases} \tag{2.3.13}$$

where *p* is a degree of Haar function, q is an order of a Haar function, and

$$N = 2^n,\ 0 \le p \le n-1,$$
$$q = 0,1 \text{ for } p = 0 \quad \text{and} \quad 1 \le q \le 2^p \text{ for } p \ne 0.$$

The discrete Haar functions are defined by sampling $h_{p,q}(x)$ at $N = 2^n$ points $x = \frac{m}{N}$, for $m = 0,...,N-1$. The obtained functions are the rows of the Haar transform matrix $\mathbf{H}_N$. For example, the Haar transform matrix of order 8 is the following:

$$\mathbf{H}_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

The basis images of the 2D DHT for $N = 8$ are illustrated on Figure 2.2 c).

Unlike Walsh transform, the Haar transform matrix doesn't have a recursive Kronecker product structure. It is defined as

$$\mathbf{H}_N = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{H}_{N-1} \otimes [1 \quad 1] \\ \mathbf{I}_{N-1} \otimes [1 \quad -1] \end{bmatrix}. \tag{2.3.14}$$

The Haar transform is a local transform compared to the Walsh transform which is a global transform in the sense that in a Walsh transform matrix all the entries of basis functions are nonzero, while in a Haar transform matrix all but the first two basis functions have zero entries. The nonzero entries are localized in each row. Therefore, the Haar spectral coefficients, besides the first two coefficients, contain a local information about the transformed signal.

Fast Haar Transform

In practice, a fast DHT (FHT) algorithm is often used. The matrix representation corresponding to FHT of order $N = 2^n$, without normalization, may be given as

$$\mathbf{H}_{2^n} = \prod_{r=0}^{n-1} \begin{bmatrix} \mathbf{I}_{2^r} \otimes [1 \quad 1] & \mathbf{O}_{2^n - 2^{r+1}} \\ \mathbf{I}_{2^r} \otimes [1 \quad -1] & \\ \mathbf{O}_{2^n - 2^{r+1}} & \mathbf{I}_{2^n - 2^{r+1}} \end{bmatrix}, \tag{2.3.15}$$

where $\mathbf{O}_r$ and $\mathbf{I}_r$ are the zero and identity matrices of order r, respectively.

The FHT matrix decomposition (2.3.15) can be mapped into the FHT flowgraph illustrated in Figure 2.5 b) for the case N=8.

It was shown in [2] that the DHT matrix may also be decomposed into a product of block-diagonal and permutation matrices, which corresponds to a unified representation used for synthesizing parametric transforms, as discussed in Section 2.4.

Properties of FHT

The low computational and memory requirements of the DHT made it the fastest transform among the classical DOTs. The FHT is useful in signal and image processing, logic design, pattern recognition, communications as well as in many VLSI design applications [5], [6], [10], [44], [45], [50], [51], [52].

The FHT has an FFT-like structure. However, the following features make it different from other FFT-like structures. Unlike the FFT-like algorithms, implementation of FHT requires only $O(N)$ arithmetic operations [10]. The FHT of order $N = 2^n$ may be implemented with $n = \log_2 N$ iterations. One can see that from iteration to iteration only half of the information is passed for processing. Therefore, the number of operations (additions, subtractions) is reduced by a factor of two from stage to stage.

2.3.5 Discrete Slant transform

The $N \times N$ Slant Transform (ST) matrices are defined by the following recursion [10]:

$$\mathbf{S}_n = \frac{1}{\sqrt{2}} \begin{bmatrix} \begin{matrix} 1 & 0 \\ a_n & b_n \end{matrix} & \mathbf{0} & \begin{matrix} 1 & 0 \\ -a_n & b_n \end{matrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{(N/2)-2} & \mathbf{0} & \mathbf{I}_{(N/2)-2} \\ \begin{matrix} 0 & 1 \\ -b_n & a_n \end{matrix} & \mathbf{0} & \begin{matrix} 0 & -1 \\ b_n & a_n \end{matrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{(N/2)-2} & \mathbf{0} & -\mathbf{I}_{(N/2)-2} \end{bmatrix} \begin{bmatrix} \mathbf{S}_{n-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{n-1} \end{bmatrix}, \qquad (2.3.16)$$

where $N = 2^n$, $\mathbf{I}_M$ denotes $M \times M$ identity matrix, and

$$\mathbf{S}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The parameters a_n and b_n are defined as follows

$$a_{n+1} = \left(\frac{3N^2}{4N^2 - 1} \right)^{1/2}, \quad b_{n+1} = \left(\frac{N^2 - 1}{4N^2 - 1} \right)^{1/2}, \quad N = 2^n.$$

The ST is a real and orthogonal transform. It has a piecewise linear basis functions like Walsh transform. ST has one constant basis vector (the first row of a slant matrix is a constant basis vector). Besides, it has one slant basis vector (the second row of a slant matrix is a slant basis vector). The ST has the highest energy compaction amongst the non-sinusoidal fast orthogonal transforms, especially, for images with approximately constant (or uniformly) changing grey levels over a large area. The ST has been used for signal compression, pattern recognition, image watermarking [42], [67] and in Intel's 'Indo' video compression algorithm [10], [54]. Other modified versions of ST may be found in [3], [4],[41],[98].

As an example, the ST matrix of order $N = 4$ is the following:

$$\mathbf{S}_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{bmatrix}.$$

Properties of FST

The ST has a fast Cooley-Tukey type algorithm which can be implemented with $O(N \log N)$ operations on an Nx1 vector [11].

2.4 UNIFIED APPROACHES TO FAST TRANSFORM ALGORITHMS AND PARAMETRIC TRANSFORM FAMILIES

In this section we present a short overview of unified approaches to fast algorithms of discrete orthogonal transforms. A unified approach allows presenting many fast transforms using a single parameterized representation so that a desired transform can be selected by an appropriate parameter selection. Moreover, by varying the parameters of the unified representation many new transforms may be derived. In fact, a proper methodology of selecting parameters means a possibility of utilizing adaptive transform methods wherein the transform features are adapted with respect to the application on hand and with respect to the input signal.

The computational complexities of both the direct DOT (2.1.1) and the inverse DOT (2.1.2) are, in a general case, estimated as $C(DOT)=O(N^2)$ operations. In practical applications, much faster real-time computation is needed. As presented in previous sections, numerous fast algorithms have been developed for different fixed DOTs, *e.g.* the well-known FFT, FDCTs, FWHT, FHT, *etc.* (see e.g. [1], [5],[6],[10],[49]). Analyzing these algorithms one can see that most of them can be described in a unified form. In [1],[2],[5], [7],[8],[10],[53],[82],[83],[90],[92] several unified representations of the fast transform algorithms were described. These representations are based on presenting the transform matrix of order N in the form of a product of sparse matrices $\mathbf{H}^{(j)}$, $j=1,\ldots,m$ and permutation matrices $\mathbf{P}^{(j)}$, $(j=1,\ldots,m+1)$ as:

$$\mathbf{H}_N=\mathbf{P}^{(m+1)}\prod_{j=m}^{1}\mathbf{H}^{(j)}\mathbf{P}^{(j)}. \tag{2.4.1}$$

Transform (2.1.1) with a matrix $\mathbf{H}_N$ presented in the form of (2.4.1) may be computed with a fast algorithm in *m* stages as follows:

$$\begin{aligned}&\mathbf{x}_0=\mathbf{x}\\ &\mathbf{x}_j=\mathbf{H}^{(j)}\cdot\left(\mathbf{P}^{(j)}\mathbf{x}_{j-1}\right),\quad j=1,\ldots,m\\ &\mathbf{y}=\mathbf{P}^{(m+1)}\mathbf{x}_m.\end{aligned} \tag{2.4.2}$$

Computation of the jth stage of the fast transform algorithm (*i.e.* finding $\mathbf{x}_j$ from $\mathbf{x}_{j-1}$) consists of the following: permutation of the components with the matrix $\mathbf{P}^{(j)}$ and implementation of the transform with a sparse matrix $\mathbf{H}^{(j)}$. After *m* stages, the final result $\mathbf{y}$ is obtained by permutation of $\mathbf{x}_m$ with the matrix $\mathbf{P}^{(m+1)}$. The total arithmetic complexity C(FDOT) of the Fast DOT (FDOT) algorithm is defined as the sum of complexities $C\left(\mathbf{H}^{(j)}\right)$, $j=1,\ldots,m,$ of the transforms by sparse matrices:

$$C(\mathrm{FDOT}) = \sum_{j=1}^{m} C\left(\mathbf{H}^{(j)}\right). \tag{2.4.3}$$

In [1],[5],[8],[10],[53],[82],[83],[90],[92] the sparse matrices were represented as Kronecker products of identity matrices of appropriate sizes with one small sized (typically of order 2) kernel. In this way each sparse matrix was constructed using only a single kernel. Though many classical fast transform algorithms, including FFT, FWHT and FDCTs can be represented by using a single kernel within each sparse matrix, however, this is not the case for the FHT.

In order to increase the number of fast transforms presented in a unified form, in [92], sparse layered matrices were considered where each layer is a Kronecker product of identity matrices with a single kernel. This has increased the number of fast transforms presented in a unified form but the representation has become more complex. In addition, all the mentioned representations use only several specific permutation matrices, which also restricts significantly the number of represented fast transforms and, more importantly, is inconvenient from implementation point of view, especially when considering parallel implementations or mapping the algorithms to hardware accelerators.

Another restrictive feature of the above unified fast transform representations is that they all are valid only for transforms of orders $N = r^m$ with r and m being positive nonzero integers (in fact, mainly the case $r = 2$ and sometimes the case $r = 3$ were considered).

In [2],[7] , the above mentioned representations were generalized. First, arbitrary orders of transforms $N = r_1 r_2 \ldots r_m$ were allowed. Second, arbitrary permutation matrices were allowed. Third, the sparse matrices were presented as block diagonal matrices with arbitrary kernels:

$$\mathbf{H}^{(j)} = \bigoplus_{k=0}^{\frac{N}{r_k}-1} \left(\mathbf{V}^{(j,s)}\right), \quad j = 1, \ldots, m, \tag{2.4.4}$$

where "$\oplus$" stands for direct sum of matrices and $\mathbf{V}^{(j,s)}$ are the spectral kernels, which may be different. Since the complexity of a transform of order r_j is upper bounded by r_j^2 operations, the complexity of the fast algorithm presented in the form of (2.4.1), (2.4. 4) is upper bounded by

$$C(\mathrm{FDOT}) = \sum_{j=1}^{m} C(\mathbf{H}^{(j)}) = \sum_{j=1}^{m} \sum_{s=0}^{N_j - 1} C(\mathbf{V}^{(j,s)}) \le N \sum_{j=1}^{m} r_j^2 \,. \tag{2.4.5}$$

In [2],[7],[17] within the class H of transforms that are representable in the form of (2.4.1) and (2.4.4), two families of transforms, namely, the family $\overline{H}$ of uniformly bounded (or Hadamard-like or Fourier-like) fast transforms and the family $\underline{H}$ of unbounded (or Haar-like) transforms were introduced. In the case of uniformly bounded transforms, all the spectral kernels in the representation (2.4.1) and (2.4.4) are unitary matrices with all nonzero entries.

Classical representatives of this family are the DFT and WHT. In the case of unbounded transforms, all the spectral kernels $\mathbf{V}^{(j,s)}$, $j = 1,...,m$, $s = 0,..., \prod_{t=0}^{j-1} r_t - 1$, ($r_0 = 0$), are again unitary with all nonzero entries but the spectral kernels $\mathbf{V}^{(j,s)}$, $j = 1,...,m$, $s = \prod_{t=0}^{j-1} r_t,..., \prod_{t=0}^{j-1} r_t \prod_{l=j+1}^{m-1} r_l - 1$, are all identity matrices. In addition, permutation matrices in (2.4.1) are such that

$$\mathbf{P}^{(j)} = \mathbf{P}_1^{(j)} \oplus \mathbf{I}_{N - \prod_{t=0}^{j-1} r_t},$$

where $\mathbf{P}^{(j)}$ is a permutation matrix of order $\prod_{t=0}^{j-1} r_t$. The classical representative of this family is the FHT.

It is easy to see from the definition that the complexity of uniformly bounded transforms achieves the upper bound of (2.4.5) (since $C\left(\mathbf{V}^{(j,s)}\right) = O\left(r_j^2\right)$ for all $s = 0,..., N_j - 1$) while the complexity of unbounded transforms is linear with respect to the transform order (since $C(\mathbf{V}^{(j,s)}) = O(r_j^2)$ only for $s = 0,1,...,\left(r_0 r_1 ... r_{j-1} - 1\right)$ but $C\left(\mathbf{V}^{(j,s)}\right) = 0$ for $s = \left(r_0 r_1 ... r_{j-1}\right),..., N_j - 1$):

$$C(\mathbf{H}_N) = \begin{cases} O\left(N \sum_{j=1}^{m} r_j^2\right) & \text{if } \mathbf{H}_N \in \overline{H} \\ O(N) & \text{if } \mathbf{H}_N \in \underline{H}. \end{cases} \tag{2.4.6}$$

Figure 2.6 illustrates the generic flowgraph of fast algorithms described by the unified representation (2.4.1), (2.4.4). As can be seen, it has a very regular simple structure where at each of m stages a permutation followed by $N_j = N / r_j$ transforms by spectral kernels of order r_j, $j = 1,...,m$, are implemented. Let us note that all the fast transform flowgraphs presented in Section 2.3 are particular cases of this generic flowgraph. From the definition of the families $\overline{H}$ and $\underline{H}$ the flowgraphs of uniformly bounded transform are of "semirectangular" form (in the sense that there is approximately the same number of nodes at each stage) whereas the flowgraphs of unbounded transforms are of semitriangular form (the number of nodes reduces approximately linearly from stage to stage).

The unified representation (2.4.1), (2.4.4) covers all the classical fast transforms of Section 2.3. This was shown in [2],[7],[17] by finding, for each of these (and some other) transforms, closed form solutions for spectral kernels and permutation matrices that when being used in (2.4.1), (2.4.4) result into the matrix of the corresponding transform.

Therefore, many important fast transforms can be presented in a unified form, which allows designing unified implementations of families of transforms where the desired transform can be selected by specifying the spectral kernels and permutation matrices corresponding to this transform. Here the spectral kernels and permutation matrices become parameters.

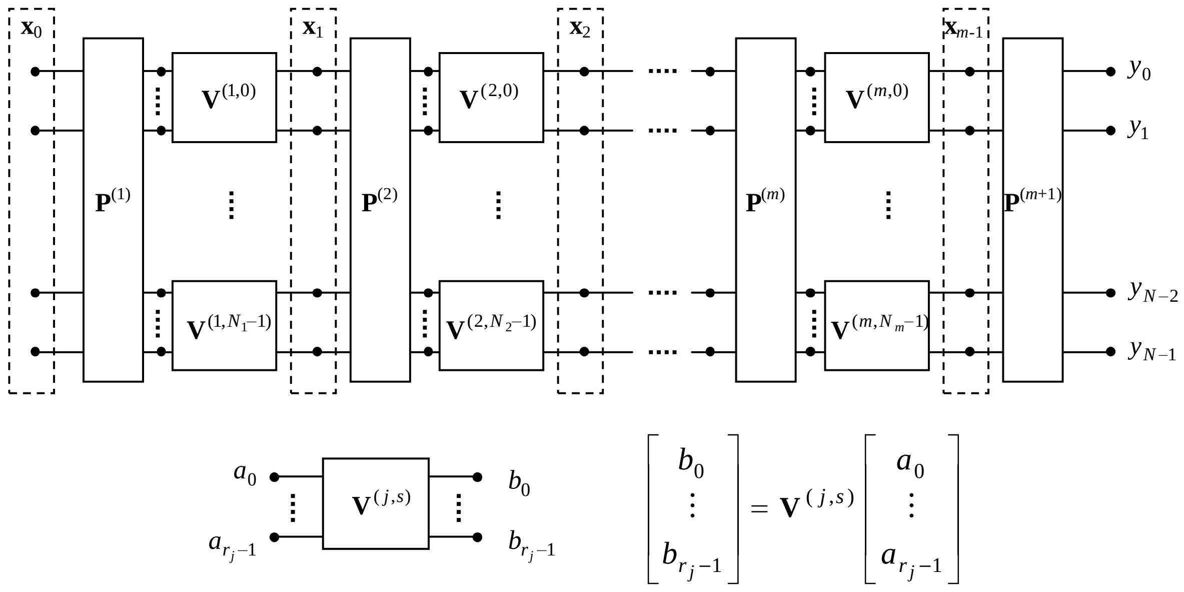


Figure 2.6 Generic flow-graph of the unified fast transform algorithm ($N = r_1 r_2 ... r_m$, $N_j = N / r_j$).

By varying these parameters not only all classical fast transforms may be obtained but also an infinite number of new transforms may be synthesized that a priori will be possible to implement with a fast algorithm of a unified structure. Let us note that the resulting transform implemented by a fast algorithm of the form (2.4.1), (2.4.4) is unitary provided all the spectral kernels in (2.4.4) are unitary.

Several generalizations of the unified fast transform representation (2.4.1), (2.4.4) were also proposed by allowing arbitrary functions to be mapped to the nodes of the flowgraph corresponding to the fast algorithm (see Figure 2.6) instead of only the linear transforms with spectral kernels. In particular, in [20], replacing the linear transforms by sorting operations, a family of so called FFT-LF filters was synthesized that combines good noise reduction properties of linear and non-linear filters. Another extension of the representation (2.4.1), (2.4.4) was proposed in [17], where binary transforms were used in the nodes of the flowgraph. This way a new family of fast binary polynomial transforms, including the classical conjuctive (Reed-Muller) transform, were synthesized.

The unified representation (2.4.1), (2.4.4) is very convenient from implementation point of view as it was shown in [2],[7],[17],[18],[20], [57],[58],[60]. However, high efficiency is achieved only for transform orders $N = r_1 ... r_m$ with m being large. Even though the composite number can formally be an arbitrary positive integer, the efficiency of the corresponding fast algorithm is not high for numbers presented with a smaller number of factors. In particular, in an extreme case where N is a prime number the corresponding "fast" algorithm becomes, in fact, the direct matrix-vector multiplication method with the computational complexity of $O(N^2)$.

In our early work [57] the representation (2.4.1), (2.4.4) was modified so that fast transform algorithms of arbitrary order N are introduced with complexity that does not depend on the number of factors of N. In this representation, the sparse matrices are presented as direct sum of spectral kernels of order two. If N is odd, in addition to spectral kernels of order two, the identity matrix of order one is involved into the direct sum. That is, the sparse matrices involved in (2.4.1) are presented in the form:

$$\mathbf{H}^{(j)} = \left(\bigoplus_{s=0}^{k} \mathbf{V}^{(j,s)} \right) \oplus \mathbf{I}_{1-N \bmod 2} \oplus \left(\bigoplus_{s=k+1}^{\lceil N/2 \rceil - 1} \mathbf{V}^{(j,s)} \right), \quad j = 1, ..., m, \tag{2.4.7}$$

where k is a parameter that may arbitrarily be selected within the range $k \in \{0, 1, ..., \lceil N/2 \rceil - 1\}$, $\mathbf{V}^{(j,s)}$ are (2x2) matrices called *spectral kernels,* $\mathbf{I}_p$ is either an identity matrix of order 1 if $p = 1$ or an empty matrix if $p = 0$, and the sign $\lceil a \rceil$ stands for the smallest integer larger or equal to *a*. Note that now the number of stages in the fast transform flowgraph does not have a direct relation to the order of transform N.

The class of transforms representable in the form of (2.4.1), (2.4.7) is denoted by Ω. Since the matrix $\mathbf{H}^{(j)}, j = 1, ..., m,$ contains at most $4\lceil N/2 \rceil \approx 2N$ nonzero entries, the complexity of the corresponding fast algorithm is estimated as $O(mN)$ operations at the most instead of $O(N^2)$ in the direct method. Thus, the transforms from Ω possess fast algorithms.

Fast transform algorithms of the form (2.4.1), (2.4.7) may nicely be presented by the flowgraph, generically illustrated in Figure 2.6 with only kernels of order two (or one) being used, that is, in this case, nodes represent simple *2*-point discrete orthogonal transforms or “butterfly” operations implying the multiplication of a *2x2* unitary matrix with a 2-point vector. Recall that the general form of an orthogonal *2x2* matrix is

$$\mathbf{V} = \begin{bmatrix} u & v \\ v & -u \end{bmatrix} \tag{2.4.8}$$

where $u^2 + v^2 = 1$ and the “minus” sign may float to each one among the four entries.

Similarly to the case of the class H also within the class Ω the family $\overline{\Omega}$ of uniformly bounded or Hadamard-like orthogonal transforms and the family $\underline{\Omega}$ of unbounded transforms are introduced. For the transforms from $\underline{\Omega}$ all the spectral kernels are unitary with all nonzero entries. The classical WHT and DFT of order $N = 2^m$ belong to the family $\overline{\Omega}$ of Hadamard-like transforms. An example of a new Hadamard-like fast transform flow-graph of order *N*=11 is shown in Figure 2.7.

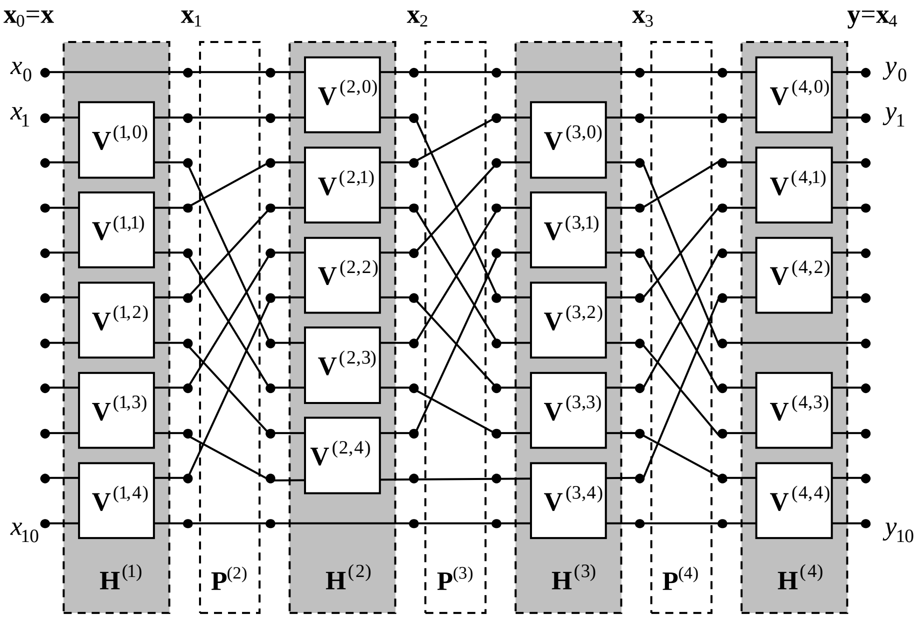
Note that for transforms from $\overline{\Omega}$, each matrix $H^{(j)}, j=1,...,m,$ contains exactly $4\lceil N/2 \rceil \approx 2N$ nonzero entries. Therefore, for transforms from $\overline{\Omega}$ the complexity of the corresponding fast algorithm is estimated as $O(mN)$.

Unbounded or Haar-like transforms within the class Ω are defined so that they are representable in the form of (2.4.1), (2.4.7) where all spectral kernels are unitary and:

- all entries of the spectral kernels $\mathbf{V}^{(j,s)}$, $j=1,...,m,$ are nonzero

  for $s=0,...,N_j-1$, where $N_j = \overbrace{\left\lceil \left\lceil \lceil N/2 \rceil /2 \right\rceil / \cdots /2 \right\rceil}^{j\ times}$
- $\mathbf{V}^{(j,s)} = \mathbf{I}_2$ for $s=N_j,...,\lceil N/2 \rceil -1$, $j=1,...,m$
- $\mathbf{P}^{(j)} = \mathbf{P}_1^{(j)} \oplus \mathbf{I}_{N-N_j}$, where $\mathbf{P}_1^{(j)}$ is a permutation matrix of order N_j.

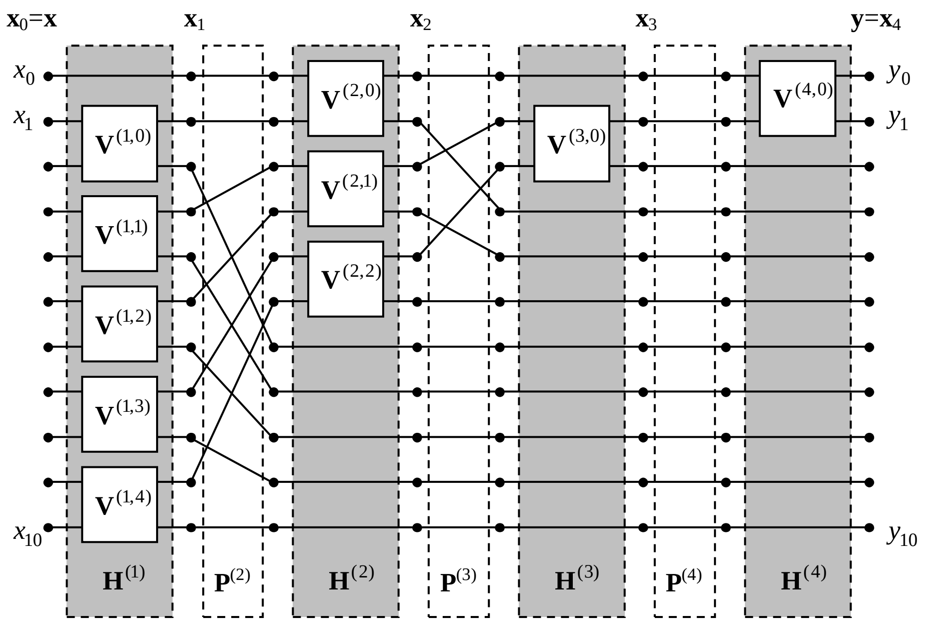
The Haar transform is a classical representative of $\underline{\Omega}$. An example of a new Haar-like fast transform flowgraph of order N=11 is shown in Figure 2.8. It should be noted that the Haar-like (and Hadamard-like) transform of any order may be designed in a similar way.



$$\mathbf{H}^{(j)} = \mathbf{I}_1 \oplus \bigoplus_{s=0}^{4} \mathbf{V}^{(i,j)}, j=1,4,\ \mathbf{H}^{(2)} = \left(\bigoplus_{s=0}^{4} \mathbf{V}^{(2,s)} \right) \oplus \mathbf{I}_1,\ \mathbf{P}^{(1)} = \mathbf{P}^{(5)} = \mathbf{I}_{11}$$

$$\mathbf{H}^{(4)} = \bigoplus_{s=0}^{4} \mathbf{V}^{(4,s)} \oplus \mathbf{I}_1 \bigoplus_{s=0}^{4} \mathbf{V}^{(4,s)},\ \mathbf{P}^{(2)} = \mathbf{P}^{(4)} = \mathbf{I}_1 \oplus \mathbf{P}^{sh}(10),\ \mathbf{P}^{(3)} = \mathbf{P}^{sh}(10) \oplus \mathbf{I}_1$$

Figure 2.7 Fast Hadamard-like transform, N=11.



$$\mathbf{H}^{(1)}=\mathbf{I}_1\oplus\bigoplus_{s=0}^{4}\mathbf{V}^{(1,s)},\ \mathbf{H}^{(2)}=\left(\bigoplus_{s=0}^{2}\mathbf{V}^{(2,s)}\right)\oplus\mathbf{I}_5,\mathbf{H}^{(3)}=\mathbf{I}_1\oplus\mathbf{V}^{(3,0)}\oplus\ \mathbf{I}_8,$$

$$\mathbf{H}^{(4)}=\mathbf{V}^{(4,0)}\oplus\mathbf{I}_9,\ \mathbf{P}^{(1)}=\mathbf{P}^{(4)}=\mathbf{P}^{(5)}=\mathbf{I}_{11},\ \mathbf{P}^{(2)}=\mathbf{I}_1\oplus\mathbf{P}^{sh}(10),\ \mathbf{P}^{(3)}=\mathbf{P}^{sh}(6)\oplus\mathbf{I}_5$$

Figure 2.8 Fast Haar-like transform, N=11.

Note that for transforms from $\underline{\Omega}$, the matrix $H^{(j)}, j=1,...,m,$ contains only $4N_j=N/2^{j-2}$ nonzero entries. Therefore, the complexity of the corresponding fast algorithm is estimated as $O(N)$. Thus, the transforms from $\underline{\Omega}$ possess fast algorithms, which are even faster than those for the family $\overline{\Omega}$, for which the complexity is $O(mN)$.

This can also be noted from the structures of the flow-graphs. While the flowgraphs of Hadamard-like transforms have a “semirectangular” structure (equal number of nodes or butterflies at each stage), the flowgraphs of Haar-like transforms have “semitriangular” structure (approximately twice reduced number of nodes from one stage to the next). These two structures were utilized in designing new Haar-like and Hadamard-like transforms [57].

Let us note that that the number of stages m is an independent parameter. For most of classical transforms from $\overline{\Omega}$, $m=\log N$ which is a logical but not a compulsory choice.

Let us also note that the complexity of the fast algorithms described by (2.4.1) and (2.4.7) does not depend on the number of factors of N or any of its other properties rather than the value.

Concluding this section let us consider the case of a class of transforms of order $N=2^m$ where all the permutation matrices are fixed and only orthogonal kernels of order 2 are used.

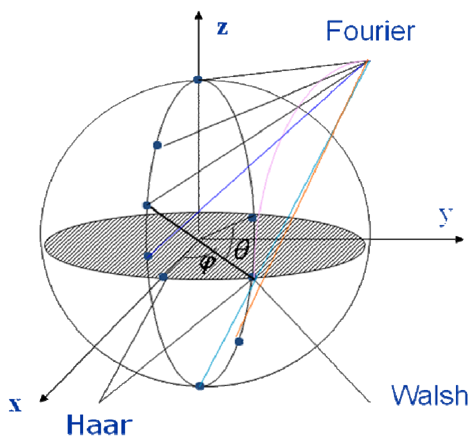


Figure 2.9. The geometrical place of all kernels of order 2.

Each such kernel has the following form:

$$\mathbf{V} = \begin{bmatrix} \cos\varphi & e^{j\theta}\sin\varphi \\ \sin\varphi & -e^{j\theta}\cos\varphi \end{bmatrix}, \quad j=\sqrt{-1},$$

where φ and θ are parameters. Varying these parameters one can synthesize different bases. The kernel of this type (see [32], [82], [83]) describes a rotation by angle φ in the horizontal plane and by angle θ in the vertical plane. Therefore, there is a one-to-one correspondence between the orthogonal spectral kernels of order 2 and the points on the sphere of radius one (see Figure 2.9). The classical transforms correspond to only few points on the sphere. On the other hand, the whole surface of the sphere can be utilized to synthesize new transforms.

Thus, the representation (2.4.1), (2.4.7) gives an important possibility of synthesizing an infinite number of new fast transforms. In order to make a practical benefit out of this possibility, there is a need for developing efficient methods of parameter selection in order to synthesize fast transforms with desired features. In publication [P1], a method (first introduced in our early work [57], [58]) is described for synthesizing fast transforms with predefined basis functions. More precisely, an algorithm is proposed that given a set of orthogonal $(1\times N)$-vectors $\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_k$, $k<N$, produces an orthogonal matrix $\mathbf{H}_N$ such that it has the vectors $\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_k$ in its first k rows and it can be presented in the form of (2.4.1), (2.4.7). Moreover, one can select whether the corresponding transform is Haar-like or Hadamard-like. The vectors $\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_k$ are called generating vectors. In our parametric transform based algorithms for signal/image compression or denoising, we mainly utilize the case where only a single generating vector is used to synthesize a single fast transform.

Chapter 3

Applications of transforms in digital image compression

Image compression is one of the classical problems related to compact representation of images. In computer systems, digital images represented in terms of pixels require a huge amount of storage. Typically, digital images are redundant. The aim of an image compression technique is to remove the redundancy in an image, that is, to represent the image data more compactly before storage or transmission. Traditional image compression methods are based on spectral techniques which have been one of the main tools in signal analysis and are the base of many signal compression algorithms. This chapter summarizes the transform based image coders.

3.1 IMAGE COMPRESSION TECHNIQUES BASED ON FIXED TRANSFORMS

The goal of image compression is to reduce the redundancy of an image data in order to be able to store or transmit it more efficiently. The compression methods are divided into lossless and lossy ones. In lossless compression the reconstructed image is an exact copy of the original source image. Normally, only a small amount of compression may be achieved by a loss less compression method. Such methods are useful in, for example, medical imaging, astronomical image processing *etc*, where any visible distortion is unacceptable. In lossy compression methods, the redundant information is discarded as much as possible while still keeping acceptable visual quality of the image. This enables a much higher compression. At the same time, no visible artifacts may be noticed under normal viewing conditions (visually lossless).

Early image compression schemes use predictive coding. In predictive coding, existing information is used to predict future values, and only the difference is coded. Differential Pulse Code Modulation (DPCM) is one particular example of predictive coding. On the other hand, transform coding first transfers an image from its spatial domain representation to a spectral domain representation using a well-known transform, and then codes the transformed coefficients. This method provides more data compression compared to predictive methods at the expense of more computation.

The traditional transform based lossy image compression/decompression scheme is presented in Figure 3.1.

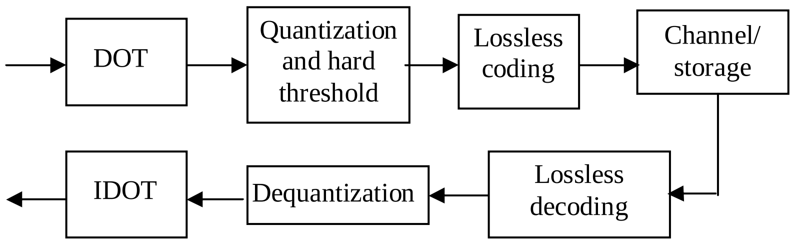


Figure 3.1. Conventional simple image coder/decoder.

In this diagram, the input, which may be a whole image or one of its blocks, is first transferred into a spectral domain. In principle, any transform *e.g.* fixed DOT (block or global) or a parametric transform may be used. A properly selected DOT concentrates most of the image energy in few spectral coefficients only, that is, transferring an original image into a spectral domain results in energy compaction or concentration of low-frequency data in the transform domain. The quantization process applied after the transform uses this phenomenon. It carefully quantizes the significant (low-frequency) information and removes the remaining (high-frequency) coefficients while preserving main content of the image. At this step information loss takes place, which is the source of a distortion in the image after it is reconstructed. This step brings to bit-rate reduction. At the next step, the quantized spectral coefficients are compressed using any lossless coder. This leads to significantly higher compression rates as compared to using the lossless coder directly to the image in the spatial domain. The coded result is either stored or sent via a channel and is then reconstructed in the decoder, which has the reverse order of inverse operations compared to the encoder.

The first image compression systems were based on splitting images into fixed-size blocks and applying a fixed block transform to each block. This leads to occurrence of the blocking artifacts caused by mismatches between neighboring blocks. One of the popular DOTs in image compression is the DCT, which is included in many standards, such as the JPEG image compression standard [68], [97], and which offers acceptable compression gain for not very high compression rates. Image compression using discrete wavelet transform (DWT) gained special attention due to the good decorrelation and localization properties of DWT. The research in this direction brought to the JPEG2000 image compression standard [68], [80]. It was shown that the DWT based methods provide better quality than other fixed transform based methods for high compression ratios. However, the use of JPEG2000 in modern systems is limited. This is mainly due to the complexity of DWT which is applied to the large regions or to the whole image. Unlike to JPEG, this brings not only to big memory demand but also to use of huge amount of computational resources. Due to this, the development of improved image compression methods in a block-based manner is still an actual task. Moreover, the block transform based methods still have a potential of being improved by making them more adaptive to image content.

The following subsections give a short overview on image compression techniques based on DCT and wavelet based compression.

3.1.1 The DCT based approaches

DCT based compression schemes are widespread and are the bases of many compression standards. The most well-known compression standard based on the DCT is the JPEG still image compression standard [68].

An important point of the JPEG is that it specifies only the decoder, thus allowing for possible improvements of the encoder. The JPEG standard consists of four modes: sequential or baseline JPEG; progressive encoding; hierarchical encoding (pyramidal encoder); lossless encoding (does not use the DCT, and it is based on predictive encoding).

The transform coding for the baseline JPEG standard is schematically presented in Figure 3.2. The input is assumed to be 8 bits (or 12 bits). Colors are treated separately. The JPEG is operating in a block-wise manner where the image is split into small non-overlapping blocks and the DCT is applied to each of these blocks. DCT concentrates most of the energy into few coefficients, with the largest one being the DC coefficient located at the top-left corner of the transformed image block.

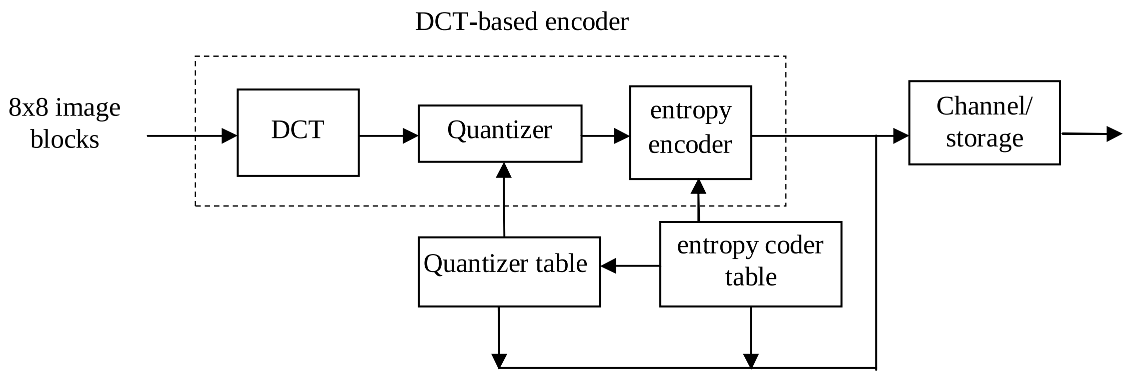


Figure 3.2. Transform encoding in JPEG.

After the DCT transform the quantization is performed according to a quantization table (where each entry is an integer from {1,…,255}). At this step, the DCT coefficients are divided by quantization table entries, and, then, are rounded towards the nearest integer. The quantization tables which are based on visual experiments can also be specified by a user. After the quantization, 2D image blocks are rearranged into 1D vectors according to a zigzag scanning order. The DC coefficients are differentially encoded, that is, differences

$\Delta_l = DC_l - DC_{l-1}$ are coded to remove some of the correlations between DC coefficients DC_l and DC_{l-1} of consecutive blocks *l* and *l*-1, respectively. Finally, the entropy coding based on Huffman coding with a table specified by a user (or default tables) is performed. JPEG decoder operates in the inverse order.

The JPEG image compression method has the following advantages: high efficiency in compression of textural regions of images and the possibility of using the special quantization tables that allow imitating and regulating the Human Visual System (HVS) properties. Besides, JPEG allows also lossless compression which is important in some applications (e.g. medical image coding, *etc*.). Disadvantages of JPEG are blocking artifacts in reconstructed images and ineffective compression of edges and details in images.

In literature, various modifications of the JPEG approach have been developed [21],[70],[99]. In one of them [21] authors proposed the optimization of the quantization table based on HVS for a class of images and different viewing conditions. Another DCT based method was presented in [99] where the embedded zerotree (wavelet-like) coding of DCT coefficients have been performed. However, the drawback of this method is that due to the algorithm structure it requires the same memory requirements as DWT based methods.

In [65], a block-based DCT image compression using 32×32 block is proposed that outperforms JPEG 2000 by up to 1.9dB. This method divides the quantized DCT coefficients into bit-planes. Then, the bits are coded according to probability models considering the correlation between the neighboring block coefficients as well as between values of corresponding coefficients of neighboring blocks. Finally, a DCT based post-processing or filtering is applied in order to remove blocking artifacts.

In the next section we will shortly describe the wavelet based methods where the blocking artifact is not present.

3.1.2 Wavelet based approaches

Starting from mid 1990s wavelet transforms have become very popular, in particular in image compression application. A variety of wavelet based compression techniques have been developed. The most popular examples are EZW [81], SPIHT [76], JPEG2000 [80], EBCOT [86], [80]. These techniques are briefly overviewed below.

EZW (Embedded Zerotree Wavelet Encoding)

The wavelet transform allows a multiresolution analysis of images, meaning different representations of the same image with many levels of detail. The EZW algorithm [81] was the first algorithm that utilized the multiresolution property of a wavelet transform, resulting in a compact representation of significant coefficients. The main idea of EZW is based on the following assumption: if a wavelet coefficient at a coarser level is insignificant with respect to some threshold, then all the wavelet coefficients of the same orientation in the same spatial location at the finer resolution, will be most likely insignificant with respect to that threshold. In result, a set of insignificant wavelet coefficients called a zero-tree is obtained. Based on that zero-tree, one can get a very compact representation of significant coefficients (significance map). In EZW the bits in the output bit stream are generated in order of importance, resulting in a fully embedded code. Thus, the encoder may terminate the encoding at any point, after achieving a target bit rate.

SPIHT (Set Partitioning in Hierarchical Trees Algorithm)

The SPIHT algorithm was proposed by Said and Pearlman [76] and is considered as an improved version of the EZW algorithm. SPIHT uses the same concepts as EZW, that is, the coding of significant wavelet coefficients with respect to a given threshold and then, successive improvement in selecting the significant coefficients. In addition, the following three concepts are used in SPIHT: partial ordering by magnitude of the transformed coefficients with a set partitioning sorting algorithm; ordered bitplane transmission of refinement bits; and exploitation of self-similarity of the image wavelet transform across different resolution levels of an image. As in EZW, the set partitioning rule is dividing the coefficients into the set of significant and insignificant subsets aiming in obtaining a large amount of insignificant coefficients. It was shown that the SPIHT algorithm significantly outperforms the EZW.

***JPEG* 2000**

The development of image compression techniques brought us to JPEG 2000 that was accepted as a standard [80] thought not widely used by web browsers. The Figure 3.3 illustrates the main steps of JPEG 2000.

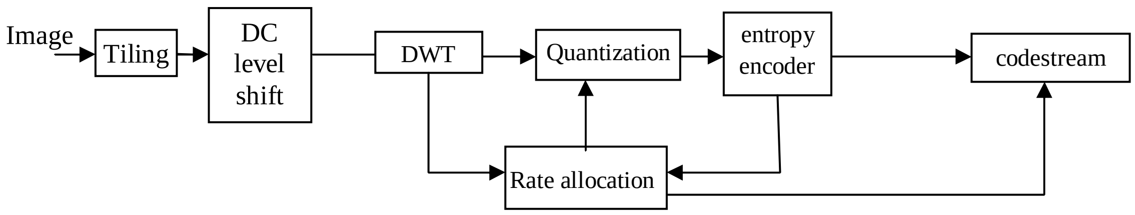


Figure 3.3.   Basic block diagram of JPEG 2000.

The main features of JPEG 2000 are the following:

- *Better compression performance.* For grayscale images, at low bitrates (<0.25bpp) JPEG2000 provides less visible artifacts (due to DWT, certain entropy encoding algorithm, *etc*.) and almost no blocking as compared to JPEG.
- *Multiple resolution representation.* JPEG2000 decomposes the image into a multiple resolution representation during its compression process.
- *Progressive transmission* (*or decoding* by pixel and image resolution accuracy). This means that after receiving a part of the file, the viewer may see a version of the final image in a lower quality. Then, by downloading more bits from the source, the image quality is progressively improved.
- *Lossless and lossy compression.* The JPEG2000 standard provides both lossless and lossy compression similarly to JPEG. Two different wavelets are used by JPEG2000: Cohen-Daubechies-Feauveau (CDF) 9/7 for lossy compression and CDF 5/3 for loss-

less coding.

- *Random code-stream access* or Region Of Interest (ROI). It gives possibility to access and process different parts of an image.

One of the main advantages of JPEG 2000 is the flexibility of code-stream, meaning the ability to truncate at any point the code-stream obtained after compression in order to obtain a lower resolution image. However, at the expense of that flexibility the JPEG2000 needs more complex and computationally demanding coder/decoder. This explains why nowadays there are just few consumer digital cameras based on JPEG 2000.

EBCOT (Embedded Block Coding with Optimized Truncation)

The Embedded Block Coding with Optimized Truncation (EBCOT) algorithm is related to the earlier developed scalable image compression techniques. Scalable compression is related with the generation of a bit-stream that contains embedded subsets, each of which represents an efficient compression of the original image at a reduced resolution (or increased distortion). In other words, EBCOT allows resolution scalability and SNR scalability. The resolution scalability means that the compressed data may be decompressed with a lower spatial resolution, while the SNR scalability allows that decoder reconstructs the image with different qualities (some subsets of the bit stream is decompressed independently).The EBCOT algorithm has been included in JPEG 2000 due to the simplicity of getting these scalable forms and a modest complexity.

3.2 MULTIPLE TRANSFORM BASED IMAGE CODING

In this section we give a brief description of some multiple transform based approaches. These techniques have been studied during the last two decades in order to improve the coding performance of conventional block transform compression techniques.

In literature, there are several interesting algorithms devoted to multibase transform coding of images [12],[43], [88]. Such techniques imply an adaptive transform coding of images that uses an appropriate transform with respect to each image block, which results in more efficient coding than non-adaptive methods. In multibase transform coding proposed in [12], [43], the image blocks are represented in a multidimensional space, for example, as points in the 64-D space. They are classified into different clusters or classes where the blocks of each class are compressed by different transforms. The aim is to partition image data (or blocks) in such a way that the overall compression ratio is maximized. The applied transforms may be either fixed or designed in an adaptive manner. In [12], [43], adaptive methods were used for creating different sets of KLTs. The adaptation starts from an initial set of KLTs which are then improved in an adaptation process. During each iteration, all the blocks are compressed with all transforms (along with DCT) and those transforms are selected that give the best compression results. The corresponding transforms are updated before next iteration. The selection of the best transform is done in suboptimal way according to a criterion that combines both the bit-rate and distortion.

A problem in multibase transform systems is the use of different classes of blocks. This introduces overhead information needed to be sent to the decoder. In [43] authors explain the effect of border translations for transform coding which becomes important for systems with several transforms. It is noted that this kind of multibase transform coding is in an intermediate stage between normal transform coding and vector quantization.

In another work on multibase transform coding [88], authors developed a hybrid technique referred to as the Multiple Bases Representation (MBR) that combines several transforms coding, vector quantization and predictive coding. Three transforms were used in MBR, namely, DCT, Haar and identity transform (IDT). After applying corresponding transforms to each block the statistical modeling is used to parameterize the coefficients and vector probability density functions (pdfs). Then, the coefficients are quantized for minimum mean square error based on parameterized pdf models. Thereafter, the quantized coefficients are entropy coded using modified Huffman coder. The results show that, for example, coding of Lena image with the MBR method can produce good quality under 1bpp.

An approach utilizing the possibility to adjust locally the transform similar to [88] and removing the blocking artifacts using the multiresolution decomposition by wavelet transform is proposed in [12]. The transformed image represented in wavelet tree-like manner is reorganized in a block-wise manner. Then, the conventional multibase transform coding of images is applied to the block-wise representation. The decorrelation is, then, performed in a wavelet transform domain. In this algorithm, the content of different blocks is mixed by filtering across the bounds of blocks. Since the filtering is a part of the transform, there is no need for post-filtering to remove the blocking effects of decompressed images. The algorithm uses the best features from the combination of multibase and wavelet transforms. An approach for fast multibase transform based on optimal multiscanning or reordering method is presented in [13].

3.3 PARAMETRIC TRANSFORM BASED COMPRESSION TECHNIQUES

In general, there exist a potential of further improving the performance of image compression methods by adapting the different transforms to different classes of image blocks instead of applying a fixed transform to all image blocks since none of the fixed transforms may be optimal for all kinds of possible image blocks. From this point of view, the use of parametric transforms described in Section 2.4 for image compression is of an interest.

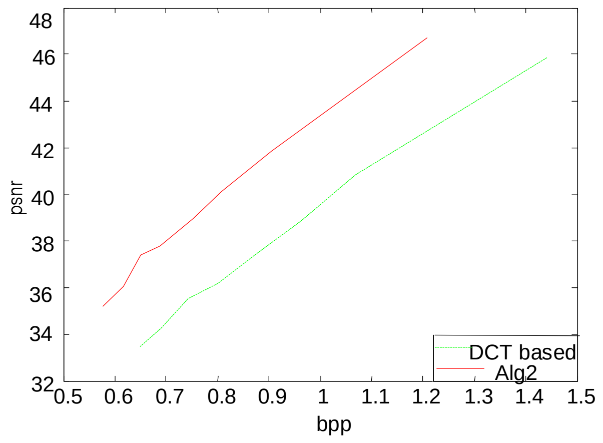
In our publication [P2], performance of the proposed parametric Haar-like transform in application to image compression is studied. The general goal of our work in [P2] is to analyze the potential advantages of adaptive transform based image compression methods over the fixed transform based ones. With this aim, two parametric Haar-like transform based adaptive image compression algorithms are proposed and their performances are compared to the performance of the similar algorithm that is based on the fixed DCT. In both algorithms, the classical DCT is used along with new transforms that are synthesized according to the input image within the class of transforms defined by the parametric Haar-like transform.

The first algorithm called *iterative image compression scheme* (IICS) [59] [P2], is based on an iterative scheme where the classical DCT is used at the first iteration and then, several iteratively synthesized Haar-like transforms are used at the following iterations to refine the compression quality. At each iteration, new Haar-like transform is synthesized according to the generating vector that is obtained from the blocks not efficiently compressed at the previous iteration. The process is iterated as long as a rewarding performance improvement may be achieved.

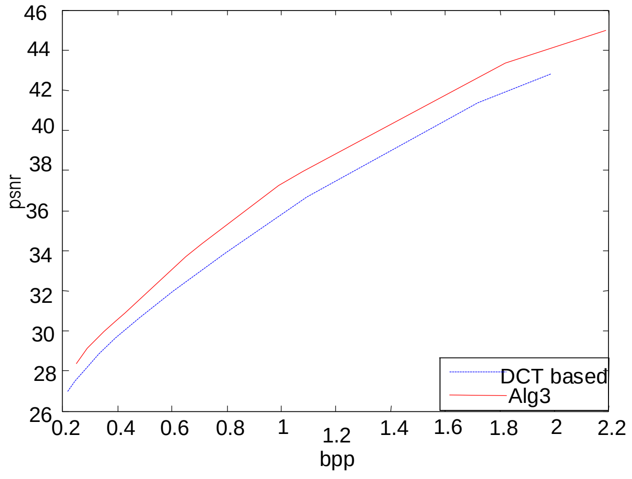
In the second algorithm, called *multiple transform image compression* (MTIC) [15], image blocks are first classified according to their "compressibility" by DCT (image blocks that are similarly distorted when compressed by DCT at a certain rate are grouped together). For each class of blocks a suitable Haar-like transform is synthesized. These Haar-like transforms along with the DCT are applied in parallel to each non-overlapping block of the input image. The transform that achieves the best result is then selected to be assigned to the corresponding block.

It should be noted that both compression schemes proposed in [P2] have a performance that is at least as good as that of the DCT based compression scheme. Extensive simulations were conducted to compare the performance of the parametric Haar-like transform based adaptive image compression methods with the performance of the similar algorithm that is based on fixed DCT. Several types of images were used. For example, Figure 3.4 illustrates PSNR vs. bit-rate plots obtained after applying proposed methods to images "*compound*" and "*cameraman*". The simulation results for other images, *e.g. Lena* and medical image *kidney* are also discussed in [P2]. As we expected, the proposed techniques are better or at least as good as that of the DCT based compression method. Experiments illustrated a moderate performance improvement for natural images and significant performance improvement for images of certain types, such as medical images and complex images consisting of fragments of essentially different types.

Beside compression, PHT based methods perform also block classification. In Figure 3.5 b) one can see not only black and white levels of intensity of blocks but also grey levels. The black blocks correspond to flat regions, which are compressed using DCT transform, while white and grey blocks (non-flat regions) correspond to different parametric Haar-like transforms.

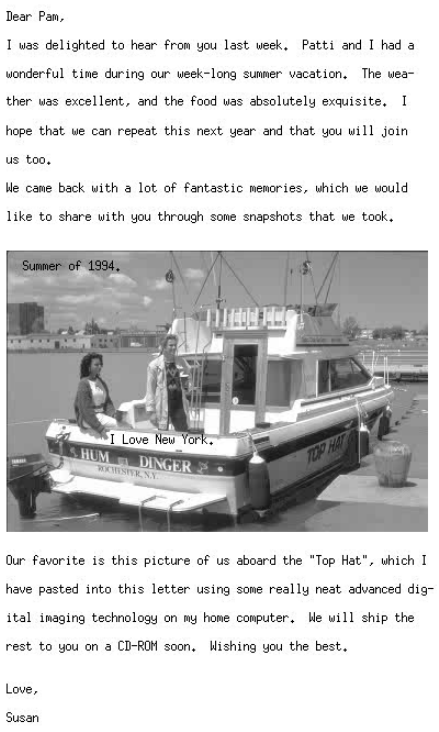


a)



b)

Figure 3.4 Experimental results of proposed methods: a) *compound* image (method *IICS*); b) *cameraman* (method *MTIC*).



Dear Pam,

I was delighted to hear from you last week. Patti and I had a wonderful time during our week-long summer vacation. The wea-ther was excellent, and the food was absolutely exquisite. I hope that we can repeat this next year and that you will join us too.

We came back with a lot of fantastic memories, which we would like to share with you through some snapshots that we took.

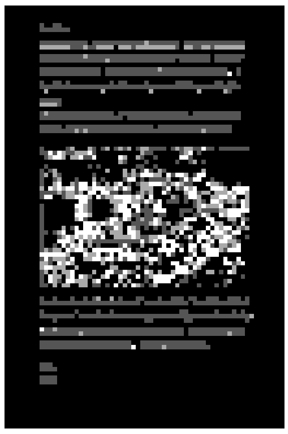
Summer of 1994.

I Love New York.

Our favorite is this picture of us aboard the "Top Hat", which I have pasted into this letter using some really neat advanced dig-ital imaging technology on my home computer. We will ship the rest to you on a CD-ROM soon. Wishing you the best.

Love,

Susan



a) b)

Figure 3.5 a) original *compound* image; b) block classification (MTIC)

Chapter 4

Applications of Transforms in Signal and Image Denoising

Digital images are produced by several physical devices such as digital image and video camera, radar, x-rays. They are used, particularly, in science, industry, medicine, military, security, advertisements, astronomy. In many cases, images may be distorted while being taken or when transmitted or received through a communication channel. There are several reasons for this, for example, blurring (due to shaking or moving a camera), noise corruption *etc*. Image distortion is introduced also by processing method applied to image *e.g.* quantization, smoothing, compression. Similarly, 1D signals created by digital devices are often corrupted, for example, due to measurement errors, quantization, transmission. In many practical both 1D and 2D cases, the corruption may be modelled as a random additive white Gaussian noise (AWGN).

It is important to develop efficient methods for reconstructing useful information about the original 1D signal or 2D image from a corrupted or noisy observation taken by a digital device. Reconstructing a signal or an image from noisy data or, in other words, denoising, is one of the important problems of signal/image analysis which has been intensively studied for many years.

4.1 CONCEPT OF DENOISING

The aim of signal/image denoising is to find an estimate $\tilde{\mathbf{X}}$ of a signal/image corrupted observation $\mathbf{Z}$. In the cases, when the corruption can be modelled as AWGN, that is, $\mathbf{W} \sim N(0,\sigma^2)$, the observation is given as:

$$z(i) = x(i) + w(i), \tag{4.1.1}$$

$$z(i,j) = x(i,j) + w(i,j), \quad i,j = 1,\dots,N \tag{4.1.2}$$

$$\mathbf{Z} = \mathbf{X} + \mathbf{W}, \tag{4.1.3}$$

where (4.1.1) and (4.1.2) correspond to the cases of 1D signal and 2D images, respectively, and (4.1.3) is the general case that represents both.

The targeted estimate $\tilde{\mathbf{X}}$ of the noise-free signal should minimize the expectation of the mean square error (MSE) between the original signal or image $\mathbf{X}$ and its estimate $\tilde{\mathbf{X}}$. In result, an estimated signal will have reduced amount of noise or in the ideal case will contain no noise and be the exact copy of the original signal.

In general, the denoising methods are divided into linear and non-linear ones. The wavelet-based thresholding based techniques [35]-[37] are an example of a non-linear method. An example of linear denoising method is Wiener filtering [49]. The linear methods tend to smooth the details (edges) after denoising. Along with Wiener filtering, the Bayesian estimate method, which is an extension of Wiener method that utilizes the knowledge of signal point statistics, became popular because the resulting images are sharper and less noisy than Wiener filtered images.

The denoising may be performed in spatial or transform domain. Transform domain denoising methods can be subdivided according to the type of basis chosen in the denoising method. Currently, one of the most popular ones are wavelet based denoising methods proposed by Donoho and Johnstone in [35]-[37]. Another popular class of transform based denoising methods are based on the sliding transform concept [63],[64].

The above mentioned transform based denoising methods are summarized in the next section. The proposed signal and image denoising methods are described in Section 4.3.

4.2 GENERAL TRANSFORM BASED DENOISING TECHNIQUES

A general transform domain denoising technique consists of the following three steps:

1. Transform the noisy image into the corresponding transform domain:

$$\mathbf{Y} = \mathbf{HZH}^T, \tag{4.2.1}$$

where **Z** is an observed noisy image, **H** is a transform and **Y** is the transformed image.

2. Apply a non-linear function on the transformed image, for example, the soft or hard thresholding function. This will suppress those coefficients that are smaller than a certain amplitude:

$$\overline{\mathbf{Y}} = F_h\left(\mathbf{Y}(i,j),t\right) \quad \text{or} \tag{4.2.2}$$

$$\overline{\mathbf{Y}} = F_s\left(\mathbf{Y}(i,j),t\right)$$

where $\mathbf{Y}(i, j)$ and $\overline{\mathbf{Y}}(i, j)$ are the entries of the matrices $\mathbf{Y}$ and $\overline{\mathbf{Y}}$, respectively, and $F_h(k,t)$ and $F_s(k,t)$ are the non-linear soft and hard thresholding functions:

$$F_h(k,t)=\begin{cases} k, & |k|\geq t \\ 0, & |k|<t \end{cases} \quad \text{and} \quad F_s(k,t)=\begin{cases} \operatorname{sgn}(k)(|k|-t), & |k|\geq t \\ 0, & |k|<t \end{cases}. \tag{4.2.3}$$

3. Transform $\overline{\mathbf{Y}}$ back into the original domain:

$$\overline{\mathbf{X}}=\mathbf{H}^{-1}\overline{\mathbf{Y}}\mathbf{H}^{-1T} \tag{4.2.4}$$

One of the most utilized transforms for denoising applications is the wavelet transform [62]. However, any other discrete transform may be applied.

The main drawback of the wavelet transform based approaches is the presence of artefacts around discontinuities called pseudo-Gibbs phenomena. Due to this, Coifman and Donoho proposed the translation-invariant (TI) wavelet denoising method presented in [28]. Authors suggested to apply the following scheme:

For a range of shifts $S=\{s_i\}_{i=1}^{N}$ the signal is first shifted by s_i, denoised, and then shifted back, that is,

Unshift(Denoise(Shift(signal))).

Finally, all the shifted estimates corresponding to shifts s_i are averaged to obtain the estimated signal which is smoother and has considerably less Gibbs phenomena compared to conventional transform based denoising.

4.2.1 Utilized thresholding rules

As mentioned in the previous section, the most popular non-linear functions used in transform based denoising techniques are soft and hard thresholding functions. However, other non-linear functions may perform better than either one of these two. In our works [P5],[P6] the customized thresholding function (CTF) introduced in [91] has been used:

$$F_C(x)=\begin{cases} x-sgn(\boldsymbol{x})(1-\alpha)\lambda & \text{if } |x|\geq\lambda \\ 0 & \text{if } |x|\leq\gamma \\ \alpha\lambda(\frac{|x|-\gamma}{\lambda-\gamma})^2\{(\alpha-3)(\frac{|x|-\gamma}{\lambda-\gamma})+4-\alpha\} & \text{otherwise} \end{cases} \tag{4.2.5}$$

where γ is the cut-off value, $0<\gamma<\lambda$, and $0\leq\alpha\leq1$ defines the shape.

The CTF can be considered as a linear combination of soft and hard thresholding functions, that is, $\alpha\cdot F_h(x)+(1-\alpha)\cdot F_s(x)$. Figure 4.1 illustrates these functions:

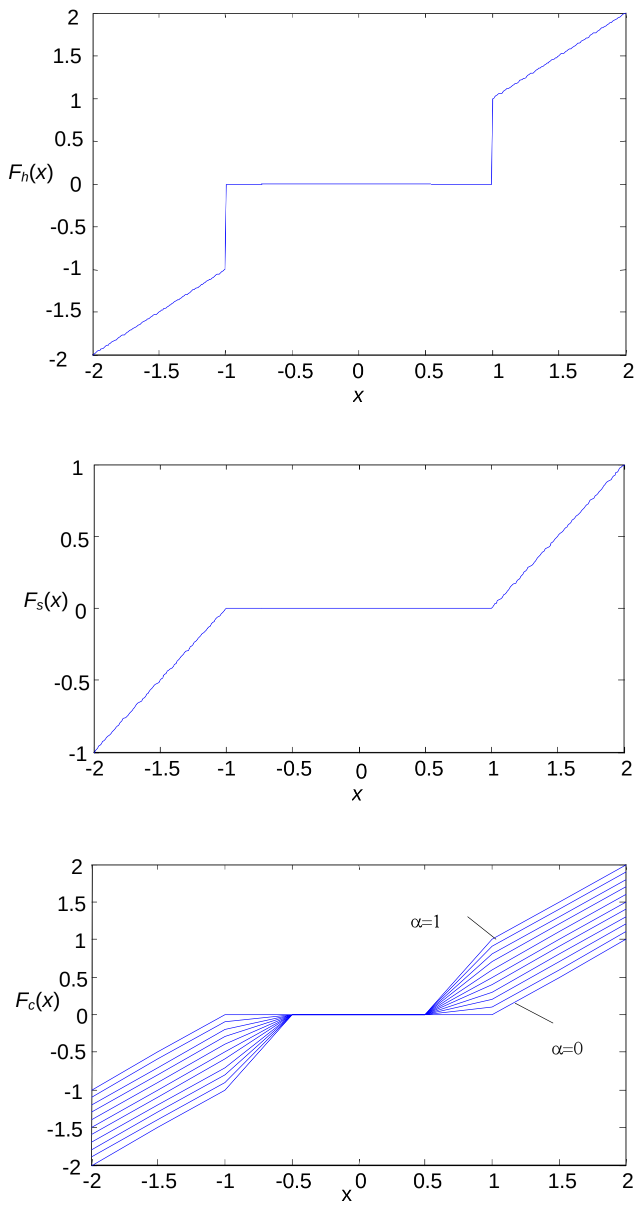


Figure 4.1 Thresholding functions: a) hard thresholding, b) soft thresholding, and c) customized thresholding for various values of α.

4.2.2 Local transform based denoising

The transform based denoising may be further improved when applied rather locally, that is, in a sliding window. Local transform based denoising (LTD) methods have certain advantages. They can be performed using locally varying parameters, for example, varying transform basis, thresholding scheme, etc. They can also use the varying transform support (window size).

The general LTD method applied to an image $\mathbf{Z}=\{z_{i,j}\}$, $i=0,\ldots,N-1$, $j=0,\ldots,M-1$ of size $(N\times M)$ consists of the following steps:

1. Transform a local portion of an input image within a moving window (*k*x*l*) located at (*i*,*j*)-th pixel:

$$\mathbf{Y}_T(i,j)=\mathbf{T}(\mathbf{Z}(i,j)),$$

$$\text{for} \qquad i\in\{0,...,N-k-1\},\ j\in\{0,...,M-l-1\}, \tag{4.2.6}$$

where $\mathbf{Z}(i,j)$ is the data portion of the noisy observation within the window

$$\mathbf{Z}(i,j)=\left[z(i,j),...,z(i+p,j+r)\right],\ p=0,...,k,\ \ r=0,...,l,$$

and $\mathbf{T}$ is a local transform of size $(k+1,l+1)$.

2. Modify the local transform coefficients according to a function η (for example, a thresholding function or element-wise multiplication to constants function) and perform the inverse transform:

$$\tilde{\mathbf{X}}_T(i,j)=\eta(i,j)\mathbf{Y}_T(i,j)$$
$$\tilde{\mathbf{x}}_T(i,j)=T^{-1}\tilde{\mathbf{X}}_T(i,j)$$

3. Combine the resulting estimates of each sample obtained from all window locations that involve that sample. The combination is performed by simple averaging procedure that is very similar to that performed in shift invariant (TI) wavelet denoising [28].

The sliding window strategy can be modified to the moving window strategy by allowing jumps with shifts up to the transform size that corresponds to the non-overlapping case. In the extreme case of non-overlapping blocks a single estimate for each input sample will be obtained that will result in blocking artefacts, while, in the other extreme case of moving the running window with single shifts, the computational complexity will be increased significantly. With this respect, the shift may be considered as an additional parameter in a denoising method. In a wavelet-denoising, this strategy is called partial cycle-spinning based on averaging over the range of shifts instead of performing full TI wavelet denoising [28].

In literature, there are many works related to local sliding window transform based denoising [33],[39],[63],[64],[75],[93]. In [64] the locally adaptive sliding window image denoising which often outperforms the wavelet-based denoising method is introduced.

In [63] local adaptive denoising with adaptively varying local transform support size was presented. Authors use the concept of intersection of confidence intervals rule for selecting the optimum local windows sizes. The algorithm provides a significant noise reduction compared to the denoising based on fixed optimum size local DCT transform, Wiener filter, TI wavelet shrinkage, and other methods.

Another denoising method was proposed in the 3D transform domain [33]. This method for still image denoising based on sliding window transform and block-matching was proposed. The algorithm performs processing of blocks in a sliding way using block similarity concept according to a certain criterion. The denoised images show good results in terms of objective criteria and visual quality.

The local adaptive filtering in transform domain for removing the mixed noise (white and impulsive) was proposed in [39]. The method first detects the “salt-and-pepper” impulses with Min and Max filters and then replaces these impulses by a weighted average or K-nearest neighbour estimate of remaining pixels within the window. Then, in each window, the local transform based filter is used in order to remove the white noise. As a multibase transform the combination of different transforms, *e.g.* Haar and DCT, Haar and Daubechies wavelet as well as their translation-invariant versions have been used. The idea of using the multibase transform is to find the “best” basis or transform not globally, for the whole image, but to find it locally, for each image block. The criterion for selecting a transform is the decorrelation feature or the number of non-zero spectral coefficients.

In [75] a new 3D DCT video denoising of video signals corrupted by additive Gaussian noise is presented. The video signal is locally filtered in sliding 3D windows consisting of highly correlated spatial layers taken from consecutive video frames selected with block-matching technique. The denoising in local windows is performed by hard thresholding of 3D DCT coefficients of each 3D array. Then, the final estimates of reconstructed pixels are obtained by a weighted average of local estimates from all sliding (overlapping) windows. The experiments show an improved performance compared to the wavelet based state-of-the-art video denosing methods in terms of both PSNR and visual appearance.

In [P3], [P4] we used the parametric transform based denoising using sliding window strategy. It can be expected that the use of signal adapted parametric transforms can improve the performance of transform based signal/image denoising. In the next section, we describe shortly the proposed parametric Haar-like transform based 1D and 2D signal denoising methods.

4.3 PHT BASED DENOISING TECHNIQUES AND EXPERIMENTAL RESULTS

As it was mentioned before, the main aim of transform based denoising consists in a separation of the original signal and the noise, in the spectrum of the corrupted input signal. If the noise is AWGN, then it is uniformly distributed in the spectrum of an orthogonal transform. Therefore, a transform with a higher energy compaction separates better the signal from AWGN. One of such transforms could be KLT. However, it is computationally demanding since besides being signal adaptive it has no fast algorithm. Therefore, other transforms such as DCT which is the closest to KLT transform can also be used. Many DOTs such as Fourier, wavelets, DCT, lapped, Haar as well as the combination of transforms (DCT+Haar) with their specific features are utilized in many algorithms. In [P5], [P6] we investigated the performance of PHTs in the noise reduction resulting in two denoising methods.

4.3.1 1D signal denoising based on PHT transform

In our publication [P5] a new denoising approach is proposed based on using parametric transform instead of one or more fixed transform. The main goal of the proposed approach is to improve the performance of fixed transform based denoising methods, in particular, the wavelet transform based approach, by utilizing adaptive methods. The main idea of the proposed approach is to apply different locally adapted transforms to different local parts of the input signal for better decorrelation between the signal and the noise. Intuitively, it is clear that significantly better performance may be achieved by utilizing a correct transform for each local part of the signal. The problem consists in finding that correct transforms. In our proposed algorithms parametric Haar-like transforms (PHTs) described in Section 2 are used. Thus, the problem is reduced to finding correct generating vectors better fitting to each local context of the signal. In the ideal case, if the generating vectors were the local parts of the noiseless signal, perfect decorrelation would take place. Unfortunately, the noiseless signal is what is searched and is not available. Therefore, different estimates of the noiseless signal have to be used. In particular, the PHT based algorithm proposed in [P5] uses the result of classical wavelet denoising method as an estimate of the signal. Thus, in this algorithm the local content within a sliding or moving window is denoised using a new PHT that is synthesized specifically for the content of the window at each location on the signal. These new parametric transforms are synthesized using generating vectors obtained from the estimate of the original signal being the result of wavelet denoising of the input noisy signal. It should be noted that the result of any other traditional denoising algorithm, in particular, any linear denoising algorithm or any transform based denoising algorithm could also be utilized. The proposed algorithm (Wavelet-PHT denoising algorithm) may be described by Figure 4.2.

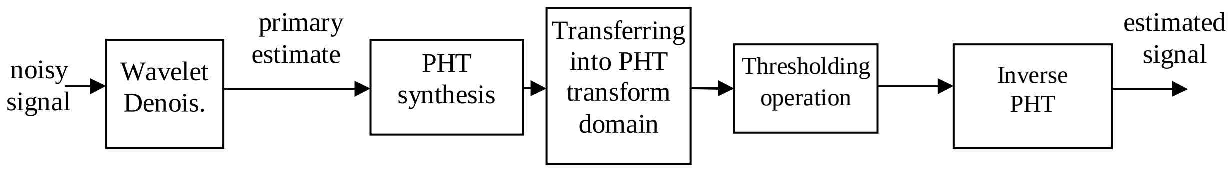


Figure 4.2 Diagram for PHT-based signal denoising algorithm.

The algorithm in Figure 4.2 may be formulated as follows:

1. The input noisy signal is denoised by a wavelet transform in order to find a primary estimate of an uncorrupted signal.
2. Input noisy signal is transferred, window by window, into the domain of PHT transforms that are synthesized on the base of the estimate of the original signal in the corresponding window. For this, both the original noisy signal and its primary estimate are divided into non-overlapping windows. For each window of the primary estimate, new PHT containing the corresponding window content as its first row is synthesized. Then, this transform is applied to the window of the original noisy signal at the same location.

3. Some thresholding function is applied. In our method, we used the CTF (see Section 4.2.2), applied to each transformed window.
4. Each thresholded window is transformed back by the inverse PHTs.

4.3.2 Image denoising based on PHT transform

The idea of this approach, presented in our publication [P6], is similar to the one described in the previous section for denoising 1D signals (Figure 4.2). However, the sliding window approach along rows and columns is used in this case. As a primary estimate the output from TI-wavelet based image denoising is utilized here.

The proposed algorithm may be briefly described as follows:

1. The given noisy image $\mathbf{I}_n$ is denoised by TI wavelet method to get an estimate $\mathbf{I}_{\mathrm{wd}}$, using, for example, hard thresholding scheme.

2. The PHTs are synthesized based on the estimate $\mathbf{I}_{\mathrm{wd}}$. For each subrow and each subcolumn of the input image within a moving window an own PHP is synthesized with the generating vector being a corresponding subrow or subcolumn of the estimate obtained at Step 1. The synthesized PHTs are applied to the corresponding subrows and subcolumns of the sliding window.

3. Thresholding is applied to the transformed windows at each sliding location. In experiments, again the CTF was used.

4. The rows and the columns of each thresholded window are transformed back with inverse PHTs. The inverse PHTs may be computed with fast algorithms.

5. In result, several estimates for each sample of the original image will be obtained (totally w^2 estimates per sample in the middle of the image but less at the edges with w being the window width). The final estimate for each sample is obtained by averaging over all its estimates.

4.3.3 Experimental results

Results on proposed signal denoising method

The proposed 1D signal denoising method has been tested on many artificial signals such as *Blocks, Bumps, HeaviSine*. The signals were corrupted by additive Gaussian noise with signal-to-noise-ratio (SNR) 7 dB, and then denoised by the proposed algorithm [P5]. In all the experiments, the Daubechies asymmetric wavelet with 8 vanishing moments and 8 decomposition levels was used at Step 1 of the algorithm. The results of the experiments were averaged over 30 runs. The parameters α, λ and γ of the utilized thresholding function were chosen empirically. The results show that mostly the proposed method reduces significantly the noise content from the original test signals. It can be seen from [P5], that the proposed method reduces noise significantly in terms of MSE and visual appearance is close to the noiseless signal.

Results on proposed image denosing method

The proposed image denoising method has also been tested on many noisy observations of images such as *Lena*, *Peppers*, *Cameraman*, *House*. The images have been corrupted with AWGN with noise levels $\sigma = 15$ and $\sigma = 20$. For test images the noisy observation images have been denoised with TI Symmlet wavelet with 5 vanishing moments and 4 decomposition levels. The obtained wavelet coefficients were modified by, for example, hard thresholding function with universal threshold value proposed by Donoho and Johnstone $\lambda_{wd} = \sigma\sqrt{2 \cdot \log n^2}$ where n^2 is the image size. This wavelet-based estimate has been used as a start up for calculating of parametric Haar-like transforms (PHT) as described above. After transferring the noisy image into PHT transform domain, the transformed coefficients have been thresholded with customized threshold function using the threshold value $\lambda_{pht} = c \cdot \sigma\sqrt{2 \log w^2}$ where w is the size of the sliding window. Experiments show that our denoising methods outperform significantly the performance of Wiener filtering and the wavelet based method.

Chapter 5

Spectral methods/transforms in logic design

In the design of digital devices, there are many various ways to describe the input and output signals. The inputs and outputs are usually mathematically modelled by functions, while the input and output relations may be represented by operators in some properly selected function space. In digital design, a system is represented by its logic function, namely, Boolean or switching function (BF). The BF of *n* variables $x_1, x_2, ..., x_n$ is a mapping $f:\{0,1\}^n \to \{0,1\}$. The classical representations of a BF are the truth tables, Kaurnaugh map, canonical sum-of-products (SOP) which is the analytical representation of BFs, and the graphical representation of BFs by Decision Diagrams (DD) [24], [77],[79]. Another way is the spectral representation [77], [45]. The conversion from the original domain into an equivalent spectral domain representation is performed through a spectral transform which has an inverse transform to ensure the possibility of obtaining the original domain representation back from its spectrum without loss of information. In general, spectral transforms are orthogonal. One of the reasons to use the spectral representation is due to the fact that sometimes the spectral coefficients provide more valuable information about the input features than just its Boolean representation. This aspect was used in many areas of logic design, optimization and testing of digital circuits [45], [52]. It helps to solve many problems ([44],[45],[27],[52]), such as BF classification into equivalent classes, Boolean matching, disjoint decomposition of Boolean functions, spectral based network synthesis, fault detection.

Another aspect of spectral methods in logic design is the transformation of Boolean functions from AND-OR into AND-XOR circuits which are easily testable and reduce the hardware costs (number of logic gates) compared to the first circuit realization [74].

Many spectral transforms have been widely used in analysis, synthesis and testing of logic functions. Among them the most popular are Reed-Muller (RM), Haar, Walsh and Arithmetic transforms as well as their generalizations (*e.g.* generalized Haar [5], Vilenkin-Chrestenson

[52]). The RM transform is found to be useful in terms of area, speed and testability [46],[51] [74],[78]. In [69] the symmetric functions were realized as RM expressions. The RM spectral representation represents flexibility for description of the binary as well as multiple-valued circuits. The rows of RM transform are the well-known error-correcting codes called RM codes.

The Walsh transform is widely used, for example, in fault detection [45]. The Haar transform is also quite useful, for example, in VLSI design, fault detection [44],[52],[50] and also in signal/image processing and pattern recognition due to its low computations and memory requirements. In the following sections, we will describe briefly different existing as well as proposed spectral representations of logic functions.

5.1 FUNCTIONAL REPRESENTATIONS OF SWITCHING FUNCTIONS: AND-EXOR EXPRESSIONS

In algebraic approach to logic design, switching functions are usually considered as elements of vector space $GF_2(C_2^n)$, where GF_2 denotes the finite Galois filed of order 2, GF(2), and C_2^n is the finite dyadic group of order 2^n, consisting of $\{(x_1, x_2, ..., x_n) \mid x_i \in \{0,1\}, \oplus\}$, where '$\oplus$' is the modulo 2 addition or EXOR operation, and $C_2^n = \underset{i=1}{\overset{n}{\times}} C_2$, where $C_2 = (\{0,1\}, \oplus)$ is the basic cyclic group of order 2. This formulation of switching functions is convenient due to the benefits of EXOR representations. AND-XOR circuits (realizing EXORs) are easily testable and less costly (see [74]) compared to AND-OR circuits (realizing SOPs). Moreover, they may be easily extended to multiple-valued functions. AND-EXOR representations are considered as Fourier series-like expansions in $GF_2(C_2^n)$.

Any n-variable switching function $f = f(x_1, x_2, ..., x_n)$ can be represented in one of the following EXOR expansion forms:

$f = \overline{x}_i f_0 \oplus x_i f_1,$	*Shannon* (*S*)
$f = f_0 \oplus x_i f_2,$	*positive Davio* (*pD*)
$f = f_1 \oplus \overline{x}_i f_2,$	*negative Davio* (*nD*)

where

$$f_0 = f(x_1, ..., x_{i-1}, x_i = 0, x_{i+1}, ..., x_n) = f(x_i = 0)$$

and

$$f_1 = f(x_1, ..., x_{i-1}, x_i = 1, x_{i+1}, ..., x_n) = f(x_i = 1)$$

are co-factors with respect to variable x_i, and

$$f_2 = f_0 \oplus f_1.$$

(S)-expansion

The (S)-expansion rule may be applied recursively to all the variables x_i, $i=1,...,n$, of function f bringing to the complete SOP form of f [77], [79].

In matrix notation, Shannon expansion may be expressed as:

$$f=\begin{bmatrix}\overline{x}_i & x_i\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}f_0\\ f_1\end{bmatrix}, \tag{5.1.1}$$

or, after denoting the basic matrices

$$\mathbf{X}(1)=\begin{bmatrix}\overline{x}_i & x_i\end{bmatrix},\ \mathbf{B}(1)=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\ \text{and}\ \mathbf{F}=\begin{bmatrix}f_0\\ f_1\end{bmatrix},$$

we have:

$$f=\mathbf{X}(1)\mathbf{B}(1)\mathbf{F}. \tag{5.1.2}$$

In matrix notation, the recursive application of (S)-expansion to all n variables of function f resulting in complete SOP form of function f may be expressed by the Kronecker product as:

$$f=\mathbf{X}(n)\mathbf{B}(n)\mathbf{F}=\left(\bigotimes_{i=1}^{n}\mathbf{X}(1)\right)\left(\bigotimes_{i=1}^{n}\mathbf{B}(1)\right)\mathbf{F}. \tag{5.1.3}$$

It should be noted that matrix notation is convenient for transferring from SOP into, for example, polynomial expression of switching function. For this approach, different basic matrices B(1) can be chosen to produce different expressions.

(pD)-expansion

The (pD)-expansion may be formed from the (S)-expansion by using properties of the Boolean logic. For example, after substitution $\overline{x}_i = x_i \oplus 1$ we have

$$\begin{aligned}f &= (1\oplus x_i)\,f_0 \oplus x_i f_1 = 1\cdot f_0 \oplus x_i f_0 \oplus x_i f_1 = \\ &= 1\cdot f_0 \oplus x_i\,(f_0 \oplus f_1).\end{aligned}$$

In general,

$$f=c_0 \oplus c_1 x_i, \tag{5.1.4}$$

where $c_0 = f_0$ and $c_1 = f_0 \oplus f_1$ are the RM spectral coefficients. In the matrix form (5.1.4) may be expressed as

$$f = [1 \;\; x_i]\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} f_0 \\ f_1 \end{bmatrix},$$

$$f = \mathbf{X}_R(1)\mathbf{R}(1)\mathbf{F}. \qquad (5.1.5)$$

The recursive application of (*pD*) rule to a function $f(x_1, x_2, ..., x_n)$ with respect to all the n variables produces the *Positive Polarity Reed-Muller* (PPRM) expansion or the so called Zegalkin polynomial [19],[77] :

$$\begin{aligned} f &= c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus \cdots c_nx_n \oplus \\ &\oplus c_{12}x_1x_2 \oplus c_{13}x_1x_3 \cdots \oplus c_{n-1n}x_{n-1}x_n \oplus \cdots c_{12...n}x_1x_2...x_n. \end{aligned} \qquad (5.1.6)$$

In matrix notation PPRM is the following:

$$f = \mathbf{X}_R(n)\mathbf{R}(n)\mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{X}_R(1)\right)\left(\bigotimes_{i=1}^{n} \mathbf{R}(1)\right)\mathbf{F}. \qquad (5.1.7)$$

The PPRM represents AND-EXOR expression where each variable is uncomplemented, that is, with positive polarity. Each switching function can be written in the form of PPRM.

As we can see in (5.1.7), the PPRM expression requires the Kronecker product of basic kernels **R**(1). The product terms, mentioned above in (5.1.6), (5.1.7), are generated symbolically as the Kronecker product of (1x2) vectors $[1 \;\; x_i]$, $i = 1, ..., n$, related to each switching variable, given by

$$\mathbf{X}_R = \bigotimes_{i=1}^{n}[1 \;\; x_i], \qquad (5.1.8)$$

For example, when n=2, we have

$$\mathbf{X}_R(2) = [1 \quad x_1] \otimes [1 \quad x_2] = [1 \quad x_2 \quad x_1 \quad x_1x_2]. \qquad (5.1.9)$$

These product terms represent particular switching functions, whose truth vectors are the columns of the RM transform matrix $\mathbf{R}(n)$ shown in Table 5.1.1.

Table 5.1.1 Truth vectors for products in Reed-Muller expressions

x_1x_2	**1**	x_2	x_1	x_1x_2
00	1	0	0	0
01	1	1	0	0
10	1	0	1	0
11	1	1	1	1

(nD)-expansion

Similar to the PPRM ((*pD*)-expansion), the NPRM ((*nD*)-expansion) may be derived from (*S*)-expansion by the relation $x_i = \overline{x}_i \oplus 1$. In matrix notation, the NPRM is given as

$$f = \begin{bmatrix} 1 & \overline{x}_i \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \end{bmatrix}. \tag{5.1.10}$$

Both the PPRM and NPRM expressions may be included into the so-called *fixed-polarity Reed-Muller* (FPRM) expression. The FPRM is a generalization of PPRM (NPRM), that for each variable of function allows to choose any of two decomposition rules, (*pD*) or (*nD*). It means that each variable should appear in one of two fixed polarities, *e.g.* either positive or negative, but never both at the same time. For *n* variable function *f* there are 2^n possible choices for polarities. Therefore, there are 2^n different FPRM polynomial representations for *n* variable function. With this respect, the polarity vector $h = (h_1, h_2, \ldots, h_n)$, $h_i \in \{0,1\}$ with $h_i = 0$ (positive polarity of variable) and $h_i = 1$ (negative polarity of variable) is introduced that relates to polarity of variables in f denoted as:

$$x_i \oplus h_i = \begin{cases} x_i, & h_i = 0 \\ \overline{x}_i, & h_i = 1. \end{cases}$$

In particular, the PPRM representation is the RM expression for zero-polarity vector $h = (0, 0, \ldots, 0)$ that often requires less number of product terms than the SOP which is AND-OR expression. The polarity vector *h* with minimal number of nonzero coefficients is called the minimal polarity and the FPRM corresponding to that minimal polarity is called the minimal FPRM expression for a function *f*. As in (5.1.7), the FPRM of function $f(x_1, x_2, \ldots, x_n)$ for polarity vector h may be written as

$$f = \mathbf{X}_R^h(n)\mathbf{R}^h(n)\mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{X}_R^{h_i}(1) \right) \left(\bigotimes_{i=1}^{n} \mathbf{R}_i^{h_i}(1) \right) \mathbf{F}, \tag{5.1.11}$$

where $\mathbf{X}_R^{h_i} = \begin{bmatrix} 1 & x_i^{h_i} \end{bmatrix}$ and the Kronecker product of basic matrices can be written as

$$\mathbf{R}^h = \bigotimes_{i=1}^{n} \mathbf{R}^{h_i}(1), \tag{5.1.12}$$

where $h = (h_1, h_2, \ldots, h_n)$ is the polarity vector, $h_i \in \{0,1\}, i = 1, \ldots, n,$ is a polarity value for variable x_i, and matrix $\mathbf{R}^{h_i}(1)$ is one of following two RM matrices:

$$\mathbf{R}^{h_i}(1) = \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, h_i = 0 \\ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, h_i = 1. \end{cases} \tag{5.1.13}$$

The generalization of FPRM gives the mixed polarity Reed-Muller (MPRM), also referred to as the Kronecker expression, that allows a free choice for each variable among the three decomposition rules, namely, (*S*), (*pD*) and (*nD*), with no products consisting of the same set of variables. In this case, we have three kernels, that is, identity kernel, PPRM and NPRM and, therefore, three choices. Thus, for each function of n variables 3^n possible Kronecker expansions will be obtained.

The more is the number of possible expressions, the more is the possibility of getting the expression with smaller number of products. However, finding an optimal expression of a function with a minimal number of products is an NP-hard problem [77].

In *generalized RM* (GRM) the polarity of each variable may be different in different product terms. There are $2^{n2^{n-1}}$ possible generalized RM expressions for an *n* variable function. The set of variables in product terms is unique, that is, there are no products with the same set of variables.

The GRM expression for function of *n* variables has the following form:

$$f = c_0 \oplus c_1\tilde{x}_1 \oplus c_2\tilde{x}_2 \oplus \cdots c_n\tilde{x}_n \oplus c_{12}\tilde{x}_1\tilde{x}_2 \oplus c_{13}\tilde{x}_1\tilde{x}_3 \cdots \oplus c_{n-1n}\tilde{x}_{n-1}\tilde{x}_n \oplus \cdots c_{12\ldots n}\tilde{x}_1\tilde{x}_2\ldots\tilde{x}_n,$$

where $c_i \in \{0,1\}$ and $\tilde{x}_i$ is either x_i or $\overline{x}_i$.

Another AND-EXOR expression is the EXOR sum of product expressions (ESOPs) which are the most general class of expressions defined as an EXOR sum of arbitrary product terms of the form:

$$f = \bigoplus_S \tilde{x}_1\tilde{x}_2\ldots\tilde{x}_n,$$

where *S* is the set of all possible products and $\tilde{x}_i$ is 1, x_i or $\overline{x}_i$.

There is the following relation between different expressions: $PPRM \subset FPRM \subset GRM \subset ESOP$ [79]. These expressions provide different number of products (or nonzero spectral coefficients) in direction from left to right. That is, on average, for many BFs, the PPRM produces the largest number of products, while the ESOP gives the least number of products.

In the above described expressions, the spectral coefficients c_i and the basis functions are logic 0 and 1. Such expressions where coefficients and basis functions take values in a finite field are called *bit-level expressions*. The bits may be binary as well as multiple-valued. In the case of RMs the calculations are modulo 2 or over finite field GF(2). In the following section we discuss the *word-level expressions* having real or complex-valued coefficients.

5.2 WORD-LEVEL EXPRESSIONS

Examples of word-level expressions are the Arithmetic, Walsh and Haar expressions [79]. Arithmetic expressions are derived from RMs when operations over GF(2) are replaced, for example, by the operations over the field of rational numbers Q.

In general, the following relation exists between logic (disjunction '$\vee$', conjunction '$\wedge$', negation '$\neg$', EXOR '$\oplus$') and arithmetic operations (addition, subtraction, multiplication):

$$\begin{aligned} x \wedge y &\equiv xy \\ x \vee y &\equiv x + y - xy \\ x \oplus y &\equiv x + y - 2xy \\ \overline{x} &\equiv 1 - x. \end{aligned} \tag{5.2.1}$$

An arithmetic expression may be obtained from (S)-expansion by using the last substitution in (5.2.1). In this case, the arithmetic expansion (decomposition) rule with respect to variable x_i is:

$$f = 1 \cdot f_0 + x_i \cdot (-f_0 + f_1). \tag{5.2.2}$$

For n variable function f the arithmetic expression similar to (5.1.7) is the following:

$$f = \mathbf{X}_A(n)\mathbf{A}(n)\mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{X}_A(1) \right) \left(\bigotimes_{i=1}^{n} \mathbf{A}(1) \right) \mathbf{F}, \tag{5.2.3}$$

where $\mathbf{X}_A(1) = [1 \;\; x_i]$ with integer values 0 and 1 of x_i, and $\mathbf{A}^{-1}(1) = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$ is the basic arithmetic transform kernel.

Another word-level expression is the Walsh expression. The columns of the basic Walsh transform $\mathbf{W}(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ may be represented with switching variables as $[1 \;\; 1 - 2x_i]$ and

$$f = \frac{1}{2}[1 \;\; 1 - 2x_i] \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \end{bmatrix} = \frac{1}{2}\left(1 \cdot (f_0 + f_1) + (1 - 2x_i)(f_0 - f_1)\right). \tag{5.2.4}$$

This expression is called Walsh expansion rule with respect to variable x_i. The recursive application of Walsh rule to all variables will give the following:

$$f = \mathbf{X}_W(n)\mathbf{W}(n)\mathbf{F} = \left(\bigotimes_{i=1}^{n} \mathbf{X}_W(1) \right) \left(\bigotimes_{i=1}^{n} \mathbf{W}(1) \right) \mathbf{F}, \tag{5.2.5}$$

where $\mathbf{X}_W(1) = [1 \;\; 1 - 2x_i]$.

5.3 SPECTRAL APPROACH: BOOLEAN VS. TRANSFORM DOMAIN

A logic function can be represented by many bases. The choice of a basis plays an important role in solution of analysis and synthesis problems using spectral representations of logic functions, particularly, it determines the number of nonzero spectral coefficients which, in turn, determines the complexity of implementation of logic functions. In this section, we give a short description of RM, arithmetic and Walsh transforms which have a Kronecker product structure allowing fast FFT-like calculation for their transform matrices (see Chapter 2).

Reed-Muller transform

The RM transform performs the transformation of a BF from its original domain into the Reed-Muller domain, and vice versa. The expression (5.1.7) may be represented in matrix form as following:

$$f = \mathbf{X}_R \mathbf{R}(n)\mathbf{F},$$

where $\mathbf{F} = \left[f(0), f(1),..., f(n-1)\right]^T$ is the truth vector of n-variable switching function f and $\mathbf{R}(n)$ is the RM transform matrix of order $N = 2^n$:

$$\mathbf{R}(n) = \bigotimes_{i=1}^{n} \mathbf{R}(1), \quad \mathbf{R}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{5.3.1}$$

The RM transform of order $2^n \times 2^n$ is recursively defined as

$$\mathbf{R}(n) = \begin{bmatrix} \mathbf{R}(n-1) & \mathbf{O}(n-1) \\ \mathbf{R}(n-1) & \mathbf{R}(n-1) \end{bmatrix},$$

where $\mathbf{O}(n-1)$ is the zero matrix of order 2^{n-1}. The elements of $\mathbf{R}(n)$ are the logical 0 and 1, and the calculations are done over $GF(2)$. The matrix $\mathbf{R}(n)$ is self-inverse, therefore, $\mathbf{R}^{-1}(n) = \mathbf{R}(n)$.

For a truth vector $\mathbf{F}$ of n variable switching function f the forward and inverse RM transforms are:

$$\begin{aligned} \mathbf{R}_f &= \mathbf{R}(n) \times \mathbf{F} \\ \mathbf{F} &= \mathbf{R}(n) \times \mathbf{R}_f, \end{aligned} \tag{5.3.2}$$

where the calculations are performed over the field GF(2).

The forward and inverse transforms are given by the same matrix, since the RM transform matrix $\mathbf{R}(n)$ is self-inverse over $GF(2)$. The relations (5.3.2) form the RM transform pair.

Arithmetic Transform

The arithmetic transform is defined as

$$\mathbf{S}_A = \mathbf{A}(n)\mathbf{F}, \tag{5.3.3}$$

where $\mathbf{S}_A$ and $\mathbf{F}$ are the arithmetic spectrum and truth vector, respectively. Arithmetic transform matrix $\mathbf{A}(n)$, as Walsh and RM transforms, is derived recursively as

$$\mathbf{A}(1) = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, \qquad \mathbf{A}(n) = \begin{bmatrix} \mathbf{A}(n-1) & \mathbf{O}(n-1) \\ -\mathbf{A}(n-1) & \mathbf{A}(n-1) \end{bmatrix}, \tag{5.3.4}$$

or, alternatively, $\mathbf{A}(n) = \bigotimes_{i=1}^{n} \mathbf{A}(1) = \mathbf{A}(1)^{\otimes n}$.

It should be noticed that the inverse of arithmetic transform, $\mathbf{A}^{-1}(1)$, formally, is the same as the $\mathbf{R}(1)$ containing the logical values of 0 and 1. However, the values of $\mathbf{A}^{-1}(1)$ are integer 0 and 1 values.

Walsh transform

The Walsh transform is defined as

$$\mathbf{S}_W = \mathbf{W}(n)\mathbf{F}, \tag{5.3.5}$$

where $\mathbf{S}_W$ and $\mathbf{F}$ are the Walsh spectrum and truth vector, respectively. Walsh transform matrix $\mathbf{W}(n)$ is derived recursively (see also (2.3.3)):

$$\mathbf{W}(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \mathbf{W}(n) = \begin{bmatrix} \mathbf{W}(n-1) & \mathbf{W}(n-1) \\ \mathbf{W}(n-1) & -\mathbf{W}(n-1) \end{bmatrix} \tag{5.3.6}$$

or, alternatively, $\mathbf{W}(n) = \bigotimes_{i=1}^{n} \mathbf{W}(1) = \mathbf{W}(1)^{\otimes n}$.

For some applications, in computing the Walsh transform of a BF, it is convenient to perform the following encoding: the elements of **F** are sometimes encoded from {0, 1} to {1, -1}, where logic 0 is encoded as -1 and logic 1 is encoded as +1.

5.4 REED-MULLER TRANSFORM FOR MULTIPLE-VALUED LOGIC

The Reed-Muller domain can be extended from binary logic, related to Boolean algebra, to the multiple-valued logic (MVL) suggesting much more benefits. It was shown that MVL circuits enhance circuit performance in terms of chip area, operation speed and power consumption. Any function f with p-valued inputs and p-valued outputs may be considered as a mapping: $f:\{0,1,2,\ldots,p-1\}^n \to \{0,1,2,\ldots,p-1\}$. The PPRM for *p*-valued function f with truth vector **F** has the following form:

$$f_{rp}(x) = \mathbf{X}_{rp}(n) \times \mathbf{R}_p^{-1}(n) \times \mathbf{F},$$

where $\mathbf{X}_{rp}(n) = \bigotimes_{i=1}^{n}[1 \;\; x_i \;\; x_i^2 \;\; x_i^3 \ldots x_i^{p-1}]$ and $\mathbf{R}_p^{-1}$ is a matrix over GF(*p*).

Here we consider an example of the simplest case of multiple-valued logic, the case of three-valued input, three-valued output function $f:\{0,1,2\}^n \to \{0,1,2\}$. The PPRM transform for such function is

$$\mathbf{G} = \mathbf{R}_3(n) \times \mathbf{F} \quad \text{over} \quad GF(3), \tag{5.4.1}$$

where **F** and **G** are the ternary truth vector and the PPRM spectrum of length 3^n, respectively; $\mathbf{R}_3(n)$ is a $3^n \times 3^n$ PPRM transform matrix over GF(3) defined as

$$\mathbf{R}_3(n) = \bigotimes_{i=1}^{n} \mathbf{R}_3(1) \quad \text{and} \quad \mathbf{R}_3(1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix}. \tag{5.4.2}$$

For example, the PPRM expression for a truth vector **F** of a 3-valued function *f* of *n* variables is given as follows:

$$f_{r3}(x) = \mathbf{X}_r(n) \times \mathbf{R}_3(n) \times \mathbf{F},$$

where $\mathbf{X}_r(n) = \bigotimes_{i=1}^{n}\begin{bmatrix} 1 & x_i & x_i^2 \end{bmatrix}$ and $\mathbf{R}_3^{-1}(1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$.

It should be noted that the addition and multiplication operations over GF(3) are modulo 3 addition and multiplication. However, this is not true for the general GF(*p*) case.

5.5 REED-MULLER HAAR TRANSFORM

This section describes the matrix representation of Reed-Muller Haar transform (RMH) for different algebraic structures. The idea of the RMH transform can be found in more detail in [84]. Here we bring only briefly the definition of that transform.

Denote by $P(G)$ the space of the functions $f:G \to P$, where *G* (domain) is a finite group and *P* (range) is a field that may be the complex field *C* or a finite (Galois) field *GF*(*p*). In this notation, we consider the space of *n* variable switching functions $GF_2(C_2^n)$, defined in section 5.1.

For a function $f \in GF_p(C_p^n)$ given by its truth vector $\mathbf{F} = [f(0),..., f(p^n - 1)]^T$ the Haar spectrum and the series expansion are defined as:

$$\begin{aligned} \mathbf{S}_f &= \mathbf{H}^{-1}(n)\mathbf{F} \\ \mathbf{F} &= \mathbf{H}(n)\mathbf{S}_f, \end{aligned} \tag{5.5.1}$$

where $\mathbf{S}_f$ is the vector of Haar spectral coefficients and, according to definition, Haar functions in $GF_p(C_p^n)$ are defined by the columns of the following matrix:

$$\mathbf{H}(n) = \begin{bmatrix} \mathbf{H}(n-1) \otimes \mathbf{q}_0 \\ \mathbf{I}(n-1) \otimes \mathbf{q}_1 \\ \vdots \\ \mathbf{I}(n-1) \otimes \mathbf{q}_{p-1} \end{bmatrix}, \tag{5.5.2}$$

where $\mathbf{q}_i$, $i = 0,..., p-1$ are rows of basic matrix $\mathbf{Q}^T(1)$ for *P*=*GF*(*p*), and $\mathbf{I}(r)$ is the identity matrix of order r.

For example, depending on the order *p*=2,3,4 of the finite filed $GF(p)$ the following basic matrices (kernels) $\mathbf{Q}^T(1)$ for definition of Haar functions are possible:

$$\mathbf{R}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \mathbf{GF}_3(1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}, \mathbf{GF}_4(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 3 & 2 & 1 \end{bmatrix}.$$

In particular, for *p*=3 the RMH of order 3^n is the following:

$$\mathbf{TH}(n) = \begin{bmatrix} \mathbf{TH}^T(n-1) & \otimes_3 & [1\ \ 1\ \ 1] \\ \mathbf{I}(n-1) & \otimes_3 & [0\ \ 1\ \ 2] \\ \mathbf{I}(n-1) & \otimes_3 & [0\ \ 1\ \ 1] \end{bmatrix}, \tag{5.5.3}$$

where $\mathbf{TH}(1) = (\mathbf{GF}_3(1))^T$ is the RMH transform of order 3, and $\otimes_3$ is the Kronecker product modulo 3.

5.6 HYBRID TRANSFORMS

In this section we introduce briefly the proposed hybrid spectral transforms based techniques that were designed in order to suit better the needs of a particular application. The basis functions of the hybrid transforms are designed after analysis of the features of an input logic function. Each transform has both more or less good properties when used in certain task or application. In order to use the best features of different transforms, many new transforms are designed that combine these features [3],[41],[61],[73].

5.6.1 Hybrid Reed-Muller Haar transform

In [P1], [61], a new Hybrid Reed-Muller Haar Transform (HybRMH) transform was proposed which is a hybrid transform obtained by combining the RM and RMH transforms. We give also a fast calculation algorithm for this transform.

The forward (0-polarity) HybRMH transform of order $N = 2^n$ an integer parameter r is defined as

$$\mathbf{T}(n,r) = \mathbf{R}(r) \otimes \mathbf{H}(n-r), \quad 0 \le r \le n \tag{5.6.1}$$

where $\mathbf{R}(r)$ is the Reed-Muller transform of order 2^r and $\mathbf{H}(n-r)$ is the Reed-Muller Haar transform of order 2^{n-r}.

The parameter r can be selected appropriately to best exploit the features of the function to be represented. In the case $r = 0$ the hybrid transform coincides with the RMH transform, while in case $r = n$ the hybrid transform coincides with the RM transform of order 2^n.

For n = 3 and r = 1, the hybrid transform matrix $\mathbf{T}(3,1)$ is the following:

$$\mathbf{T}(3,1) = \mathbf{R}(1) \otimes \mathbf{H}(2) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Both the RM and RMH transforms have a fast calculation algorithm. The fast algorithm for the RM transform has a structure similar to that of the Hadamard transform, while the flow-graph of the RMH transform is similar to that of the fast Haar transform. The butterfly diagram of hybrid transform has a structure of fast Hadamard-Haar transform in the sense that some of the flowgraph stages are similar to that of fast Hadamard transform while the others are similar to that of fast Haar transform. The number of additions of fast hybrid RM-RMH transform of order N and the parameter r, is $2^r \cdot (N/2^r - 1) + r \cdot (N/2)$. The computational costs are different for different values of r. In other words, for $r = 1,2,\ldots,n-1$ the number of additions in the fast algorithm is between the number of additions corresponding to the RMH and RM transforms.

The fixed-polarity expression for HybRMH transform is defined in [P1]. Fast implementation algorithm can be derived also for the fixed polarity HybRMH transform.

The binary HybRMH transform may be generalized to the case of multiple-valued functions. In particular, the ternary hybrid transform with definitions of ternary fixed-polarity hybrid expressions is described in [P1].

5.6.2 Binary hybrid Reed-Muller Haar-like transform

In [P4] we presented another new Haar-like transform referred to as the Binary Reed-Muller Haar-like (BRMH) transform over Galois fields of order 2.

The proposed forward (and inverse) BRMH transform, similarly to the HybRMH, is a discrete binary transform which may be defined as follows:

$$\mathbf{T}(n,k) = \mathbf{R}(n-k) \otimes \mathbf{H}_1(k) \tag{5.6.2}$$

where $\mathbf{R}(n-k)$ is the RM transform matrix of order 2^{n-k} and $\mathbf{H}_1(k)$ is the Binary Parametric Haar-like Transform (BPHT) matrix of order 2^k to be defined later. By varying the parameter k one may get different transforms, *i.e.*, RM (k=0), RMH (k=n) as well as other binary transforms corresponding to intermediate values of parameter k.

The BPHT is a PHT transform over GF(2) designed for each given input signal by taking into account specific features of the input binary signal. Like PHT, the BPHT has the fast algorithm with the structure similar to that of the fast Haar transform.

Similar to the PHT, the BPHT is based on a generating vector which is an input to a fast implementation algorithm flowgraph (Figure 2.5 b)). In our method, a switching truth vector serves as such a generating vector and an input to that flowgraph. As in the case of FHT algorithm, the fast BPHT of order $N = 2^n$ has $m = \log_2 N$ stages, with the jth stage, $j = 1,...,m$, consisting of $N_j = N/2^j$ butterflies.

The design of the fast BPHT transform starts from analyzing each pair $[u\ v]^T$ of the input binary truth vector of a logic function. At each stage of the fast algorithm (Figure 2.5), for each pair $[u\ v]^T$ of an input binary vector a kernel $\mathbf{V}$ is specified in such a way that

$$\mathbf{V} \cdot [u\ v]^T = [d\ \ 0]^T, \text{ where } d \neq 0.$$

In particular, for each stage j and for each pair $[u\ v]^T$ of the input vector, the sth kernel is defined in the following way:

$$\mathbf{V}^{(1,s)} = \begin{cases} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, & \text{if } u_{1,s} = v_{1,s} = 1 \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, & \text{if } u_{1,s} = 1, v_{1,s} = 0, \text{ or } u_{1,s} = v_{1,s} = 0 \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, & \text{if } u_{1,s} = 0, v_{1,s} = 1 \end{cases}$$

These spectral kernels form the block-diagonal matrices in (2.4.4). The resulting BPHT matrix is constructed according to (2.4.1). The BPHT is then combined with RM by Kronecker product operation to get a new transform (5.6.2).

5.7 TERNARY HAAR-LIKE TRANSFORM

In [P3], the spectral technique based on prior analysis of input logic function and its further processing by a new local ternary Haar-like transform (THT) is proposed. The THT is a linearly independent transform over the finite Galois field GF(3) and a signal adaptive parametric transform allowing to use specific features of a signal during construction of the transform. It also has a fast implementation algorithm in structure similar to that of the fast ternary Haar transform. Given an input signal, we specify the parametric THT by its fast computation algorithm allowing an iterative analysis of the signal by its decomposition into input steps of fast algorithm. For each step, the set of suitable parameters (*e.g.* ternary butterflies and permutations) are specified depending on the structure of the input signal at that step. The construction of parametric THT (as in the case of PHT) is based on a generating vector which is the input to the fast implementation algorithm. In our method, a ternary signal serves as such an input. The proposed transform have been utilized in compact representation of ternary logic functions containing the logic values {0, 1, 2} from GF(3).

The design of fast THT transform starts from examining each triple $[u\ v\ w]^T$ of the input ternary truth vector of a logic function. For each set of components in an input triple, a kernel V is specified in such a way that

$$V \cdot [u\ v\ w]^T = [d\ 0\ 0]^T, \text{ where } d \neq 0.$$

With this respect, in [P3] as parameters, a library out of four different kernels and permutation matrices were specified in order to get a structure similar to that of the fast ternary Haar-like algorithm.

5.7.1 Application of HybRMH, BRMH and THT transforms

The HybRMH transform (with fixed polarity expressions) and the BRMH were utilized in compact representation of different binary benchmark functions in order to reduce the hardware requirements. The THT transform based approach was tested on arbitrary selected ternary logic functions. The proposed methods show an improvement in reduction of number of nonzero coefficients. The results were compared with RM, RMH (also FPRM, FPRMH), ternary transforms [40] and show that on average the number of products is smaller for the proposed transforms. The results of the proposed methods may be found in tables of [P1] and [P4].

Summary of Publications

The thesis is based on six publications given in the list of publications. The first two publications are journal papers whilst the others are conference papers.

In [P1] we propose a new transform which is a combination of two transforms, namely, Reed-Muller and Reed-Muller Haar, called hybrid Reed-Muller-Haar transform (HybRMH). We show that the matrix of the HybRMH transform may be factorized, which provides a fast calculation algorithm based on the classical theory of FFT-like algorithms. As one of many potential applications of this transform, we consider the reduction of the number of nonzero coefficients in HybRMH expressions. The extension of HybRMH to multiple-valued case, in particular, the ternary case is considered as well.

In [P2] a class of parametric transforms that are based on a unified representation of transform matrices in the form of sparse matrix products is described. Parametric transform allows controlling some of transform features by proper parameter selection. In our work, a method for parameter selection is proposed that allows synthesizing specific transforms with matrices containing predefined row(s). Furthermore, different families of transforms are defined within the introduced class of parametric transforms in [P2]. All transforms of one family can be computed with fast algorithms similar in structure to each other. In particular, the family of parametric Haar-like transforms has been introduced that consists of discrete orthogonal transforms of arbitrary order such that they all may be computed with a fast algorithm that is in structure similar to the classical fast Haar transform. Also the potential of the proposed class of Haar-like parametric transforms to improve the performance of fixed block transforms in image compression is investigated in [P2]. With this purpose, two image compression schemes are proposed where a number of Haar-like transforms are synthesized, each adapted to a certain set of blocks within an image.

In [P3] a signal-adaptive Haar-like transform, defined over the field GF(3), that is intended for processing ternary functions is presented. The proposed transform possesses a fast computation algorithm similar to the fast algorithms for the classical Haar transform on finite dyadic groups as well as the generalized Haar transforms for multiple-valued functions. The proposed transform is utilized in reduction of the number of nonzero coefficients in spectral representations of ternary functions.

The [P4] presents three methods for reduction of the number of nonzero coefficients in the spectra of binary vectors using newly introduced Reed-Muller Haar-like transform. The transform, used to transfer a given truth vector of a switching function into the spectral domain, is based on the Kronecker product of the Reed-Muller and binary parametric Haar-like transforms (BPHT) which is a signal adapted discrete binary valued transform that may be computed with a fast algorithm having the structure similar to that of classical fast Haar transform. The new hybrid transforms introduced in this paper combine properties of both the Reed-Muller and BPHT transforms, due to which they provide improved reduction of the number of nonzero coefficients in spectral representations of switching functions.

In [P5] the capability of parametric Haar-like transforms in 1D signal denoising application is explored. A new PHT based post-processing algorithm for 1D signal denoising is proposed which may be combined with another denoising method in order to improve the quality of the output signal. Here the basic wavelet thresholding based signal denoising method was complemented with the proposed post-processing algorithm.

The aim of work [P6] is to study a potential use of parametric Haar-like transforms in image denoising. A PHT-based post-processing method is proposed which may improve a denoising method based on fixed transforms. In particular, it is shown that the proposed method may significantly improve the performance of wavelet thresholding based image denoising.

Bibliography

[1] S. Agaian, ”Optimal Algorithms of Fast Orthogonal Transforms and their Implementation on Computers,” *Kibernetika i Vichislitelnaya Teknika*, issue 2, pp.231-319, Nauka, 1986 (in Russian).

[2] S. Agaian, D.Z. Gevorkian, “Synthesis of a class of orthogonal transforms. Parallel SIMD- algorithms and specialized processors,” *Pattern Recognition and Image Analysis*, vol.2, No4, pp.394-408, 1992.

[3] S. Agaian, K. Tourshan, J.P. Noonan, “Generalized Parametric Slant-Hadamard Transform,” *Signal Processing*, vol.84, issue 8, August, 2004, 1299-1306.

[4] S. Agaian, K. Tourshan, J.P. Noonan, "Parametrization of slant-Haar transforms," *IEE Proc. Vis. Image Signal Process* vol. 150, No. 5, 2003, pp.306-311.

[5] S. Agaian and A. Matevosyan, “Generalized Haar Transforms and Automation Systems for Testing Quality of Circuits,” *Acta Cybernetica*, Szeged, vol.5, pp.345-362, 1981.

[6] S. Agaian, J.T. Astola, and K. Egiazarian, *Binary Polynomial Transforms and Nonlinear Digital Filters*. Marcel Dekker, New York, 1995.

[7] S. Agaian, D. Gevorkian, “Complexity and Parallel Algorithms of Discrete Orthogonal Transforms,” Kibernetika i Vichislitelnaya Teknika, issue 4, pp.124-169, Moscow, Nauka,1988 (in Russian).

[8] H.C. Andrews, K.L. Caspary, “A Generalized Technique for Spectral Analysis,” *IEEE Transactions on Computers*, vol.19, No 1, pp. 16-25, 1970.

[9] N. Ahmed, T. Natarajan, and K.R. Rao, “Discrete Cosine Transform,” *IEEE Trans. Comput.*, vol.C-23,pp.89-93, 1974.

[10] N. Ahmed and K. Rao, Orthogonal Transforms for Digital Signal Processing, Springer-Verlag, 1975.

[11] N. Ahmed, M.C. Chen, “A Cooley-Tukey Algorithm for the Slant Transform,” *Int. Jour.of Computer Mathematics,* vol.5, issue1-4,1976 , pages 331 – 338.

[12] D. Akopian, M. Helsingius, J. Astola, “Multibase/Wavelet Transform Coding of Still Images Without Blocking Artifacts,” *32nd Asilomar Conf. on Signals, Systems & Computers*, vol.1, pp. 154-158, 1998.

[13] D. Akopian, M. Helsingius, J. Astola, “An Optimized Multiscanning Approach for Still Image Compression,” *IEEE International Workshop on Multimedia Signal Processing* (*MMSP*'99), September 13-15, 1999, Copenhagen, Denmark.

[14] J. Astola and D. Akopian, "Architecture oriented regular algorithms for a class of trigonometric transforms," *IEEE Transactions Signal Processing.*, April 1999, pp. 1109-1124.

[15] J. Astola, S. Minasyan, D.Guevorkian, “Multiple adaptive transform based image compression technique,” *Proc. IASTED Int. Conf. on Visualization, Imaging, and Image Processing, VIIP 2005,* Benidorm, Spain, Sept. 2005.

[16] J. Astola, L. Yaroslavsky, *Advances in Signal Transforms, Theory and Applications,* Hindawi Publishing, 2007.

[17] J. Astola, S. Agaian, D. Gevorkian, “Parallel Algorithms and Architectures for a Family of Haar-like Transforms,” *Proc. of SPIE’s Int. Symp. on Electronic Imaging: Science and Technology*, 2421, Feb., 1995, USA.

[18] J. Astola, S. Agaian, K.O. Egiazarian, D. Gevorkian, “Parallel Algorithms for Binary Polynomial Transforms and their Realization with Unified Series of Processors,” *Proceed. of SPIE’s Int. Symp. on Electronic Imaging: Science and Technology,* 2421, Feb., 1995, USA.

[19] J. Astola, R. Stankovic, Fundamentals of Switching Theory and Logic Design. A Hands On Approach, Springer, 2006.

[20] S.M. Atourian, D.Z. Gevorkian, K.O. Egiazarian, J.T. Astola, “Efficient Nonlinear Transform Methods for Image Processing,” *Journal of Electronic Imaging*, vol.5(3), July 1996, pp.323-334.

[21] S. Battiato, M. Mancuso, A. Bosco, M. Guarnera, “Psychovisual and Statistical optimization of quantization tables for DCT compression engines,” In Proc. of 11th Int. *Conf. on Image Analysis and Processing,* pp.602-606, Sept.2001.

[22] K.G. Beauchamp, *Walsh functions and their Applications*, Academic Press, London, 1975.

[23] K.G. Beauchamp, Applications of Walsh and related functions - with an introduction to sequency theory, Academic Press, 1984

[24] R.E. Bryant, “Graph-based algorithms for Boolean Functions Manipulation,” *IEEE Trans. Computers*, vol.C-35, n.8, 1986, pp.667-691.

[25] W.H. Chen, C.H. Smith, and S.C. Fralick, “A fast Computational algorithm for the discrete Cosine Transform,” *IEEE Trans. Communic.*, vol. COM-25, pp. 1004-1008, 1974.

[26] S.G. Chang, B. Yu, and M. Vetterli,“Adaptive wavelet thresholding for image denoising and compression”, *IEEE Trans. Image Proc.*, vol.9, pp. 1532-1546, Sep.2000.

[27] E.M. Clarke, K.L. McMillan, X. Zhao, M. Fujita, and J. Yang, “Spectral Transforms for Large Boolean Functions with Application to Technology Mapping,” In *Proc. of the 30th ACM/IEEE Design Automation Conference*, 1993.

[28] R.R. Coifman, D.L. Donoho,”Translation-invariant denoising”, In *Wavelets and Statistics. Lectures Notes in Statistics*, Springer Verlag, (1995), pp.125-150.

[29] J.W. Cooley, J.W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” *Math. Computation*, vol.19, pp.297-301, 1965.

[30] M.J. Corinthios, “3-D cellular arrays for parallel/cascade image/signal processing,” in Spectral Techniques and Fault Detection, (M. Karpovsky, ed.), New York, Academic Press, 1985.

[31] J. Granata, M. Conner, R. Tolimieri, “Recursive Fast Algorithms and the Role of the Tensor Product,” *IEEE Trans. Signal Processing*, vol.40, no.12, 1992.

[32] D. Guevorkian, “Computational Aspects of Discrete Linear Transforms and Rank Order Based Filters,” PhD Thesis, Finland, Tampere, 1997.

[33] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, “Image Denoising with block-matching and 3D filtering,” *Proc. of SPIE-IS&T Electronic Imaging, SPIE,* vol. 6064, 2006.

[34] G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw Hill, 1994.

[35] D.L. Donoho and I.M. Johnstone, “Ideal Denoising in an Orthonormal Basis Chosen from a Library of Bases”, Technical Report, Department of Statistics, Stanford University, 1994.

[36] D.L. Donoho, ”Denoising by soft thresholding”, *IEEE Trans. Inform. Theory,* vol.41,1995, pp. 613-627.

[37] D.L. Donoho and I.M. Johnstone, “Adapting to unknown smoothness via wavelet shrinkage,” Technical Report 425, Stanford University, June, 1993.

[38] P. Duhamel, H. Hollman, “Split-radix FFT algorithm,” *Electron. Lett.*, vol. 20, no.1, pp.14-16, 1984.

[39] K. Egiazarian, J. Astola, “New Algorithm for Removing of Mixed (White and Impulsive) Noise from Images,” *Proc. SPIE*, vol. 3646, pp. 78-89, *Part of the IS&T/SPIE Conf. on Nonlinear Image Processing X*, 1999.

[40] C. Fu, B. Falkowski, “Generation of multi-polarity Helix Transform over GF(3),” *IEICE Electron., Express* 1, pp. 211-216, 2004.

[41] B.J. Fino, V.R. Algazi, “Slant Haar transform,” *Proceedings of the IEEE*, vol.62,is.5, May 1974, pp.653 – 654.

[42] A. Ho T. S., X. Zhu, J. Shen, ”Slant transform watermarking for digital images, *Visual Communications and Image Processing 2003*. Proceed. of the SPIE, vol.5150, pp. 1912-1920 (2003).

[43] M. Helsingius and P. Kuosmanen, “Image Compression using multiple transforms” in Signal Processing: Image Communication, *Elsevier*, vol.15, issue 6, March 2000, pp.513-529.

[44] S. L. Hurst, "The Haar transform in digital network synthesis," *Proc. 11th Int. Symp. on* Multiple-valued Logic, 1981, pp.10-18.

[45] S. L. Hurst, D. M. Miller, J. C. Muzio, *Spectral Techniques in Digital Logic*, London, Academic Press, 1985.

[46] S. L. Hurst, *The Logical Processing of Digital Signals*, Crane, Russak & Company, Inc. New York, 1978.

[47] A. C. Hung and TH-Y Meng, "A Comparison of fast DCT algorithms,” *Multimedia Systems*, No.5, Vol.2, Dec. 1994.

[48] I. J. Good, “The interaction algorithm and practical Fourier analysis,” *Jour. of the Royal Statist.* Soc., B 20. 361-372, 1958.

[49] A. Jain, Fundamentals of Digital Image Processing, Prentice Hall.

[50] M. G. Karpovsky, (ed.), *Spectral Techniques and Fault Detection*. Academic Press, Orlando, Florida, 1985.

[51] M. Karpovsky, R. Stankovic, J. Astola, *Spectral Logic and its applications for the Design of Digital Devices*, A John Wiley & Sons, Inc, Publication, 2008.

[52] M. Karpovsky, Finite Orthogonal Series in the Design of Digital Devices, John Wiley, New York, 1976.

[53] V. G. Labunets, “Fast Transform Algorithms,” in *Primenenie Ortogonalnikh Metodov pri Obrabotke Signalov i Analize System*”, pp.4-14, Sverdlovsk, Ural Polytech. Institute.

[54] P.C. Mali and D. D. Majumder, “An analytical comparative study of a class of discrete linear basis tramsforms”, *IEEE Trans. Syst., Man & Cyber.* 24(3), pp.531-535, 1994.

[55] S. Mallat, A wavelet Tour of Signal Processing. Academic Press, 1999.

[56] H. Malvar, *Signal Processing with Lapped Transforms***,** Norwood, MA, Artech House, 1992.

[57] S. Minasyan, D. Guevorkian, H. Sarukhanyan, “ On parameterized fast Haar- and Hadamard-like transforms of arbitrary order,” in *Proc. 3rd Int.Conf. Comp.Science and Inform.Technologies* (CSIT’ 01), pp. 294–298, Yerevan, Armenia, September 2001.

[58] S. Minasyan, D. Guevorkian, S. Agaian, H. Sarukhanyan, “On ‘slant-like’ fast orthogonal transforms of arbitrary order,” in *Proc. of the 4th EURASIP IEEE Region and Int. Symp. on Video/Image Processing and Multimedia Communications* (*VIPromCom’02),* pp. 309–314, Zadar, Croatia, June 2002.

[59] S. Minasyan, J. Astola, D. Guevorkian, “An Image Compression Scheme Based on Parametric Haar- like Transform”, *Proc. of IEEE Int. Symposium on Circuits and Systems, ISCAS-2005,* Kobe, Japan, May, 2005.

[60] S. Minasyan, J. Astola, D. Guevorkian, “On unified architectures for synthesizing and implementation of fast parametric transforms”, *5th Int. Conf. on Information, Communications and Signal Processing, ICICS 2005,* Bangkok, Thailand.

[61] S. Minasyan, R. Stankovic, J. Astola, K. Egiazarian, ”Hybrid Reed-Muller Haar Transform and its Application in Reduction the Spectral Representations of Logic Functions,” *Int. Symposium on Multiple-Valued Logic* (*ISMVL’2007*), Texas, USA, May 22-24, 2007.

[62] P. Moulin, J. Liu, “Analysis of Multiresolution Image Denoising Schemes using Generalized Gaussian and Complexity Priors”, *IEEE Trans. on Information Theory,* vol.45, April,1999.

[63] H. Oktem, V. Katkovnik, K. Egiazarian, J. Astola, “Local Transform based Image Denoising with Varying Window Size,” *Proc. of Conf. on Image Processing,* Vol. 1, pp. 273-276, 2001.

[64] R. Oktem, L. Yaroslavskiy, K. Egiazarian, “Signal and Image Denoising in Transform Domain and Wavelet Shrinkage: A Comparative Study,” *Proceedings of EUSIPCO-98 European Signal Processing Conference,* vol.4, pp.2269-2272, Rhodes, Greece, 8-11,September, 1998.

[65] N. Ponomarenko, V. Lukin, K. Egiazarian, J. Astola, ”DCT-based High Quality Image Compression,” in *Proc. of 14th Scandinavian Conf. on Image Analysis*, Joensuu, Finland, pp.1177-1185, June,2005.

[66] P. Porwik, “The Spectral Test of the Boolean Function Linearity,” *Int. Jour. Appl. Math. Comput.Sciences*, 2003, vol.13,N.4, pp.567-575.

[67] W. K. Pratt, W. H, Chen and L. R. Welch, “Slant transform image coding”, *IEEE Trans.Communications*, vol.(OM-22), pp1075-1093,Aug,1974.

[68] W. Pennebaker, J. Mitchell, *JPEG Still Image Data Compression Standard*, New York, 1993.

[69] E. Pogossova, K. Egiazarian, “Reed-Muller Representation of Symmetric Functions, ” *Journ. of Multiple-Valued Logic and Soft Computing*, Old City Publishing, Inc., vol. 10, N1, 2004, pp. 51-72.

[70] R.L. Queiroz, P. Fleckenstein, “Very fast compression using hierarchical vector quantization,” *IEEE Signal Processing Letters*, 7(5), pp.97-99, May 2000.

[71] C.M. Rader, N.M. Brenner, “A new principle for fast Fourier transformation,” *IEEE Trans.*, 1976, ASSP-24, pp.264-265.

[72] K. R. Rao, P. Yip, Discrete Cosine Transform, Algorithms, Advantages, Applications, Academic Press Inc., 1990.

[73] K. Rao, A. Jalali, P. Yip, “Rationalized Hadamard-Haar Transform,” in Proc. *11st Ann. Asilomar Conf. Signals Syst. Comput.*, 1977, pp.194-203.

[74] S.M. Reddy, “Easily Testable Realizations for Logic Functions,” *IEEE Trans. on Computers*, pp. 1183-1188, 1972.

[75] D. Rusanovskyy, K. Egiazarian, “Video Denoising Algorithm in Sliding 3D DCT Domain,” ACIVC 2005, Belgium.

[76] A. Said, W. Pearlman, “A new fast and efficient image codec based upon set partitioning in hierarchical trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, June 1996, pp. 243-250.

[77] T. Sasao, *Switching Theory for Logic Synthesis.* Kluwer Academic Publishers, Boston, MA, USA, 1999.

[78] T. Sasao, “Easily Testable Realizations for Generalized Reed-Muller Expressions,” *IEEE Trans. on Computers*, vol.46, n.6, 1997.

[79] T. Sasao, M. Fujita, *Representations of Discrete Functions*, Kluwer Academic Publisher, 1996.

[80] A. Skodras, C. Christopoulos, T. Ebrahimi, “JPEG 2000 still image compression standard,” *IEEE Signal Process. Mag.*, vol.18, no.5, pp. 36-58, 2001.

[81] J. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients, *IEEE Transactions of Signal Processing*, vol.41, pp. 3445-3462, December, 1993.

[82] A.I. Solodovnikov, I.A. Kanatov, A.I. Spivakowskii, “Synthesis of Orthogonal Bases from a Generalized Spectral Kernel,” Voprosi *Teorii Sistem Avtomaticheskogo Upravlenija*, LGU, Leningrad,1978, issue 2, pp.99-112.

[83] A.I. Solodovnikov, “Synthesis of Complete Orthonormal Systems of Functions having fast Transform Algorithm,” in *Voprosy Teorii System Avtomaticheskogo Upravleniya*, LGU, Leningrad, 1978, issue 2, pp.99-112 (Russian).

[84] R. Stankovic, C. Moraga, “Design of Haar Wavelet Transforms and Haar spectral transform decision diagrams for multiple-valued functions,” *Proc. of 31 IEEE Int. Symp.on Multiple-valued Logic* (*ISMVL*’01), 2001, pp.311-316.

[85] H. V. Sorensen, M. T. Heideman, and C. S. Burrus, "On computing the split-radix FFT," *IEEE Trans. Acoust., Speech, Signal Processing* 34 (1), 152-156 (1986).

[86] D. Taubman, High Performance Scalable Image Compression with EBCOT, submitted to *IEEE Tran. IP*, Mar. 1999.

[87] J. Takala, D. Akopian, J. Astola, J. Saarinen, “Constant geometry algorithm for discrete cosine transform,” *IEEE Trans. on Signal Processing*, vol. 48, issue 6, 2000, pp.1840-1843.

[88] J. F. Tilki, A. A. Beex, “Image Data Compression Using Multiple Bases Representation”, *26th IEEE South.Symp. on System Theory* (*SSST*’94), Greece, pp.457- 461, 1994.

[89] M. A. Thornton, D. M. Miller, R. Drechler, *Spectral Techniques in VLSI CAD*, Kluwer Academic Publishers, 2001.

[90] A.M. Trahtman, V.A. Trahtman, *Osnovi Teorii Discrtetnih Signalov na Konechnih Intervalah*, Moscow, Sovetskoe Radio, 1975 (in Russian).

[91] B.-J. Yoon, P.P. Vaidyanathan, “Wavelet-based denoising by customized thresholding”, *ICASP*-2004.

[92] L.P. Yaroslavskiy, “Some Questions of the Theory of Discrete Orthogonal Transforms of Signals,” in *Tsifrovaya Obrabotka Signalov i ee Primeneniya*, Moscow, Nauka, 1981, pp. 33-71 (in Russian).

[93] L.P. Yaroslavskiy, “Local Adaptive Image Restoration and Enhancement with use of DFT and DCT in a Running Window,” *Proceeding, Wavelets Application in Signal and Image Processing IV, SPIE Proc. Series*, v.2825, pp.1-13, August, Colorado, 1996.

[94] M. Vetterli and H. J. Nussbaumer, “Simple FFT and DCT algorithms with reduced number of operations,” Signal Processing, vol.6, no.4, pp.267-278, 1984.

[95] Z. Wang, “Prunning the Fast Discrete Cosine Transform,” *IEEE Trans. on Communic.*, vol.39, N.5, 1991.

[96] S. Winograd, “On Computing the discrete Fourier transform,” *Mathematics of Computation*, vol.32, no.1, pp.175-199, 1978.

[97] G. K. Wallace, “The JPEG still image compression standard”, *IEEE Trans. Consum. Electronics*, vol.38, no.1, pp.xviii-xxxiv, 1992.

[98] J.Xie, S.Agaian,J. Noonan, “Digital watermarking in parametric slant transform domain,” *Proc. of SPIE on Multimedia on Mobile Devices*, vol.6821, pp. 68210C-68210C-8,2008.

[99] X. Xiong, O. Guleryuz, M.Orchard, Z. Xiong, “A DCT-based embedded image coder,” *IEEE Signal Processing Letters*, 3, pp.289-290, 1996.

[100] L. Zalmanson, ”Fourier, Walsh and Haar transforms and their application in control, communication and other fields, Moscow, Nauka, 1989.