Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Telecomunicações
DECOM-FEEC-UNICAMP

Contributions in Image and Video Coding

(Contribuições em Codificação de Imagens e Vídeo)

Autora: Vanessa Testoni

Orientador: Prof. Dr. Max Henrique Machado Costa

Tese submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do título de

Doutora em Engenharia Elétrica.

Área de concentração: Telecomunicações e Telemática

Comissão Julgadora:

Prof. Dr. Max Henrique Machado Costa - FEEC/UNICAMP

Prof. Dr. Dalton Soares Arantes - FEEC/UNICAMP

Prof. Dr. Yuzo Iano - FEEC/UNICAMP

Prof. Dr. Nelson Delfino d'Ávila Mascarenhas - UFSCAR

Prof. Dr. Rui Seara - UFSC

Campinas, 25 de novembro de 2011.

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

T289c

Testoni, Vanessa
Contribuições em codificação de imagens e vídeo / Vanessa Testoni. --Campinas, SP: [s.n.], 2011.

Orientador: Max Henrique Machado Costa.
Tese de Doutorado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Compressão de imagens - Codificação. 2. Videodigital - Codificação. 3. Telecomunicações - Codificação. I. Costa, Max Henrique Machado. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Contributions in image and video coding
Palavras-chave em Inglês: Image compression - Coding, Video digital - Coding, Telecommunications - Coding
Área de concentração: Telecomunicações e Telemática
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Dalton Soares Arantes, Yuzo Iano, Nelson Delfino d'Ávila Mascarenhas, Rui Seara
Data da defesa: 25-11-2011
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidata: Vanessa Testoni

Data da Defesa: 25 de novembro de 2011

Título da Tese: "Contributions in Image and video Coding (Contribuições em Codificação de Imagens e Vídeo) "

Prof. Dr. Max Henrique Machado Costa (Presidente): _____

Prof. Dr. Nelson Delfino d'Ávila Mascarenhas: _____

Prof. Dr. Rui Seara: _____

Prof. Dr. Dalton Soares Arantes: _____

Prof. Dr. Yuzo Iano: _____

Abstract

The image and video coding community has often been working on new advances that go beyond traditional image and video architectures. This work is a set of contributions to various topics that have received increasing attention from researchers in the community, namely, scalable coding, low-complexity coding for portable devices, multiview video coding and run-time adaptive coding.

The first contribution studies the performance of three fast block-based 3-D transforms in a low complexity video codec. The codec has received the name Fast Embedded Video Codec (FEVC). New implementation methods and scanning orders are proposed for the transforms. The 3-D coefficients are encoded bit-plane by bit-plane by entropy coders, producing a fully embedded output bitstream. All implementation is performed using 16-bit integer arithmetic. Only additions and bit shifts are necessary, thus lowering computational complexity. Even with these constraints, reasonable rate versus distortion performance can be achieved and the encoding time is significantly smaller (around 160 times) when compared to the H.264/AVC standard.

The second contribution is the optimization of a recent approach proposed for multiview video coding in videoconferencing applications or other similar unicast-like applications. The target scenario in this approach is providing realistic 3-D video with free viewpoint video at good compression rates. To achieve such an objective, weights are computed for each view and mapped into quantization parameters. In this work, the previously proposed ad-hoc mapping between weights and quantization parameters is shown to be quasi-optimum for a Gaussian source and an optimum mapping is derived for a typical video source.

The third contribution exploits several strategies for adaptive scanning of transform coefficients in the JPEG XR standard. The original global adaptive scanning order applied in JPEG XR is compared with the localized and hybrid scanning methods proposed in this work. These new

orders do not require changes in either the other coding and decoding stages or in the bitstream definition.

The fourth and last contribution proposes an hierarchical signal dependent block-based transform. Hierarchical transforms usually exploit the residual cross-level information at the entropy coding step, but not at the transform step. The transform proposed in this work is an energy compaction technique that can also exploit these cross-resolution-level structural similarities. The core idea of the technique is to include in the hierarchical transform a number of adaptive basis functions derived from the lower resolution of the signal. A full image codec is developed in order to measure the performance of the new transform and the obtained results are discussed in this work.

Keywords: adaptive coefficient scanning, fast video codec, hierarchical 3-D block-based transform, multiview video coding, signal dependent transform.

Resumo

A comunidade de codificação de imagens e vídeo vem também trabalhando em inovações que vão além das tradicionais técnicas de codificação de imagens e vídeo. Este trabalho é um conjunto de contribuições a vários tópicos que têm recebido crescente interesse de pesquisadores na comunidade, nominalmente, codificação escalável, codificação de baixa complexidade para dispositivos móveis, codificação de vídeo de múltiplas vistas e codificação adaptativa em tempo real.

A primeira contribuição estuda o desempenho de três transformadas 3-D rápidas por blocos em um codificador de vídeo de baixa complexidade. O codificador recebeu o nome de Fast Embedded Video Codec (FEVC). Novos métodos de implementação e ordens de varredura são propostos para as transformadas. Os coeficiente 3-D são codificados por planos de bits pelos codificadores de entropia, produzindo um fluxo de bits (bitstream) de saída totalmente embutida. Todas as implementações são feitas usando arquitetura com aritmética inteira de 16 bits. Somente adições e deslocamentos de bits são necessários, o que reduz a complexidade computacional. Mesmo com essas restrições, um bom desempenho em termos de taxa de bits versus distorção pôde ser obtido e os tempos de codificação são significativamente menores (em torno de 160 vezes) quando comparados ao padrão H.264/AVC.

A segunda contribuição é a otimização de uma recente abordagem proposta para codificação de vídeo de múltiplas vistas em aplicações de video-conferência e outras aplicações do tipo "unicast" similares. O cenário alvo nessa abordagem é fornecer vídeo com percepção real em 3-D e ponto de vista livre a boas taxas de compressão. Para atingir tal objetivo, pesos são atribuídos a cada vista e mapeados em parâmetros de quantização. Neste trabalho, o mapeamento ad-hoc anteriormente proposto entre pesos e parâmetros de quantização é mostrado ser quase-ótimo para uma fonte Gaussiana e um mapeamento ótimo é derivado para fonte típicas de vídeo.

A terceira contribuição explora várias estratégias para varredura adaptativa dos coeficientes da transformada no padrão JPEG XR. A ordem de varredura original, global e adaptativa do JPEG XR é comparada com os métodos de varredura localizados e híbridos propostos neste trabalho. Essas novas ordens não requerem mudanças nem nos outros estágios de codificação e decodificação, nem na definição da bitstream.

A quarta e última contribuição propõe uma transformada por blocos dependente do sinal. As transformadas hierárquicas usualmente exploram a informação residual entre os níveis no estágio da codificação de entropia, mas não no estágio da transformada. A transformada proposta neste trabalho é uma técnica de compactação de energia que também explora as similaridades estruturais entre os níveis de resolução. A ideia central da técnica é incluir na transformada hierárquica um número de funções de base adaptativas derivadas da resolução menor do sinal. Um codificador de imagens completo foi desenvolvido para medir o desempenho da nova transformada e os resultados obtidos são discutidos neste trabalho.

Palavras-chave: varredura adaptativa de coeficientes, codificador de vídeo rápido, transformada 3-D hierárquica por blocos, codificação de vídeo de múltiplas vistas, transformada dependente do sinal.

"But life is short and information endless...

Abbreviation is a necessary evil and the abbreviator's business is to make the best of a job which, although intrinsically bad, is still better than nothing."

Aldous Huxley - Brave New World Revisited, 1958

Acknowledgements

I am deeply grateful to my advisor, Max Costa, for his guidance, encouragement, depth of knowledge, and for always believing in my potential. He truly opened me the doors of the research world and provided me the opportunity to understand and live it in a great level of completeness and intensity.

Three of the four contributions presented here are collaborative works with Darko Kirovski and Dinei Florêncio, who are both researchers at Microsoft Research (MSR). These contributions were all initiated during my internships at MSR, where Darko and Dinei were my mentors, and became a key part in my formation as a researcher. Therefore, I am also very grateful to them. Darko really acted as a mentor for me in the last years. He was always teaching me Matlab and LaTeX commands, but was much more worried about teaching me how to properly think as a researcher, how to make the right questions, how to choose research projects and how to plan my research career. His strong belief in the importance of being creative and expanding my research horizons in other areas of the signal processing world gave me the opportunity to work in additional interesting projects not included in this thesis, but also very enriching to my PhD experience. Two of the four contributions presented here were developed through a very close work with Dinei. He certainly taught me much more about image and video coding than he can imagine. His strong and endless support was also fundamental for the practical aspects of the development of this thesis, once also included a grant for my research laboratory at Unicamp.

I also would like to thank all the members of ComL@b, specially to Fábio Lumertz Garcia, Veruska Rodrigues and Cláudio Ferreira Dias. Their support and friendship were incredible valuable to me, mainly in the final months of this work. I am also grateful to my university, Unicamp, for providing a dynamic and free environment that enabled me the opportunity to

develop several enriching activities other than research, such as initiating and co-organizing the first local signal processing symposium and founding the IEEE Women in Engineering group.

In addition, I would like to thank my parents and my grandmother. My mother, Magáli Ana Sovierzoski, placed the love for the knowledge in my heart and the importance of it in my brain even before I had any sense of what the word knowledge meant. My father, Walmor Testoni, early introduced me to his so-loved Math world, which would also quickly become my favorite world. My grandmother, Eleuter Dynarowski Sovierzoski, had a fundamental contribution in all aspects of my life and enabled my dreams to come true.

Finally, my most heartfelt gratitude goes to the two most special people in my life. My sister, Sheila Testoni da Rocha, has always supported me in thousand ways. She is the best gift I could have won in this life. I gave her the unique opportunity to perfect her skills of putting all my huge incredible hard problems into perspective during these PhD times. And she did great! My partner, Dr. Alexandre, the Great, was by my side since the beginning of my doctoral studies, while also pursuing his own PhD. I could not have asked for a better partner in this grand adventure of the PhD, which was certainly filled with strong emotions for both of us. I thank them both for always being there and believing in me. This work is dedicated to them.

This research was supported in part by a grant from CAPES (Ministry of Education, Brazil) and in part by a Microsoft Research PhD fellowship.

Contents

List of Figures

List of Tables

List of Symbols

ME Motion Estimation

MC Motion Compensation

DCT Discrete Cosine Transform

DHT Discrete Haar Transform

PLHaar Piecewise-Linear Haar

FEVC Fast Embedded Video Codec

YC_oC_g Luminance Offset Orange Offset Green

QCIF Quarter Common Intermediate Format

TCG Transform Coding Gain

QP Quantization Parameter

BPP Bits per pixel

PSNR Peak Signal-to-Noise Ratio

FVV Free Viewpoint Video

MVC Multiview Video Coding

GOP Group of Pictures

MVD Multiview Video plus Depth

JVT Joint Video Team

JPEG XR Joint Photographic Experts Group Extended Range

HDR High Dynamic Range

ROI Region of Interest

POT Photo Overlap Transform

PCT Photo Core Transform

HSDT Hierarchical Signal Dependent Transform

EZW Embedded Zerotrees Wavelet

KLT Karhunen-loève Transform

LAS Local Activity Spectrum

SASI Statistical Analysis of Structural Information

LBP Local Binary Pattern

PCA Principal Components Analysis

SAD Sum of Absolute Differences

Author's Publications

Testoni, V. and Costa, M. H. M.(2011). Block-based 3-D Fast Transforms applied to an Embedded Color Video Codec, *Proceedings of the International Workshop on Telecommunications (IWT) - Rio de Janeiro, Brazil*.

Testoni, V. and Costa, M. H. M.(2010). Entropy Coders and 3-D Hadamard Coefficients Sequency Scan Order for a Fast Embedded Color Video Codec, *Elsevier's Computers & Electrical Engineering Journal, Vol. 36, No. 4, pp. 676–690*.

Testoni, V. and Kirovski, D.(2010). On the Inversion of Biometric Templates by an Example, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Dallas, USA*.

Testoni, V.; Costa, M. H. M; Kirovski, D. and Malvar, H.(2010). On the Adaptive Coefficient Scanning of JPEG XR/HD Photo, *Proceedings of the Data Compression Conference (DCC), Snowbird, USA, pp. 69–78*.

Testoni, V.; Florêncio, D. and Costa, M. H. M..(2009). Optimizing QPs for Multiview Image Coding for Free Viewpoint Video, *Proceedings of XXVII Brazilian Symposium on Telecommunications (SBrT), Blumenau, Brazil*.

Testoni, V. and Costa, M. H. M.(2008). Three-Dimensional Transforms and Entropy Coders for a Fast Embedded Color Video Codec, *Proceedings of XXI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), Campo Grande, Brazil, pp. 147–154*.

Testoni, V. and Costa, M. H. M.(2007). 3D-Hadamard Coefficients Sequency Scan Order for a Fast Embedded Color Video Codec, *Proceedings of the International Conference on Signal Processing and Communication Systems (ICSPCS), Gold Coast, Australia, pp. 75–82*.

Testoni, V. and Costa, M. H. M.(2007). Fast Embedded 3D-Hadamard Color Video Codec, *Proceedings of XXV Brazilian Symposium on Telecommunications (SBrT), Recife, Brazil*.

Chapter 1

Introduction

Digital image and video are everywhere and usually require a large amount of storage or transmission capacity. Therefore, video and image coding is essential for any application in which storage capacity or transmission bandwidth is constrained. Almost all consumer applications developed by the digital image and video related industry fall into this category. This industry is huge and is still in expansion. A few examples of such applications are: digital television broadcasting, internet video streaming, tele-conferencing, multiview video, high dynamic range imaging, mobile video and image streaming, video calling and image region-of-interest extraction for security and medical applications.

Pushed by some increasingly innovative and demanding applications, image and video codecs are always evolving and aggregating new features. At the time of this writing, the H.264/AVC standard is still considered the state of the art video codec, while the JPEG XR standard is the emerging state of the art image codec. According to [1], in 2010, almost 70% of the videos online were already being coded with H.264. All YouTube videos are available in H.264 and YouTube alone represents 40% of all videos online. The successor of H.264, named High Efficiency Video Coding (HEVC), is expected to be published as an international standard in 2013 and its current development is in line with the expected online usage models. On the other hand, the image coding scenario is not so well defined. The JPEG 2000 standard, although efficient, did not establish itself as the appropriate JPEG successor. Therefore, JPEG XR was standardized in 2009 with the purpose of addressing the limitation of previous formats, while achieving high image quality and low bit-rates. The format supports High Dynamic Range (HDR) imaging for a new generation of digital cameras and other imaging applications, such as interactive online imaging. Other JPEG XR features also makes it suitable for efficient design of hardware implementation, memory buffers and region-of-interest (ROI) coding applications.

As the state of art codecs do not always evolve as fast as new application needs arise, the image and video coding research community has often been working to fill in the gaps left by the standards and to push codec evolution. This thesis is a set of contributions to various topics that have received attention from researchers in the community. Each chapter handles a different problem and the contributions, although all related to image and video coding, are heterogeneous. Therefore, each chapter can be read independently. Just a brief overview of them is presented in the sequel, as each chapter is already equipped with introduction.

The second chapter brings a contribution to the development of low-complexity video codecs for portable devices. It studies the performance of a set of fast block-based 3-D transforms in a low-complexity video codec application. New implementation methods and scanning orders are proposed for the transforms.

The third chapter deals with the problem of achieving efficient multiview video coding to provide free viewpoint video. It examines the optimized mapping between pixel weights and quantization parameters to reduce transmission requirements in multiview coding.

The fourth chapter proposes a contribution to the JPEG XR standard by studying an adaptive coefficient scanning order scheme that seeks to exploit and highlight the scalable (progressive reconstruction) characteristics of the standard.

The fifth chapter investigates a new adaptive and scalable transform, that is intended to exploit the residual redundancies across hierarchical decomposition levels.

As each chapter already contains its own conclusions, the sixth chapter completes the thesis with possible extensions, improvements and future work proposals for these topics.

Chapter 2

FEVC - Fast 3-D Embedded Video Codec

2.1 Introduction

Research in video coding systems typically looks for techniques that can reach the highest possible compression rate while not exceeding a given level of distortion. Increased compression rate is generally achieved by means of higher coding complexity, which is supported by increased availability of computational power. However, in some video coding and transmission applications, i.e., portable digital applications like smartphones and digital video cameras, the use of higher performance processors is not convenient or cost-effective and some of the critical issues may be low complexity implementation, low power consumption and restricted computational resources. Also, in some applications, the codecs need to be implemented by software only, as hardware implementation may not be of convenience. Although there is significant research in reducing the computation for H.264, one of the leading compression standards, the complexity is still high for the requirements of some of these applications.

In order to reduce the video codecs computational complexity, three-dimensional (3-D) transforms have been investigated by many researchers as an alternative approach to fully avoid the very efficient but time-consuming motion estimation (ME) and motion compensation (MC) techniques. This work studies the performance of some of the simpler and faster block-based 3-D transforms. The compared transforms are Hadamard (4x4x4 and 8x8x8), H.264/AVC integer DCT-like (4x4x4) and Piecewise-Linear Haar - PLHaar (4x4x4 and 8x8x8). Although being a wavelet-like transform, the PLHaar is also applied here in a block-based fashion.

The environment where the performance of the block-based 3-D transforms is being compared is the color video codec named FEVC (Fast Embedded Video Codec). The FEVC was firstly described at the author's MSc dissertation [2]. The codec is designed in C# language, especially

to be executed in a video set-top box device. A large number of these set-top boxes are projected to be used as the interface between a fiber optical network and its users. This device will process digital signals to extract video, audio and data information and send that information to an output device. Among functions such as Internet accessibility and voice over IP, the set-top box is conceived to support video on demand and video conference applications. Due to cost and scale restrictions, the video set-top box is to be designed with integer based processor and low cache memory requirements. Because of this, the video codec implementation is designed to avoid multiplication and division operations and to make efficient use of available memory. All multiplication and division operations are carried out by binary shifts, equivalent to multiplication or division by integer powers of two. Moreover, the system is implemented exclusively with 16-bit integer arithmetic, which also requires some approximations. The errors introduced by these approximations can be compensated for some reduction in the compression rate. This is acceptable, as the video codec is intended for high capacity optical links.

Due to its purpose, the FEVC is focused on reduced execution times and is less concerned with high compression performance. Under such conditions, not only the 3-D transforms were chosen to be fast and simple, but all other codec stages (i.e., entropy coding) were designed to be computationally efficient. All codec stages that compound the FEVC structure are shown in Fig. 2.1. Initially, only the Hadamard 3-D transform was considered and the codec was named Fast Hadamard Video Codec (FHVC). The work presented here describes the following improvements on this video codec: a new coefficient scanning method for the Hadamard 3-D transform, the addition of the 3-D transforms H.264/AVC integer DCT-like and PLHaar with innovative 3-D implementations and coefficient scanning methods, and the study of an alternative Golomb entropy encoder. All these new contributions are partially published in [3–7].

Each codec stage shown in Fig. 2.1 is detailed in the subsequent sections of this chapter. The color space conversion, which is the first codec stage, is described in Section 2.2. The second and third codec stage are jointly presented in Section 2.3, once each 3-D transform has its own optimized coefficient scanning method. And, finally, the fourth codec stage, which is dedicated to the entropy coding, is explained in Section 2.4. The objective of each codec stage is to respectively reduce the intrinsic visual, spatial, temporal and entropic redundancies of raw video sequences. The explanation of each video redundancy and some alternative works in the area literature that

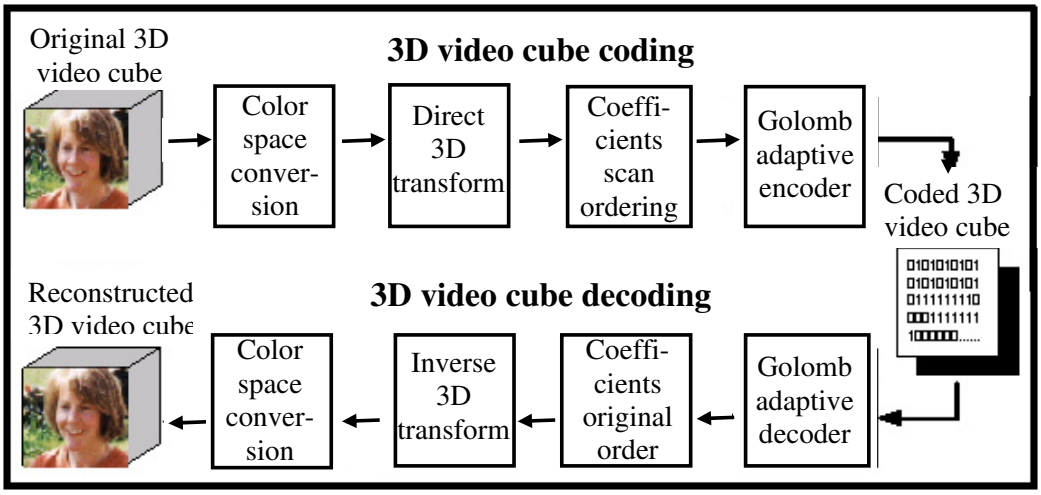


Figure 2.1: FEVC block diagram.

aim to reduce them are included in [2]. Sections 2.5 and 2.6 present the overall implementation results and conclude the chapter.

2.2 Video Codec Color Spaces

The FEVC is able to read color video sequences stored in tristimulus color space, such as RGB and YUV 4:2:0[1]. Each such a color plane is encoded separately and the allowed pixel bit-rate is divided into the color planes according to its significance. So, for the RGB format, the pixel bit-rate is equally divided, but in the YUV 4:2:0 format, the luminance plane receives more bits than the chrominance planes (because the U and V chrominance planes are one-fourth the size of the luminance Y plane). In the FEVC, only approximately 10% of the luminance rate is spent on the chrominance signal. This simple weighted bit-rate allocation procedure allows achieving higher compression rates.

In order to get the well-known advantages of the L-C (Luminance - Chrominance) formats, it is possible to convert an original RGB video sequence to a different internal color space (such as YC_oC_g) before beginning the coding process. Other color spaces are also supported by the FEVC and the conversions among them, described in [8], are also implemented.

[1] A detailed explanation about the whole human visual system (including the advantages of the L-C formats) and the video patterns (such as YUV 4:2:0) is presented in [2].

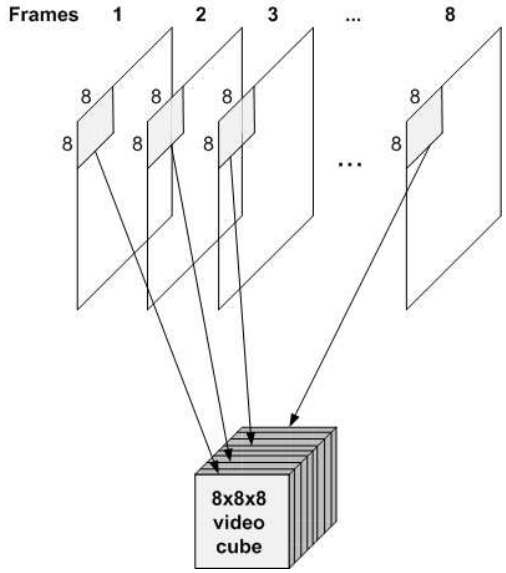


Figure 2.2: The composition of one 8x8x8 video cube.

2.3 Block-based Three-dimensional Fast Transforms

The FEVC applies 3-D transforms to reduce redundancies in both spatial and temporal dimensions. The video sequence being encoded is partitioned into cubes, as shown in Fig. 2.2, and the transform is separately applied per cube dimension (first to columns, then to rows and finally to frames).

The FEVC implements the following fast transforms in a block-based fashion: Hadamard, H.264/AVC integer DCT-like and PLHaar. The dynamic range gain, the transform coding gain (TCG), the energy concentration capacity and the coefficients scanning method of each one of these transforms are described in the next sections. These transforms were chosen because they can be computed exactly in integer arithmetic, thus avoiding inverse transform mismatch problems. Furthermore, only additions and bit shifts are necessary, thus minimizing computational complexity.

To evaluate the cube's size effect in coding performance, cubes of 4x4x4 or 8x8x8 sizes are used for the Hadamard transform and for the PLHaar transform. Only cubes of 4x4x4 size are used for the H.264/AVC integer DCT-like transform because of the higher complexity of the matrix of its 8x8x8 transform.

2.3.1 ***3-D Hadamard Transform***

The Hadamard transform was implemented in the FEVC for practical reasons, because it has the simplest basis functions (composed only of $+1$ and -1 elements) and it is identical to its inverse. The transform computations do not require multiplications [9], as a result of the binary transform kernel. The Hadamard transform matrix H_n of order N is a squared matrix, where $N = 2^n$ for some integer n. H_n can be generated starting with the core matrix

$$H_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.1}$$

and applying the Kronecker product recursion

$$\begin{aligned} H_n &= H_{n-1} \oplus H_1 = H_1 \oplus H_{n-1} \\ &= \frac{1}{\sqrt{2^n}} \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}. \end{aligned} \tag{2.2}$$

As can be noticed in Eq. (2.2) it is easy to extend smaller transform sizes to larger ones, such as 8x8x8 or greater. As an example, the Hadamard matrix H_3 for $N = 8$ is given by

$$H_3 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{matrix} \textbf{\textit{Sequency}} \\ \mathbf{0} \\ \mathbf{7} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{1} \\ \mathbf{6} \\ \mathbf{2} \\ \mathbf{5} \end{matrix} . \tag{2.3}$$

The basis vectors of the Hadamard transform can also be generated by sampling the class of functions called Walsh functions shown in Fig. 2.3. These functions also take on the binary values ± 1, exclusively, and form a complete orthonormal basis for square integrable functions.

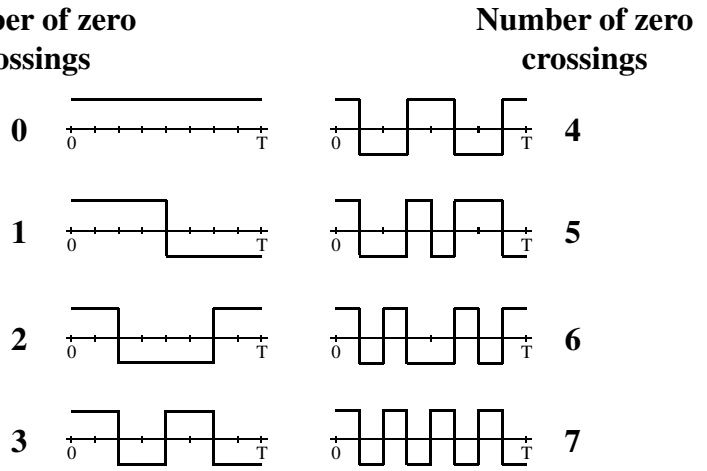


Figure 2.3: Walsh functions sampled to the Hadamard matrix generation.

The number of zero crossings of a Walsh function or the number of transitions in a basis vector of the Hadamard transform denotes its "sequency". This is a similar concept to the notion of frequency of sinusoidal signals, also related to zero crossings.

In the Hadamard matrix generated by the recursion in Eq. (2.2), the row vectors are not sequency ordered, as shown in Eq. (2.3) by the column labeled ***Sequency***. The resulting sequency order of these vectors is called the *Hadamard order* because it is the order in which the transform coefficients are generated.

Several Hadamard transform fast calculation methods are available in the literature. The method presented in [9] was chosen to be implemented in the FEVC because it is based on the fact that the H_n matrix can be written as a product of $log_2 N$ or n sparse matrices $\widetilde{H}$. Each multiplication by $\widetilde{H}$ implies the execution of N additions or subtractions. As this multiplication is repeated $log_2 N$ times, the total number of operations is of the order of $N \times log_2 N$, a significant reduction with respect to the N^2 operations involved in a straight matrix multiplication operation.

2.3.1.1 Dynamic Range Gain

As the Hadamard transform matrix H_n shown in Eq. (2.2) is composed only of $+1$ and -1 values, the one-dimensional transform has a dynamic range gain of $N/\sqrt{N} = \sqrt{N}$, where $N = 2^n$. If $N = 8$, for instance, the dynamic range gain is $\sqrt{8}$ and for the three-dimensional transform, the total dynamic range gain is $8^3/\left(\sqrt{8}\right)^3 = 8\sqrt{8}$.

The division by $\sqrt{N}$ is done in each cube dimension to preserve the signal energy in the transform domain. Similarly, the divisions must be performed in the decoder because the same

transform is used in the inverse operation, as the Hadamard transform is real, symmetric and orthogonal. In order to avoid the square root operation, the fractional coefficients generated by these divisions and to reduce the coefficient magnitudes resulting from the three Hadamard transform calculations, a new 3-D implementation method is proposed in [2].

In this new approach, the Hadamard transform is initially applied to the cube columns and, subsequently, to the cube rows. These two Hadamard transform calculations require two divisions of the coefficients by $\sqrt{N}$, but when performed jointly[2], only one division by N is required. Since the supported values for N are powers of 2, this division by N can be implemented through binary shifts. Therefore, after the two Hadamard transform calculations, the coefficients are right binary shifted by $log_2 N$ positions.

After the application of the Hadamard transform in the cube columns and rows, the transform must be applied finally to the cube frames. This application requires a new division of the coefficients by $\sqrt{N}$. However, this is the last Hadamard transform calculation done during the coding process. There is no other division term to be grouped with this division in order to remove the square root operation. This problem is solved in [2] by transferring to the encoder the first of the Hadamard transform calculations performed by the decoder. This can be done because the same Hadamard transform is applied by the decoder and the first decoding operation is the coefficient division by $\sqrt{N}$.

Two alternative ways to remove the square root normalization operations of Hadamard calculations are grouping all divisions by $\sqrt{N}$ operations and performing them before all three transform calculations, or grouping the divisions and performing them right after all three transform calculations. These options can not be applied because the intermediary results often exceed the limits of the 16-bit integer coefficient representation, to the left or to the right.

Conversely, in the decoding process, the Hadamard transform is applied, respectively, to the cube frames, rows and columns and, in the end, the recovered pixel values just must be shifted by $log_2 N$ positions to the right.

In order to illustrate the conventional and the order of operations proposed in [2], the conventional coding and decoding 3-D Hadamard transforms are shown, respectively, by the left

[2]The division operations can be grouped because Hadamard is a linear transform.

and right sides of Eq. (2.4), where *PC* means *Pixels Cube*, *CC* means *Coefficients Cube* and $H' = \sqrt{N} \times H_n$.

$$\underbrace{\tfrac{1}{\sqrt{N}}\times\left(H'\left(\underbrace{\tfrac{1}{\sqrt{N}}\times\left(H'\left(\underbrace{\tfrac{1}{\sqrt{N}}\times\left(H'(PC)\right)}_{columns}\right)\right)}_{rows}\right)\right)}_{frames} \equiv \underbrace{\tfrac{1}{\sqrt{N}}\times\left(H'\left(\underbrace{\tfrac{1}{\sqrt{N}}\times\left(H'\left(\underbrace{\tfrac{1}{\sqrt{N}}\times\left(H'(CC)\right)}_{frames}\right)\right)}_{rows}\right)\right)}_{columns}. \quad (2.4)$$

The same transforms implemented according to the method proposed in [2] are shown by Eq. (2.5).

$$\underbrace{\tfrac{1}{\sqrt{N}}\times\tfrac{1}{\sqrt{N}}\left(H'\left(\underbrace{\tfrac{1}{\sqrt{N}}\times\tfrac{1}{\sqrt{N}}\left(H'\left(\underbrace{H'(PC)}_{columns}\right)\right)}_{rows}\right)\right)}_{shifts} \equiv \underbrace{\tfrac{1}{\sqrt{N}}\times\tfrac{1}{\sqrt{N}}\left(H'\left(\underbrace{H'\left(\underbrace{H'(CC)}_{frames}\right)}_{rows}\right)\right)}_{columns}. \quad (2.5)$$

As the first decoding division by $\sqrt{N}$ was carried out at the encoder, the total dynamic range gain becomes $8^3/\left(\sqrt{8}\right)^4 = 8$. Since $log_2 8 = 3$, only 3 additional bits are required to store the transform coefficients. Therefore, the total dynamic range gain was reduced from $8 \times \sqrt{8}$ to only 8. This analysis for $N = 8$ is sufficient for the Hadamard transform because the maximum supported cube size in the FEVC is 8x8x8.

2.3.1.2 Transform Coding Gain

The transform coding gain (TCG) is a measure of the transform energy compaction which is defined as the ratio between the arithmetic mean and the geometric mean of the variances σ_i^2 of

all components in the transformed vector [10] as shown by

$$TCG = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{\frac{1}{N}}}. \tag{2.6}$$

From a compression standpoint, for a stationary Gauss-Markov input with correlation coefficient $\rho = 0.9$, the one-dimensional TCG of the 4x4 Hadamard matrix H_2 is 5.03 dB.

A comparative analysis of the Hadamard coding gain with other transforms is provided in Section 2.3.2.2.

2.3.1.3 Energy Concentration and Coefficients Scanning

The variances of the transform coefficients, and therefore the signal energy associated with these coefficients should be arranged in a "decreasing in the average" order to increase the efficiency of the entropy coding stage (which is the next stage in the coding process) and to support an embedded progressive encoder. This requires an appropriate scanning order for the transform coefficients, according to the expected energy distribution pattern. Deterministic and specific scanning orders give generally better results than the traditional 3-D zig-zag scanning.

To determine a fast and fixed scanning order (independent of the cube information content), an analysis was made of several video sequences and an approximately common spreading energy pattern was identified. As expected, the energy tends to be concentrated in the DC coefficient [corresponding to the 0 sequency in Eq. (2.3)] and in the AC coefficients of low sequency numbers (from 1 to 4). The other AC coefficients are associated with higher sequency numbers and tend to have smaller energy values. This energy distribution pattern can be seen in Fig. 2.4, where the gray level is proportional to the coefficient energy concentration.

Therefore, to ensure that the coefficients with the higher energy values will be scanned first, it is necessary to take into account the three sequency numbers of each coefficient (one sequency number for each cube dimension).

The method proposed here for the appropriate consideration of the three sequency numbers of each coefficient is to consider the coefficients in the increasing order of the product of their

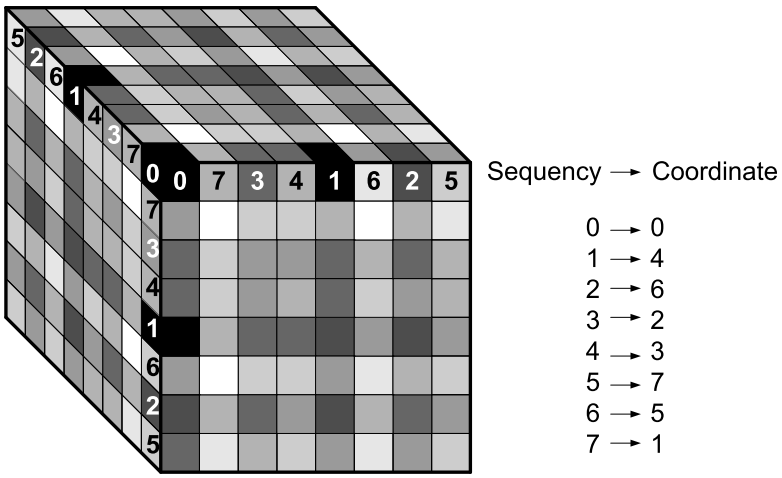


Figure 2.4: Translation of the sequency numbers to the cube coordinates.

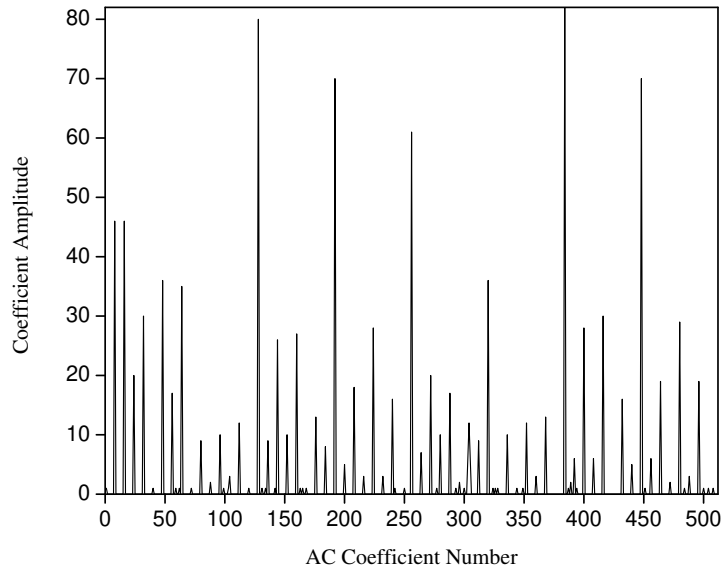
three sequency numbers, each added by one. Since the sequency number can vary from 0 to 7, the multiplication of the incremented sequency numbers varies from 1x1x1 = 1 to 8x8x8 = 512.

Additionally, to provide a deterministic rule for the ordering of coefficients with the same product value, the order of the triples composed by the incremented sequency numbers of frame, row, and column is considered. For instance, the product value of 8 is associated with the triples $(1, 2, 4)$, $(1, 4, 2)$, $(2, 2, 2)$, $(2, 4, 1)$, $(4, 1, 2)$, and $(4, 2, 1)$, and they will be considered by the encoder in this order. This is done because the coefficients energy is concentrated first in the low frame sequency numbers, then in the low row sequency numbers, and finally in the low column sequency numbers.

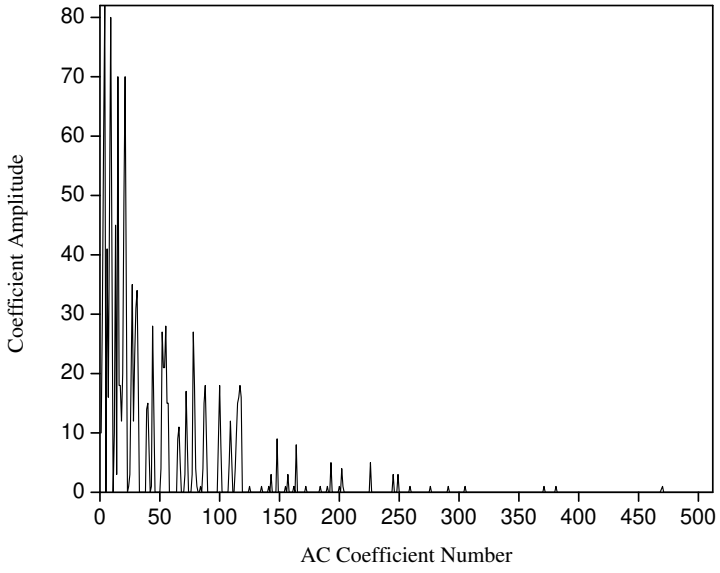
The last procedure to complete the scanning stage is to translate the sequency numbers to the real cube coordinates. This translation is shown in Fig. 2.4 and it is necessary because the *Hadamard order* differs from the *sequency* order, as explained at the beginning of Section 2.3.1.

In order to illustrate the cube coefficients scanning proposed for the Hadamard transform, the "Hall Monitor" QCIF video sequence in YUV 4:2:0 format is coded with the 8x8x8 transform. Fig. 2.5(a) shows the sequence corresponding to the AC coefficients of the cube in position 7x8 belonging to the #264-271 luminance frames block read by a simple column-row-frame scanning order. This scanning order generates high energy periodic peaks at each 64 coefficients, approximately, and low energy periodic peaks spaced every 8 coefficients.

The first frame of the #264-271 frames block is shown in Fig. 2.6, with the cube in position 7x8 used in Fig. 2.5 highlighted. It can be observed that the highlighted cube has an approximately



(a)



(b)

Figure 2.5: AC coefficients of the "Hall Monitor" cube in spatial position 7x8 belonging to the range [264-271] of luminance frames read by: a) column-row-frame scan order and b) FEVC sequency scan order.

uniform content. Thus, the AC coefficients values tend to be low, which agree with the low energy values shown in Fig. 2.5.

Each higher energy peak in Fig. 2.5(a) corresponds exactly to the coefficient in the row of sequency number 0 and in the column of sequency number 0 of each frame in the coefficient cube. The eight coefficients in these positions are in the transversal axis in Fig. 2.4. The periodicity

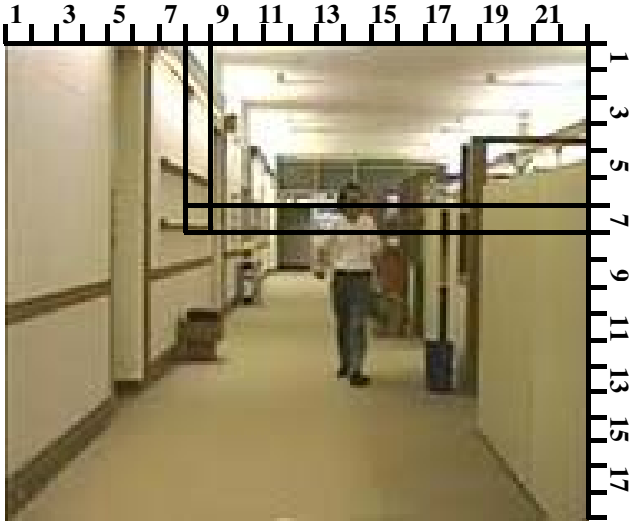


Figure 2.6: Highlighted cube 7x8 belonging to the #264 frame of the "Hall Monitor" sequence, whose AC coefficients of the luminance plane are shown in Fig. 2.5.

observed in the scanning of these coefficients is 64, as all other 63 coefficients from the previous frames are being read first.

For the lower energy peaks in Fig. 2.5(a), the periodicity is 8 because the reading order is row by row. Moreover, the energy of the peaks in the column of sequency number 0 is higher than the energy of the coefficients in other columns.

Based on the analysis of the graph in Fig. 2.5(a), it is possible to identify the pattern of energy spreading for the Hadamard coefficients cube. This motivates the adoption of the sequency-ordered scanning previously described. Fig. 2.5(b) shows the same coefficients as in Fig. 2.5(a), but read according to the proposed scanning order. As can be seen, a better grouping of the AC coefficients with similar values is in fact achieved.

Once the cube scanning order is established, it is necessary to determine a scanning order to read all cubes of the block. To exploit the correlation between coefficients located in the same position of adjacent cubes, the coefficients of all cubes are read according to a *spiral curve*, beginning with the coefficient of the left upper cube and finishing with the coefficient of the central cube. Initially, the DC coefficient of the left upper cube is read, and then the DC coefficients of its right neighbor cube is read, and so on until the DC coefficient of the central cube. Following, each AC coefficient (in the scanning order described before) is read similarly for all cubes.

2.3.2 *H.264/AVC Integer 3-D DCT-like Transform*

The 3-D DCT has been widely studied as an alternative approach to reduce video codecs complexity. The direct 3-D DCT for an image cube f with M x N pixels and L frames is defined as

$$F(\upsilon,\nu,\omega) = C \times \sum_{x=0}^{L-1}\sum_{y=0}^{N-1}\sum_{z=0}^{M-1} f(x,y,z) \times \frac{cos(2x+1)\upsilon\pi}{2L} \times \frac{cos(2y+1)\nu\pi}{2N} \times \frac{cos(2z+1)\omega\pi}{2M} \quad (2.7)$$

where $C = C(\upsilon,L) \times C(\nu,N) \times C(\omega,M)$ and $C(a,b) = \begin{cases} \sqrt{1/b} & \text{if } a = 0 \\ \sqrt{2/b} & \text{otherwise} \end{cases}$.

The main drawback of a 3-D DCT based codec is the use of a fixed-length transform regardless of the level of motion activity. To solve this problem, variable length 3-D DCT schemes with multiple thresholds have been proposed [11–14]. These techniques apply variable transforms based on certain empirical thresholds. Several other proposed models describe hybrid algorithms combining 3-D DCT-DWT as in [15], in which the Discrete Haar Transform (DHT) is used to reduce the temporal redundancy. These improvements simplify the 3-D DCT, but the transform results, being irrational numbers, are not suitable for integer arithmetic implementations.

In order to develop an integer DCT-like, the approach suggested in [16] is to round the scaled values of the DCT matrix to the nearest integers

$$H = round\{\alpha \times H_{DCT}\} \quad (2.8)$$

where H_{DCT} is the DCT matrix and $\alpha = 2.5$, which leads to

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}. \quad (2.9)$$

The H_2 matrix in Eq. (2.9) is the H.264/AVC integer DCT-like transform. For the inverse transform, in order to minimize the combined rounding errors from the reconstruction, the dynamic range gain must be reduced. The problem is in the odd-symmetric basis functions, whose

peak value is 2. Thus, the scaling of the odd-symmetric basis functions by 1/2 is proposed in [16] and the inverse H.264/AVC integer DCT-like transform matrix is defined by

$$\tilde{H}_2^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & 1 & -1 \\ 1 & -1/2 & 1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \tag{2.10}$$

where $\tilde{H}_2^{-1}$ is a scaled inverse of H_2, i.e.,

$$\tilde{H}_2^{-1} \times \begin{bmatrix} 1/4 & 0 & 0 & 0 \\ 0 & 1/5 & 0 & 0 \\ 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 1/5 \end{bmatrix} \times H_2 = I. \tag{2.11}$$

The multiplications by 1/2 are implemented by 1-bit right shifts. The small errors caused by the right shifts are compensated by the 2-bit gain in the dynamic range of the input to the inverse transform.

It is not easy to extend the 4x4x4 H.264/AVC integer-DCT to larger transforms, such as 8x8x8. The 8x8x8 transform cannot be directly obtained from the 4x4x4 transform as can be done for the Hadamard matrix by Eq. (2.2). Besides, the matrix values range is substantially increased. For instance, in the H.264/AVC integer-DCT 8x8x8 transform, the matrix values ranging from -12 to 12.

2.3.2.1 Dynamic Range Gain

For the integer DCT, the maximum sum of absolute values in any row of H_2 in Eq. (2.9) equals 6, so the increase of the maximum dynamic range gain for the 3-D transform is $log_2(6^3) = 7.76$, requiring 8 additional bits to store the transform coefficients.

Because of these additional 8 bits in the coefficient values, the FEVC performance with the integer DCT was not as satisfactory when compared to the performance with the Hadamard transform. The reason was that as much as 16 bit planes could have to be entropy encoded[3]

[3]The entropy coding performed in the FEVC is explained at Section 2.4.

with the integer DCT, while only a maximum of 11 bit planes have to be entropy encoded in the Hadamard transform case.

In order to control the 3-D integer DCT dynamic range gain, a new calculation method is proposed here, in which a scaling factor of 1/4 was extracted from the inverse transform shown in Eq. (2.11). Then, since the inverse transform is applied three times, two of the scaling factors are grouped and applied at the end of the coding process as 4-bit right shifts. When all three scaling factors were applied together at the end or during the coding/decoding process, the coefficient values became too small and lost precision. With this new computation method, enough coefficient precision is preserved and the number of additional bits needed to store the coefficient values is reduced by 4, resulting in a maximum of 12 bit planes being entropy encoded. With this reduction from 16 to 12 bit planes to be coded, the overall performance of the H.264/AVC integer 3-D DCT-like can finally be compared with the performance of the 3-D Hadamard.

2.3.2.2 Transform Coding Gain

From a compression standpoint, for a stationary Gauss-Markov input with correlation coefficient $\rho = 0.9$, the one-dimensional TCG of the integer DCT-like H_2 in Eq. (2.9), computed according to Eq. (2.6), is 5.38 dB [16].

For a comparative analysis, the one-dimensional TCG of the 4x4 DCT is 5.39 dB and the one-dimensional TCG of the 4x4 Karhunen-Loéve transform is 5.41 dB. As expected, the integer DCT-like TCG approaches the DCT TCG which, by the way, approaches the maximal Karhunen-Loéve compaction when ρ gets large.

The Hadamard transform achieves substantial less compaction than the integer-DCT like, as shown by its TCG computed in Section 2.3.1.2. However, as the transforms are applied here in three-dimensional fashion and, in practice, the empirical correlation coefficients tend to be in the neighborhood of 0.9, the Hadamard and the integer DCT performance can be comparable as can be observed by the final performance results presented in Section 2.5.

2.3.2.3 Energy Concentration and Coefficients Scanning

The more energy compacted in a fraction of total coefficients, the better energy compaction the transform achieves. One measure of this energy compaction is the one-dimensional TCG previously described. Besides providing high energy compaction in few coefficients, it is also desirable

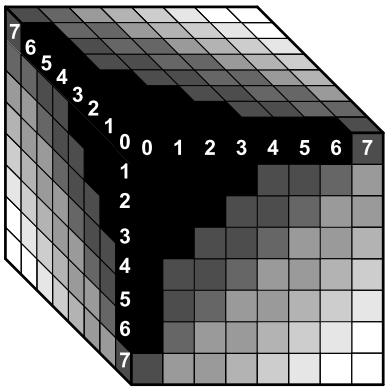


Figure 2.7: Coefficients energy distribution in the DCT cube.

that the transform provides a stable energy distribution pattern. Therefore, the high energy coefficients will always be firstly read according to a fixed scanning order and the coefficients will be arranged in a "decreasing in the average" order.

Conversely to the spreading energy pattern shown by the 3-D Hadamard Transform in Section 2.3.1.3, the probability distribution of the dominant DCT coefficients is clearly concentrated along the major axes of the cube [17, 18]. This DCT advantage is illustrated in Fig. 2.7, where the gray level is proportional do the coefficient energy.

The scanning order developed for the 3-D H.264/AVC DCT-like transform is the same scanning order based on the product of the three sequency numbers of each coefficient that was developed for the 3-D Hadamard transform and explained in the Section 2.3.1.3. The correlation between coefficients of adjacent cubes located in the same position is also exploited through a *spiral curve* scanning of coefficients. The difference is that for the DCT transform the sequency numbers correspond in fact to the cube coordinates. Therefore, the sequency-based scanning order reads firstly and appropriately the coefficients along the main cube axes.

2.3.3 ***3-D Piecewise-Linear Haar (PLHaar) Transform***

The PLHaar transform [19] is a reversible *n-bit* to *n-bit* transform, which results in no dynamic range expansion, based on the Haar wavelet transform. It is an integer and continuous transform suitable for lossy and lossless image compression. In fact, images transformed by PLHaar and reconstructed lossily have increased contrast, which helps preserve lines and edges in the image.

By keeping the coefficients to n bits, the PLHaar transform is particularly suited for use in hardware environment with limited bit arithmetic implementations. The transform guarantees

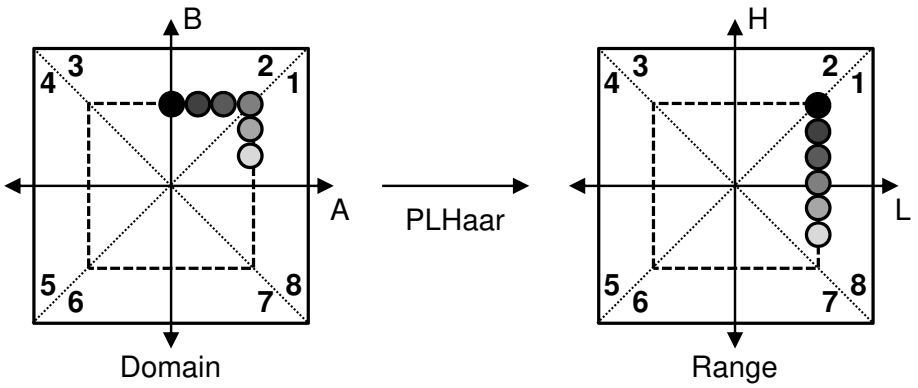


Figure 2.8: PLHaar rotation in L_∞ space.

no overflow or underflow of pixel values, i.e., for a 3-D point of a triple of pixel values in the range of $[0, 255] \times [0, 255] \times [0, 255]$, PLHaar transforms the point into a new point in the same range of $[0, 255] \times [0, 255] \times [0, 255]$. The PLHaar transform maps integers to integers and is an autohomeomorphism (meaning it maps the domain into itself).

While the Haar transform is defined as a 45-degree rotation in Euclidean L_2 space, the PLHaar is a similar rotation in L_∞ space. In this space, points that are equidistant from the origin lie on the perimeter of a square. A one-eighth rotation (analogous to the 45-degree rotation of the Haar transform) about the origin in this space moves a point one-eighth of the distance along the perimeter of its square. If the domain and range are divided into octants, as shown in Fig. 2.8, an one-eighth rotation moves all points from their positions in a given octant into the next lower octant (with wraparound). The transform as a whole is nonlinear, but when taken on a piecewise (octant-by-octant) basis, the transform is linear. It is from this property that the name "Piecewise-Linear Haar" was derived.

The PLHaar transform of a point at coordinates *(A, B)* is given by

$$\begin{bmatrix} L \\ H \end{bmatrix} = R \begin{bmatrix} A \\ B \end{bmatrix} \tag{2.12}$$

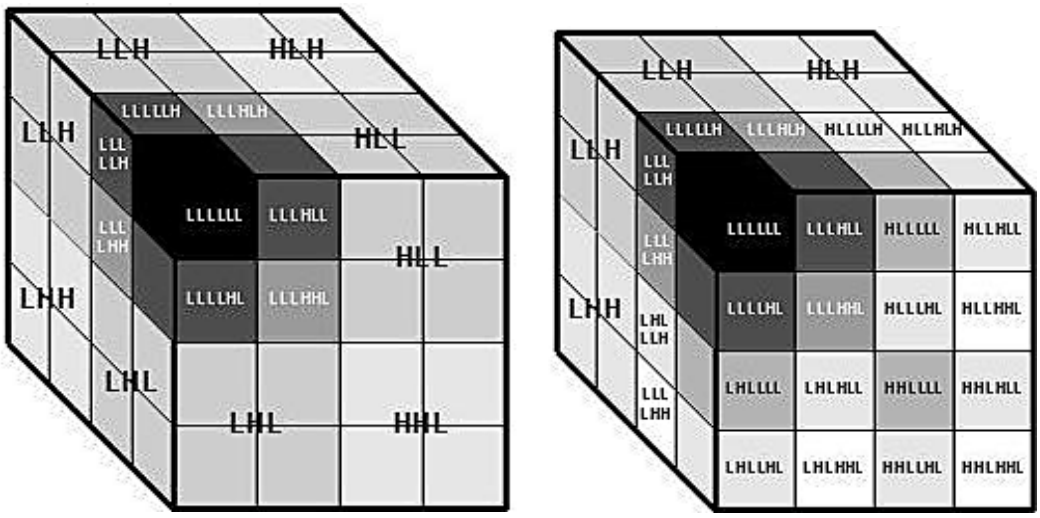
where R depends on the octant where the point *(A, B)* is located and is defined as

$$R = \begin{cases} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} & \text{if} \quad 0 \leq +B \leq +A \quad \textbf{\textit{Oct.}\,1} \quad \textit{or} \quad 0 \leq -B \leq -A \quad \textbf{\textit{Oct.}\,5} \\ \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} & \text{if} \quad 0 \leq +A \leq +B \quad \textbf{\textit{Oct.}\,2} \quad \textit{or} \quad 0 \leq -A \leq -B \quad \textbf{\textit{Oct.}\,6} \\ \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} & \text{if} \quad 0 \leq -A \leq +B \quad \textbf{\textit{Oct.}\,3} \quad \textit{or} \quad 0 \leq +A \leq -B \quad \textbf{\textit{Oct.}\,7} \\ \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix} & \text{if} \quad 0 \leq +B \leq -A \quad \textbf{\textit{Oct.}\,4} \quad \textit{or} \quad 0 \leq -B \leq +A \quad \textbf{\textit{Oct.}\,8} \end{cases} . \tag{2.13}$$

Eqs. 2.12 and 2.13 describe both the direct and the inverse transforms which are easily extended to larger transforms once PLHaar is a wavelet-based transform. However, in this work, the transform is evaluated in a block-based fashion, which means it will be applied only in 4x4x4 and 8x8x8 cubes. When the 4x4x4 PLHaar is applied as a 3-D transform (first to columns, then to rows and finally to frames), the 4x4x4 resulting coefficients are arranged according to the subbands shown in the Fig. 2.9(a). One can note in this figure that the first-stage 4x4x4 PLHaar was applied in the whole cube and then the second-stage 2x2x2 PLHaar was applied only to the first-stage subband cube LLL. This is the conventional result obtained with the application of the two-stage 4x4x4 PLHaar transform, but in this work, better results were achieved when this second-stage 2x2x2 PLHaar was also applied to the other low energy subbands: HLL, LHL and HHL. The final 4x4x4 subbands cube obtained in this work for the PLHaar transform is shown in Fig. 2.9(b).

2.3.3.1 Dynamic Range Gain

For the PLHaar transform, as previously described, there is no dynamic range expansion. So, if the domain is $[0, 255]$, the range of the transform is the set of all possible output coefficient pairs in this same domain. The result is that as much as 8 bit planes must be entropy encoded irrespective of the transform cube size.



(a) (b)

Figure 2.9: Two-stage PLHaar subbands cubes: *a*) conventional and *b*) proposed.

2.3.3.2 Transform Coding Gain

For the PLHaar transform, Eq. (2.6) is the ratio of the arithmetic mean of the variances σ_i^2 in each subband with respect to the geometric mean. So, the approximate one-dimensional TCG of the two-stage 4x4 PLHaar is 4.758 dB.

The PLHaar provides limited coding gain due to the property of not having dynamic range expansion and to the small transform size (4x4). Better coding gains are expected for the PLHaar with longer kernels or even enclosing the whole frame, but in this work, the PLHaar transform is applied in a small-block-based way.

2.3.3.3 Energy Concentration and Coefficients Scanning

Fig. 2.10 shows the energy distribution after the three-scale PLHaar decomposition, where the gray level is proportional to the coefficient energy. As previously explained in Sections 2.3.1.3 and 2.3.2.3, it is desirable that the cube coefficients be read in a "decreasing in the average" order.

Each coefficient in the subbands of the PLHaar cube represents a bandwidth and the energy distribution pattern is clearly related to low-high frequencies in Fig. 2.10. The scanning order developed in this work for the PLHaar cube takes into account the fact that the second-stage and/or the third-stage transforms were applied to all low energy subbands as shown in Fig. 2.9(b), not only to the lowest energy subband. Therefore, the scanning procedure begins with the lowest energy coefficients of all 2 or 4 front frames (respectively for 4x4x4 or 8x8x8 transforms), continues

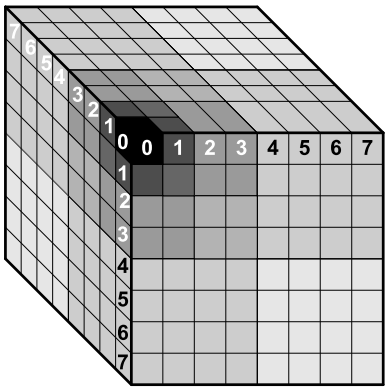


Figure 2.10: Coefficient energy distribution in the PLHaar cube.

with the scanning of the remaining coefficients in these front subbands and finishes with the scanning of the high energy back subbands.

2.4 Entropy Coding

Most video codecs perform quantization of the coefficient values before the entropy coding stage. The FEVC does not perform this explicit quantization step and so, it can also be used in a lossless manner. In fact, the FEVC performs an implicit coefficient quantization because the encoding is applied to bit planes (beginning with the most significant bit plane), which generates an embedded encoded bitstream. Thus, the decoding can be done aiming at a specific desired rate. Another possibility is to control the bit-rate during the encoding process generating the coded bitstream at the desired rate.

The entropy coding is performed in FEVC by adaptive versions of Golomb's run length encoding. It is important to mention that this coding is only applied to the most significant bit plane of each coefficient. The other bits (after the first most significant bit *1*) of a coefficient presents an approximately uniform distribution and, thus, they may be transmitted uncoded.

As video presents a highly non-stationary statistics, it is necessary to provide an adaptive method that adapts the encoder to the variant probability distribution of the bits in the bit planes. Two adaptive Golomb entropy encoders, versions [20] and [21], were implemented and tested.

A detailed explanation about Golomb's run length encoding, coding by bit planes and progressive video coding was included in [2].

The first Golomb entropy encoder [20] uses concepts extracted from well-known wavelet transform methods, such as wavelet scalable video compression [22], embedded zerotree wavelet (EZW) [23], and set partitioning in hierarchical trees (SPIHT) [24,25], and has a single operation mode. Considering that M is the Golomb encoder length parameter, the incomplete zero runs (runs of length equal to M) are mapped to the one-bit codeword (*0*) and the complete zero runs (runs of length $0 \leq x \leq M-1$) are mapped to codewords of length $\lfloor log_2 M \rfloor$, when

$$x < 2^{\lfloor log_2 M \rfloor + 1} - M \tag{2.14}$$

and to codewords of length $\lfloor log_2 M \rfloor + 1$, otherwise.

The adaptation strategy presented in [20] adjusts the encoder length in two situations: in the middle of a run of zeros and after the end of the run of zeros. The adaptation procedure is described in both cases as:

- When a *0* that represents a run of M zeros is sent/received by the encoder/decoder, the length parameter M is incremented by $\lfloor (M+1)/2 \rfloor$.
- When the codeword representing a complete zero run is sent/received by the encoder/decoder, a new estimate of the average run of zeros $\overline{X}$ is calculated and the encoder/decoder length is set to $\left\lfloor \left(\overline{X}+1\right)/2 \right\rfloor$. The new estimate of the average run of zeros is calculated by a first-order auto-regressive model described by

$$\overline{X}_{n+1} = \alpha \overline{X}_n + (1-\alpha) X_n \tag{2.15}$$

where $\alpha \in [0,1]$ is a memory factor.

The second Golomb entropy encoder [21] has four operation modes: a mode where no coding is performed, a mode with Huffman code, and two modes with Golomb run length codes, each with a different subset of parameters. The mode is selected by the current value of the estimated average run of zeros. The four modes are described as:

- *First mode:* the output symbol is equal to the input symbol and no coding is carried out.
- *Second mode:* the output is a Huffman code for the vector formed by one symbol plus a run of up to two zeros. It is not a pure run length code.

- *Third mode:* it is a Golomb code, as described in [20], but with $M = 2^k$. These codes are called Golomb-Rice codes.
- *Fourth mode:* it is a Golomb code with $M = 3 \times 2^{k-1}$, named *half-mode code*, where the complete zero runs of length $0 \leq x \leq 2^{k-1}$ are mapped to codewords of *10* plus the binary representation of x. Otherwise, the runs are mapped to *11* plus the binary representation of $x - 2^{k-1}$.

The Golomb entropy encoder in [21] has a complexity approximately equal to that of popular adaptive Golomb-Rice coder. However, whereas Golomb-Rice coder has an excess rate with respect to the source entropy of up to 4.2% for binary sources with unknown statistics, the encoder in [21] has an excess rate of less than 2%. The adaptation is done after the production of each output codeword. The adaptation strategy is based on the last N previously encoded strings (backward adaptation) and is done by the simple rule

$$N \times \overline{X} \equiv \frac{N - n_1}{N}(N \times \overline{X} + n_0) \tag{2.16}$$

where n_0 and n_1 are, respectively, the number of *0* symbols and the number of *1* symbols read from the input sequence to produce the output codeword. Given the updated $\overline{X}$ (or the update $N \times \overline{X}$ to keep an integer variable), a short table of transition points is created to determine the parameter k and the operation mode to be used for the next input symbol. This adaptation rule is nearly maximum likelihood and can be efficiently implemented in practice, using only additions and binary shifts.

The two adaptive Golomb entropy coders presented here produced very similar results when well adjusted and applied to the FEVC. The first entropy encoder has a fast adaptation strategy that is generally well adjusted to non-stationary data, just as the bit plane values of the video data. It can be somewhat tuned to the data by the adjustment of the memory factor α, which was adjusted to 0.7 in the FEVC. The second entropy encoder was originally designed to adapt to independent, identically Bernoulli distributed data with slowly varying statistics, but by making the backward adaptation buffer size as small as $N = 2$, the same satisfactory performance of the first entropy encoder was achieved. This performance for a small buffer size shows that the precision in the statistical analysis of the data is less significant for the FEVC than the speed of adaptation.

Although the performance of both entropy coders is quite similar, the complexity is substantially different. Due to the simpler adaptation procedure of the first entropy encoder, its execution is much faster than the second one. Therefore, only the first adaptive entropy encoder described here [20] was chosen to be implemented in the final FEVC version.

2.5 Results

In order to achieve a multi-platform code, the codec computational system is implemented in C# language, in the Microsoft Visual C# .NET environment. The encoding and decoding processes, as well as all other supported operations (e.g., parameters settings and optimized coded video file bit-rate reduction), are controlled by the user through graphical interfaces.

With the FEVC, it is possible to perform the video sequence encoding and decoding operations separately or in sequence. Fig. 2.11 presents the screen for the latter option, where the video sequence decoding begins immediately after the end of the coding process. Through this interface, the user can inform the original video file path, format, resolution and quantity of frames. The desired coded file bit-rate can also be chosen. Bit rates lower than the ones available in the interface are also obtained through an additional graphical interface of the FEVC. This interface enables a fast reduction of the coded video file bit-rate to the bit-rate desired by the user. All encoding and decoding operations can be followed by the user through the messages shown in the "Coding and Decoding Status" area. The messages shown in Fig. 2.11 correspond to the end of a successful video sequence coding and decoding processes.

In order to evaluate the FEVC computational efficiency, the encoding and decoding times of given video sequences were measured. All execution times were obtained with a Pentium-4 3.20 GHz processor, running exclusively the codec application. Although the CPU clock cycles were also used by the operating system during the codec execution, the measurement of the execution times is considered because the codec should be executed in a real-time software environment. These measurements are more significant for comparisons than the sheer number of operations (shifts, additions, subtractions, comparisons, multiplications, logical operations) because they include the influences of all relevant factors, including the available hardware, CPU clock wastes during reading and writing operations, and other ancillary tasks.

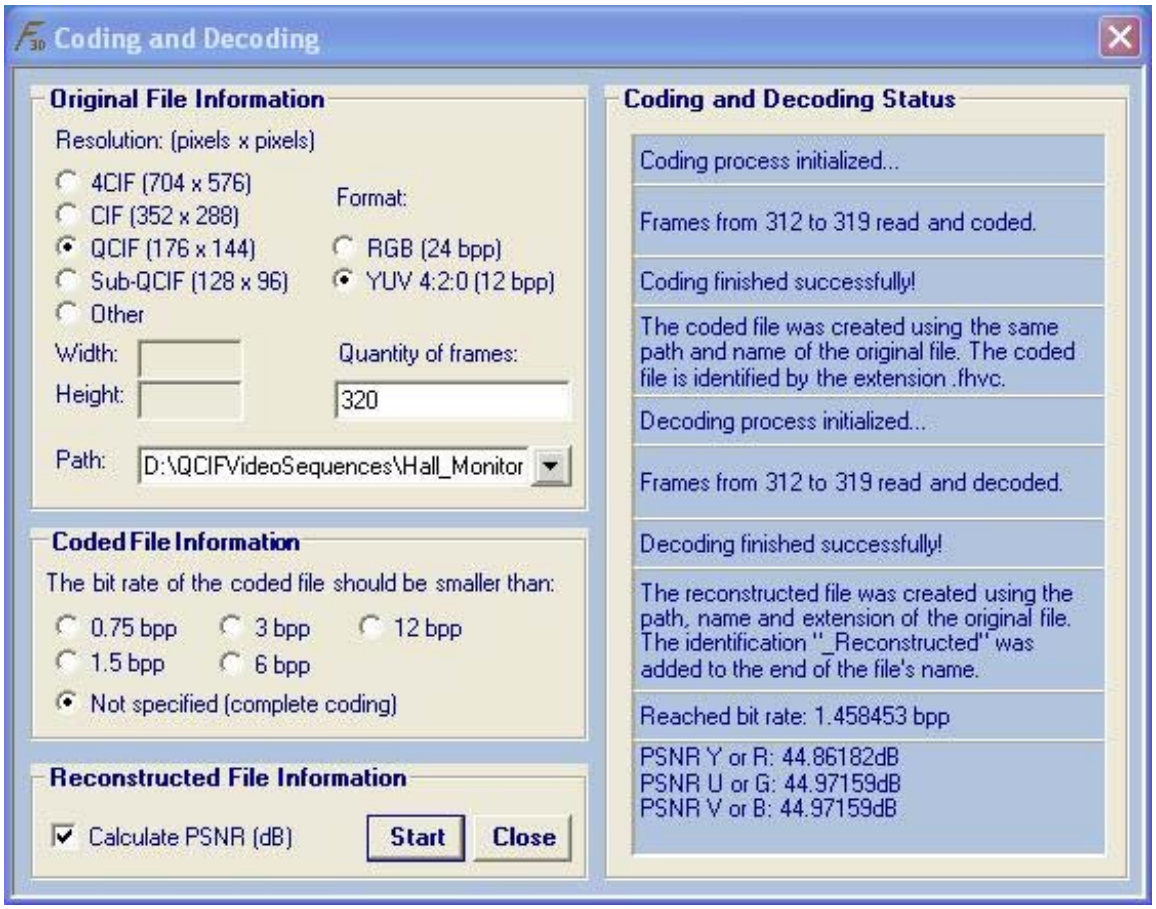


Figure 2.11: FEVC graphical interface showing the result of a video sequence coding and decoding processes.

An analysis of FEVC was done for given video sequences in order to identify the most time-consuming encoding steps. It was observed that the 3-D transform calculation occupies approximately 56% of the encoding time, followed by the Golomb entropy encoder which occupies 28%, then by the scanning process with 7% of the time and, finally, by the video files reading and writing operations with 9% of the encoding time. The use of a fast transform implementation that is optimized for the specific native machine should substantially speed up the encoding process.

For performance comparisons, the H.264/AVC official reference software was obtained in [26]. The techniques used in this standard codec are very different from the techniques used on the FEVC. Nevertheless, the comparison with the H.264/AVC is considered interesting because it is currently the industry standard and the video codec with the best overall performance.

It is very important to emphasize that there are H.264/AVC optimized implementations that run much faster than the official reference software. Even so, the FEVC performance is being compared with the official reference software performance because this application has a publicly available implementation and serves as a well established reference for the comparisons. Naturally, coding and decoding times of any other codec can also be compared with the H.264/AVC official reference software and then, be indirectly compared to the performance obtained with the FEVC.

An additional caveat is that the FEVC implementation is not optimized for the hardware where it is being executed. Also, the programming language used in FEVC (C#) is an interpreted language and a compiled code version was not generated. Therefore, the provided comparisons must be regarded as a basis for qualitative conclusions with respect to the different encoders, not as a definite measuring stick to provide peremptory assessments.

Most H.264/AVC configuration parameters were set as "default", according to the software official manual developed by the Joint Video Team (JVT). The parameters not set as "default" in the configuration file are:

- *Main profile:* level 2.0, GOP of size 15 given by I-B-B-P-B-B-P-B-B-P-B-B-P-B-B, 5 reference frames and CABAC (Context-based Adaptive Binary Arithmetic Coding) entropy coding.
- *Baseline profile:* level 2.0, GOP of size 15 given by I-P-P-P-P-I-P-P-P-P-I-P-P-P-P, 5 reference frames and CAVLC (Context-Adaptive Variable Length Coding) entropy coding.

Y-PSNR versus bit-rate curves and visual quality comparisons are presented here for three QCIF sequences in the YUV 4:2:0 format, with different motion and background characteristics. Approximately, 300 frames of each sequence were used.

Fig. 2.12 presents the results for the "Akyio" sequence. The visual quality comparison is presented in Fig. 2.13. One can notice that the performances of the 8x8x8 Hadamard, 4x4x4 Hadamard and 4x4x4 Integer DCT are quite comparable. The 8x8x8 Hadamard is slightly better, which is expected due to the high temporal correlation of this sequence. Other sequences with higher motion contents and more detailed frames would typically be better coded with only 4 frames per cube. The 4x4x4 PLHaar and the 8x8x8 PLHaar transforms are also quite comparable and the performances are significantly inferior. The PLHaar uses approximately 3 times the bit-rate of the Hadamard/Integer DCT for the luminance signal. This inferior performance can be

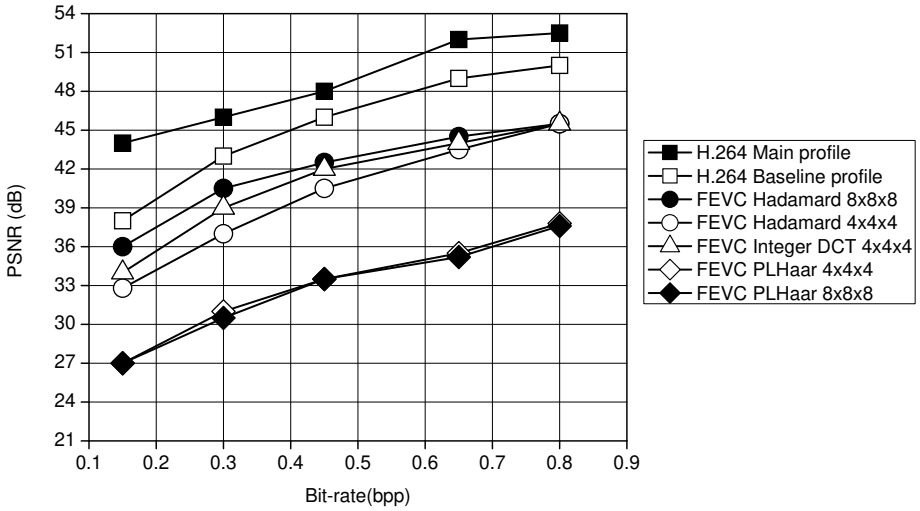


Figure 2.12: Y-PSNR versus bit-rate curves for the "Akyio" sequence.

Figure 2.13: "Akyio" frame #160 encoded by: *a*) H.264 at 0.16 bit/pixel, *b*) FEVC Hadamard 8x8x8 at 0.16 bit/pixel, *c*) FEVC Hadamard 8x8x8 at 0.30 bit/pixel, *d*) FEVC Integer DCT 4x4x4 at 0.30 bit/pixel, *e*) FEVC PLHaar 4x4x4 at 0.65 bit/pixel and *f*) FEVC PLHaar 8x8x8 at 0.45 bit/pixel.

explained due to the property of not having dynamic range expansion and to the application in a small-block-based way.

It is also shown in Fig. 2.12 that the H.264/AVC codec always achieves superior results in terms of Y-PSNR versus bit-rate. According to Fig. 2.12, the FEVC uses approximately 3 times

the bit-rate of H.264 Main profile and 1.5 times the bit-rate of H.264 Baseline profile. This rate-distortion result is justified in applications where high capacity is available (e.g. optical links) but computational resources (complexity) at the end nodes are limited. This is the case for FEVC, as it was designed to be executed by uploadable software in low cost set-top boxes on a high speed fiber optical network environment.

As expected by the Y-PSNR versus bit-rates curves, the H.264 Main profile presents good results when coding at 0.16 bit/pixel as shown in Fig. 2.13a and the FEVC 8x8x8 Hadamard produces more visible block effects as shown in Fig. 2.13b. In Fig. 2.13c, when the same frame #160 is coded at 0.30 bit/pixel, the FEVC achieves a reasonable value of 40 dB for the luminance component and the result can be considered comparable with Fig. 2.13a. When the Integer DCT is applied at 0.30 bit/pixel, the result shown in Fig. 2.13d is similar with Fig. 2.13c, except by the inferior coding of the chrominance signal. The visual results obtained with PLHaar are shown in Fig. 2.13e (cube 4x4x4) and in Fig. 2.13f (cube 8x8x8). It can be seen that the luminance signal achieves values around 35 dB, but the overall performance is better than Fig. 2.13b (with 36 dB for the luminance signal). The reduction in the block effects is a feature of the PLHaar transform, as well as the increased contrast, which helps preserve lines and edges in the image. However, the coding of the chrominance signal in Figs. 2.13e and 2.13f is significantly inferior.

The encoding and decoding times obtained with the FEVC and the H.264/AVC codec (measured at the same bit-rates) for the "Akyio" sequence are shown, respectively, in Tables 2.1 and 2.2. As the FEVC is a symmetric codec, the encoding and decoding times are almost equal, unlike H.264/AVC, where decoding is 23 times faster, in average, than encoding with the Main profile and 17 times faster than encoding with the Baseline profile.

Table 2.1 shows that encoding with H.264 Baseline profile is about 1.5 times faster, in average, than encoding with Main profile. One can see, from the bit-rate curves shown in Fig. 2.12, that the performance difference between the profiles reaches 6 dB in the low bit-rate scenario (around 0.15 bpp) and is reduced to only 2 dB in the high bit-rate scenario (around 0.8 bpp, a compression factor of about 15). These results were expected, since the H.264 Main profile has emphasis on coding efficiency, and the H.264 Baseline profile is more focused on reducing complexity, not supporting B slices and CABAC, among other features of the Main profile. So, the H.264 profile choice should depend on the available bit-rate and computational resources.

Bits per pixel	Time per frame (milliseconds)					
	FEVC Hadamard 8x8x8	FEVC Int DCT 4x4x4	FEVC PLHaar 4x4x4	FEVC PLHaar 8x8x8	H.264 Main	H.264 Baseline
0.81	17.17	13.82	19.33	18.21	3453.69	2331.54
0.65	16.28	12.91	18.89	18.34	3426.07	2325.21
0.45	15.41	12.22	17.98	17.86	3357.16	2286.11
0.31	16.67	11.31	17.26	17.02	3281.18	2237.71
0.16	13.94	10.71	17.84	16.98	3134.21	2211.93

Table 2.1: Encoding times for the "Akyio" sequence.

Bits per pixel	Time per frame (milliseconds)					
	FEVC Hadamard 8x8x8	FEVC Int DCT 4x4x4	FEVC PLHaar 4x4x4	FEVC PLHaar 8x8x8	H.264 Main	H.264 Baseline
0.81	15.67	15.62	16.78	16.51	130.62	131.35
0.65	14.78	14.48	16.35	16.09	130.31	130.52
0.45	13.91	13.95	15.28	15.11	128.99	129.35
0.31	13.17	13.21	14.59	14.58	128.79	127.72
0.16	12.44	12.45	13.95	13.33	126.97	126.51

Table 2.2: Decoding times for the "Akyio" sequence.

Based on Table 2.1, FEVC is, as expected, faster than the H.264/AVC official reference software, being approximately 200 times faster in encoding (for Integer DCT and Hadamard transforms) than the Main profile and 135 times faster than the Baseline profile. For the decoding, according to Table 2.2, the FEVC is approximately 10 times faster than both H.264 profiles. As the FEVC is faster than H.264/AVC, but consistently inferior in terms of Y-PSNR versus bit-rate, the necessary bit-rates for the FEVC to achieve a sequence with comparable visual quality to that achieved by H.264/AVC are searched. This study showed that for rates of 0.30 bit/pixel, the FEVC produces the frames shown in Fig. 2.13c, which have visual quality somewhat comparable with the H.264/AVC Main profile frame encoded at 0.16 bit/pixel, shown in Fig. 2.13a. The H.264 Main profile was chosen for this study because in the low bit-rate scenario, the H.264 Baseline profile is not so satisfactory.

According to Table 2.1, the H.264/AVC Main profile codec requires 3,134.21 ms per frame for encoding at 0.16 bit/pixel. The FEVC Hadamard requires 16.67 ms per frame for encoding at 0.31 bit/pixel, which produces similar visual quality frames for the selected sequence. Thus, at the cost of reducing the H.264/AVC Main profile compression rate by a factor of 1.875, an encoding approximately 180 times faster can be achieved with the FEVC. Another important observation is that the encoding time of 16.67 ms per frame makes it possible to have real-time video sequences encoding at 30 frames per second.

In order to validate the results obtained with the FEVC for the "Akyio" sequence, the Y-PSNR versus bit-rate curves of the "Hall Monitor" sequence are shown in Fig. 2.14. The Fig. 2.15 shows the visual results.

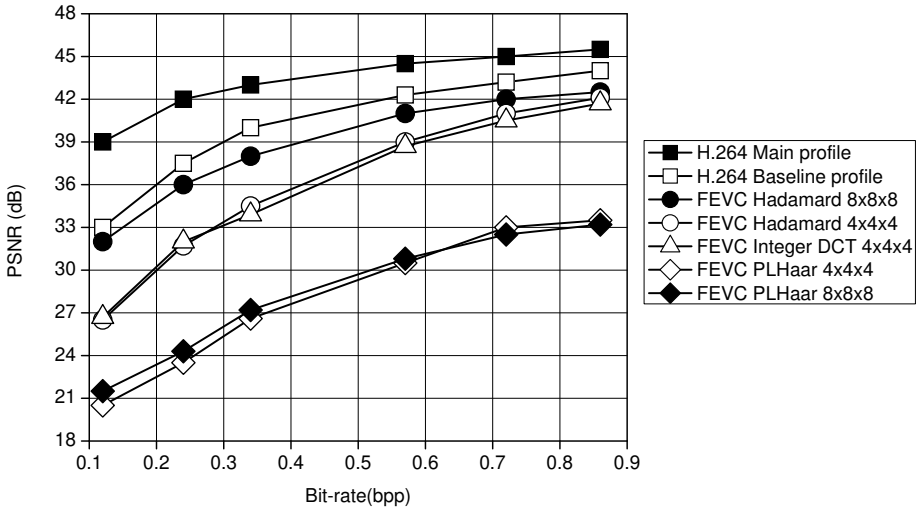


Figure 2.14: Y-PSNR versus bit-rate curves for the "Hall Monitor" sequence.

Figure 2.15: "Hall Monitor" frame #264 encoded by: *a*) H.264 at 0.12 bit/pixel, *b*) FEVC Hadamard 8x8x8 at 0.12 bit/pixel, *c*) FEVC Hadamard 8x8x8 at 0.33 bit/pixel and *d*) FEVC PLHaar 8x8x8 at 0.33 bit/pixel.

One can see in Fig. 2.14 that the FEVC reaches a satisfactory performance for the "Hall Monitor" sequence at rates of 0.12 bit/pixel (at compression factors of 100), yielding a Y-PSNR value of 32 dB for the luminance component. These results are as good as the ones obtained with the "Akyio" sequence when considering Y-PSNR degradation in respect to H.264.

Fig. 2.15 shows a sample of "Hall Monitor" sequence frames for a visual quality comparison. The frame produced by FEVC at 0.33 bit/pixel in Fig. 2.15c has visual quality that is comparable with the frame produced by H.264 Main profile at 0.12 bit/pixel, presented in Fig. 2.15a. Thus, at the cost of reducing the H.264/AVC Main profile compression rate by a factor of about 2.75, a significantly faster encoding can be achieved with the FEVC for the "Hall Monitor" sequence.

Results for the "Foreman" sequence are also shown in Figs. 2.16 and 2.17. For this sequence, which has accentuated motion content, the FEVC reaches satisfactory performance at medium bit-rates (around 0.5 bpp) and at high bit-rates, as shown in Fig. 2.16. Some "Foreman" frames are shown in Fig. 2.17. The frame produced by FEVC at 0.55 bit/pixel, shown in Fig. 2.17c, has visual quality comparable with the frame produced by H.264 Main profile at 0.25 bit/pixel, shown in Fig. 2.17a. The frames produced by PLHaar in Figs. 2.17e and 2.17f illustrate how the transform increases image contrasts.

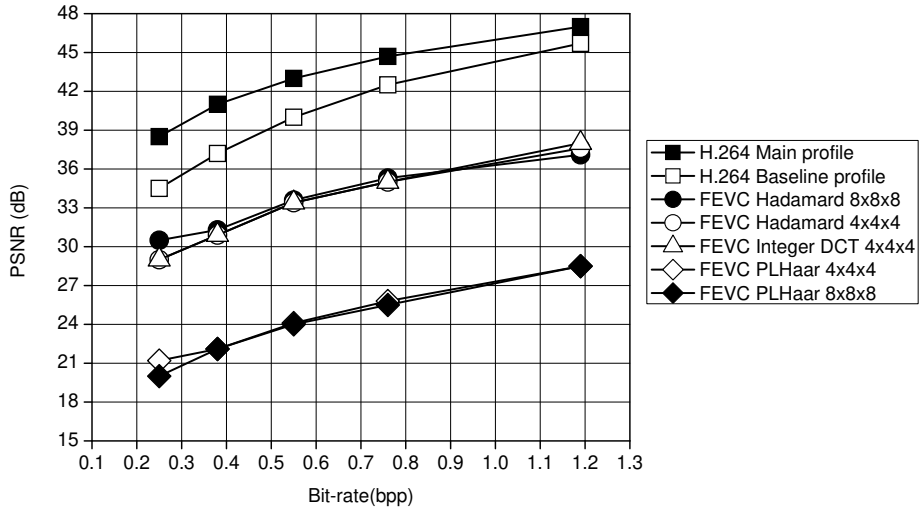


Figure 2.16: Y-PSNR versus bit-rate curves for the "Foreman" sequence.

Figure 2.17: "Foreman" frame #128 encoded by: *a*) H.264 at 0.25 bit/pixel, *b*) FEVC Hadamard 8x8x8 at 0.25 bit/pixel, *c*) FEVC Hadamard 8x8x8 at 0.55 bit/pixel, *d*) FEVC Integer DCT 4x4x4 at 0.55 bit/pixel, *e*) FEVC PLHaar 4x4x4 at 0.55 bit/pixel and *f*) FEVC PLHaar 8x8x8 at 1.2 bit/pixel.

2.6 Conclusions

This chapter presented a comparison of three simple block-based 3-D transforms and two entropy encoders applied to a fast embedded video codec (FEVC). New fast transform implementations and 3-D scanning orders for each transform were developed and implemented. Besides being fast, the new implementations adjusted the dynamic range of the coefficients to the available fixed point 16-bit representation.

The results obtained with the PLHaar transform were not impressive, possibly due to the small block-based implementation and the constrained dynamic range of the transform coefficients. The best results were obtained with the 8x8x8 Hadamard transform for slowly varying uniform sequences and with the 4x4x4 H.264/AVC integer DCT-like transform for more active detail-rich sequences. As the Integer DCT leads to the fastest implementation, due to the simplified scanning process described in this work, and yields similar compression performance to that of the Hadamard transform, the Integer DCT should be the chosen transform for the projected FEVC system.

The purposes of the FEVC are directed to video streaming and video conferencing applications, using systems with complexity and storage limitations, and possibly fixed point processors, but enjoying high bit-rate network connections (low cost codecs making use of high performance links).

For high bit-rate applications (around 0.9 bpp), the Y-PSNR degradation with respect to H.264 is less pronounced (around 3 dB for sequences with high spatial and temporal correlations) than for low bit-rate applications (around 0.1 bpp), where this degradation may be in excess of 6 dB. Performance results for three particular video sequences were shown. Results with other video sequences also led to similar conclusions and indicated that, at the cost of a reduction in H.264/AVC compression rate by a factor of 1.5 to 3, it is possible to get encoding times that are significantly smaller, as much as 160 times. For comparison purposes, in [2], the FEVC encoding process was also 160 times faster, but at the cost of a reduction in H.264/AVC compression rate by a factor of 2 to 5.

Chapter 3

Optimized Multiview Video Coding for Free Viewpoint Video

3.1 Introduction

Three-dimensional (3-D) video is currently a very active area of research. Following a slower initial phase, technology is now maturated, providing excellent quality [27]. In particular, in the entertainment world, 3-D movies and 3-D games (and stereo monitors) are now widely available, and TV manufacturers have introduced 3-D TV as the new consumer electronics revolution. Most of this revolution is due to new technologies in stereo display, which have introduced reasonable cost, high quality stereo display using shutter glasses and barrier-based autostereoscopic displays. Improved technology and reduced costs in capture, transmission, and rendering of computer graphics have all also contributed to the recent surge of 3-D.

In addition to the *stereo parallax* provided by most displays and applications, realistic 3-D video also requires *motion parallax*. Providing motion parallax for computer generated images is nearly trivial. Nevertheless, doing the same for natural images is a much more complex problem. When dealing with natural images, each step of the process is harder: capture, rendering, and transmission are all challenging. Significant research is still needed in many of these aspects before commercial systems can be widely deployed. In particular, to provide motion parallax to natural scenes, the most widely used technology is image-based rendering [28–31]. In this technique, multiple cameras capture the scene, each from a different viewpoint, which are in turn used to synthesize a new image from the desired viewpoint. The larger the number of views, the less artifacts can be expected. Typical systems vary from as low as four cameras to as high as 64 cameras. The high number of cameras implies a high requirement for bandwidth, during acquisition, transmission, storage, and every step of the processing chain.

Providing motion parallax means adapting the displayed image to the current position of the viewer. Virtual intermediate views can be rendered for any position in between the cameras, thus providing advanced free viewpoint video (FVV) functionality. This can be done, for example, by head tracking, as proposed in [32]. However, for realistic perception, the time lag between the head movement and the display of the new image has to be kept to a minimum. Otherwise, the depth perception is severely impacted, causing discomfort and possibly nausea or other side effects. To achieve such low delays, the rendering operation needs to be *local*. That is, the receiver/renderer needs instant access to all pixels required to synthesize the new view. If the images have to be transmitted over a network with a delay above just a few milliseconds, this implies encoding several video streams, and sending them to the viewer site for decoding, and generating the synthetic view locally. This leads to the need of Multiview Video Coding (MVC), which is the subject of the optimization work proposed in this chapter.

Multiview video coding encodes several video signals as one stream. Besides the spatial and temporal correlations in a single-view video sequence, MVC can also exploit the inter-view correlations between the adjacent views. There was significant effort in standardizing the technology a few years ago [33–36] and several works have been proposed since then to improve the standard [37–41]. The improvement being studied in this chapter was firstly proposed in [42]. This approach exploits the (estimated) viewer position and the video stream is encoded based on the likelihood that each view is going to actually be used to synthesize the final frame. With a simple approach, based on varying the macroblock quantization parameter (QP) according to estimated weights of each pixel in the synthesized image, the authors show gains on the order of 50% for typical cases when compared with the H.264/AVC codec.

The authors in [42] give a heuristic mapping between the pixel weights and the macroblock QP. More specifically, the mapping has two ad-hoc stages: a quadratic averaging between pixels to obtain a "mean MB weight", and a logarithmic mapping between this averaging and the QP for the macroblock. The analysis of this choice of mapping is performed in this chapter. Firstly, the optimal mapping between the pixels weights and QPs is derived mathematically and the proposed mapping in [42] is actually proved to be quasi-optimum for a Gaussian random variable. However, video signals are not Gaussian. Therefore, optimum results are derived for a typical signal based on the rate-distortion curve for that signal. Secondly, the optimum way to average the weights within a macroblock is studied.

This chapter initially gives an overview of the MVC field in Section 3.2. Then, Section 3.3 describes the optimization proposed here. Sections 3.3.1 and 3.3.2 briefly details the set up. In Section 3.3.3 the optimum mapping between weights and QPs is derived and in Section 3.3.4 the averaging step within the macroblock is discussed. Finally, Sections 3.4 and 3.5 present some experimental results and conclude the chapter. The experimental results have been partially published in [43].

3.2 Multiview Video Coding

Multiview color video can be separately compressed using H.264/AVC in a coding scheme referred as *simulcast coding*, or jointly compressed using the *MVC extension* of H.264/AVC [33], by exploiting the redundancy between views. Fig. 3.1 illustrates how the successful concept of hierarchical bidirectional (B) pictures [44] is used in simulcast coding for a sequence with five views and a group of pictures (GOP) length of 8. Fig 3.2 shows how the advantages of B pictures are combined with inter-view prediction in the MVC extension, without any changes regarding the temporal prediction structure. The first view (I-view) is initially encoded without inter-view prediction. Then, the third view (P-view) is encoded with unidirectional inter-view prediction from the reconstructed I-view as a forward reference. Finally, the second view (B-view) is encoded with bidirectional inter-view prediction from the first and third views as forward and backward references, respectively. The same procedure is repeated for the fourth and fifth views in Fig. 3.2.

Additionally to the multiview texture images, the corresponding depth maps must also be generated and coded in order to enable FVV. A depth image represents the distance between

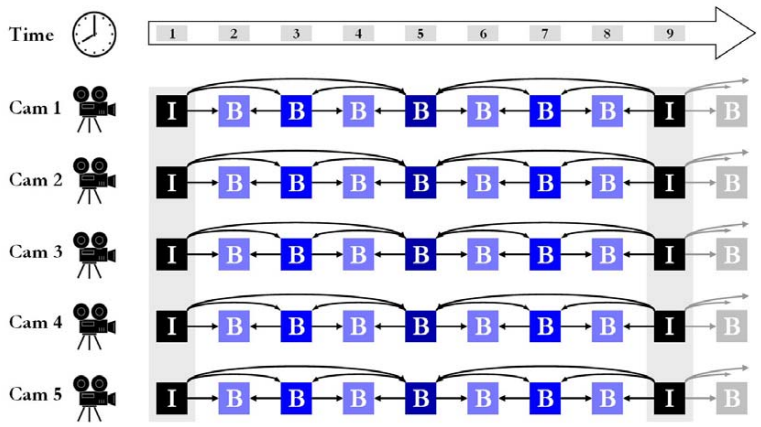


Figure 3.1: Simulcast coding structure with hierarchical B pictures for temporal prediction.

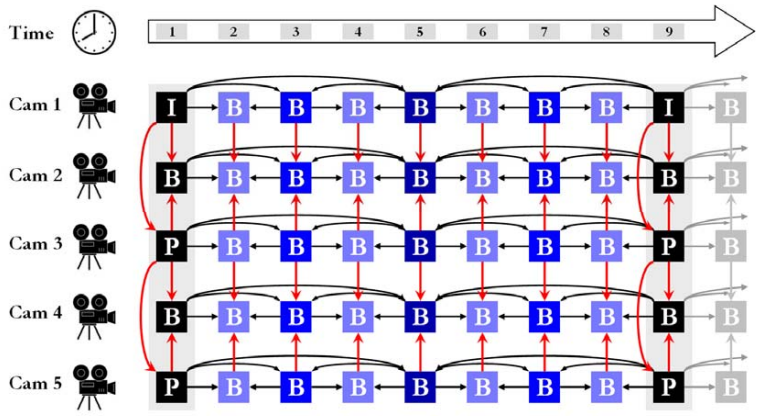


Figure 3.2: Multiview coding structure with hierarchical B pictures for both temporal and inter-view prediction.

an object and a camera as a gray scale image having pixel values between 0 and 255. It is composed mostly of flat areas with sharp edges at boundaries between objects and the background. Since it is similar to the luminance component of a color video sequence, a depth image can be compressed either by simulcast or MVC extension coding. The MPEG standard format that combines both texture and depth images is named multiview video plus depth (MVD) [34]. In the MVD representation, each texture image sequence is accompanied by the corresponding per-pixel depth map sequence.

Typically, the MVC extension has been used to compress the MVD representation (resulting into the multiview texture video and the multiview depth video in two separate MVC bitstreams), although the additional gains achieved with this coding scheme are usually in the order of only 0.5 dB [45] when compared with the simulcast coding (referred to as MVD coding with simulcast H.264/AVC).

In order to improve the MVD representation coding efficiency, several schemes have been proposed in the literature [38–41]. These schemes exploit the correlation between the texture video and the corresponding depth map and also reduce the number of bits allocated for the depth video by, for example, skipping some depth images. The depth images are only used for rendering intermediate views and the perceived quality of these rendered images is affected more by texture images than by depth images. Therefore, maintaining the fidelity of depth information is important for the rendering process, but it is crucial to strike a good balance between the fidelity of depth data and the associated compression requirement.

3.3 Multiview Video Coding based on Predicted Viewer Position

The coding scheme proposed in this chapter to efficiently encode the multiview video is focused on one particular scenario: parallax-based free viewpoint video coding for 3-D video conferencing applications or other similar unicast-like applications.

Fig. 3.3 shows an overview of the target system. It is important to note that the video transmission over the network has to be outside the synthesis loop. Otherwise, the full network round trip delay would be added to the lag between the head motion and the corresponding change in the synthetic view.

One of the main contributions of [42] is to point out that, since the user only sees the synthesized (or rendered) video, the performance measurement has to be done in that video, instead of directly in the decompressed videos. Fig. 3.4 illustrates the performance criteria, showing the PSNR comparison to be made between the final, synthesized video, and synthesized video that would be obtained using the uncompressed video frames. An indirect, and undesirable, consequence is that the performance of the compression algorithm is now attached to the algorithm used in synthesizing the video. However, these two problems are, indeed, tightly coupled, so it is only natural that tuning one of them to the other will improve final results.

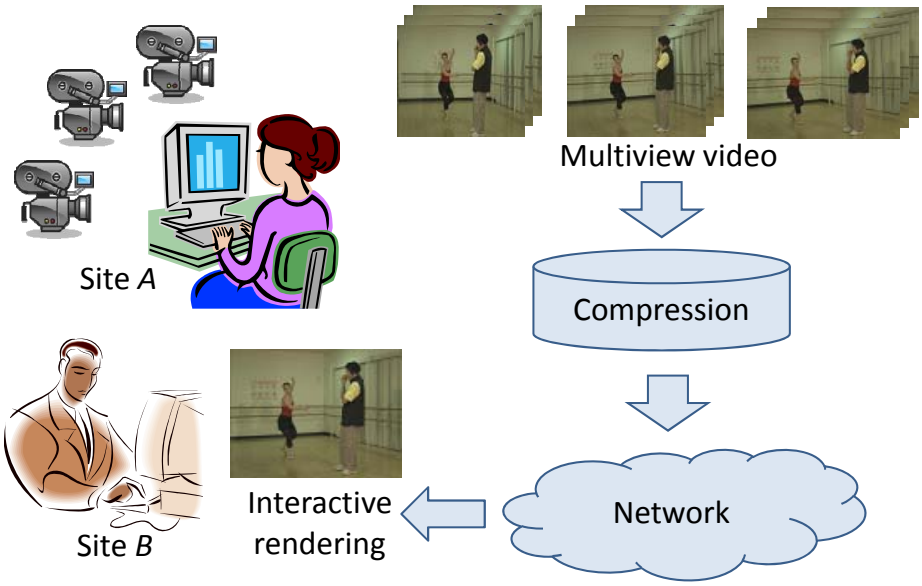


Figure 3.3: Immersive tele-conferencing scenario. The system could be symmetrical, i.e. site B could also sent its multiview video to site A.

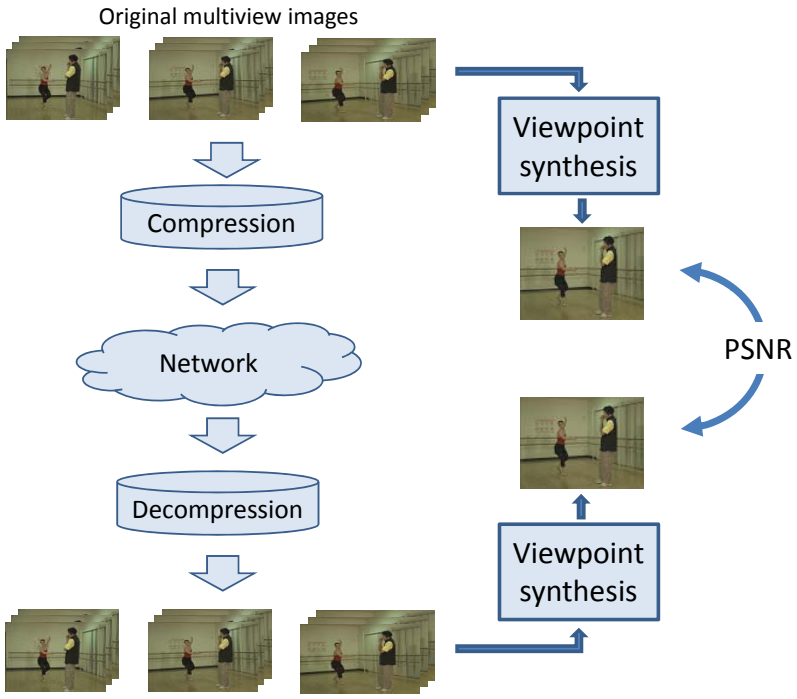


Figure 3.4: A more appropriate error criteria: instead of measuring PSNR directly on the decompressed frames, it is more appropriate to measure between the synthetic frames obtained using the decompressed images and the original ones.

3.3.1 ***Video Rendering***

The same view synthesis algorithm used in [42] is being applied here. It assumes that texture views are available from a number of cameras, as well as a single depth map to facilitate the virtual view rendering. Fig. 3.5 shows an example of a dataset for a particular frame of the *Breakdancers* multiview video sequence. As shown in Fig. 3.6, given a virtual viewpoint, firstly it is split into light rays. Then, each light ray is traced to the surface of the depth map. The intersection is obtained and reprojected into nearby cameras (Cam 3 and 4 as shown in Fig. 3.6). The intensity of the light ray is thus the weighted average of the projected light ray in Cam 3 and Cam 4. The weight can be determined by many factors [30]. In the simplest form, the angular difference between the light ray to be rendered and the light ray being projected can be used, assuming the capturing cameras are at roughly the same distance to the scene objects [36].

Fig. 3.7 shows one of the images rendered from a virtual view point nearby Cam 2 (slightly rotated view direction). The weight maps clearly demonstrate whether a pixel in a captured video frame will be useful for rendering this particular virtual view. In Fig. 3.7, brighter pixels are the ones with larger weights. One can note that even for Cam 7, which is the farthest from the virtual

Figure 3.5: An example multiview data set.

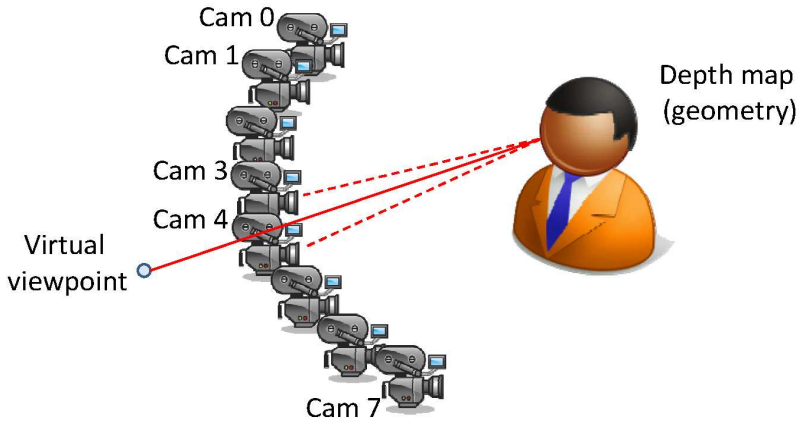


Figure 3.6: The rendering process for multiview video.

viewpoint, there are still pixels being used due to occlusions. Naturally, during compression of the multiview video, the pixels with high weights shall be encoded with high quality, while the remaining pixels can be encoded with low quality.

3.3.2 ***Mapping Weights to QPs***

The next step is to map the weights estimated for each pixel into the quantization parameters. The aim is to encode with higher fidelity portions of the image that are more likely to be used

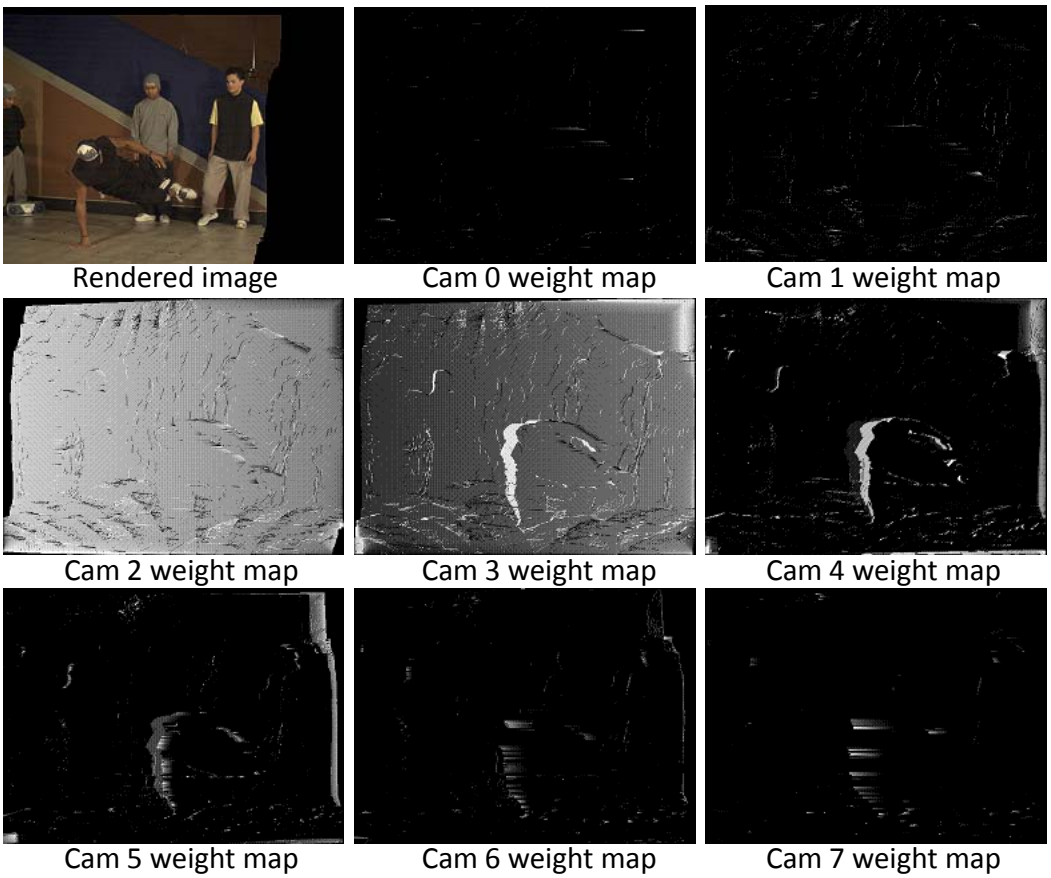


Figure 3.7: The weight maps generated by the rendering process.

by the receiver to synthesize the final view. For that aim, a modified version of the H.264/AVC codec, which allows to specify the QP for each macroblock, was used. In [42], the authors propose the following ad-hoc mapping between the pixels and the QPs:

$$QP_{mb} = QP_{base} - 6log_2\sqrt{\frac{1}{|\mathbb{M}|}\sum_{i=1}^{|\mathbb{M}|}\omega_i^2} \tag{3.1}$$

where $|\mathbb{M}|$ is the cardinality of the 16x16 macroblock, ω_i is the predicted weight for each pixel in the macroblock and QP_{base} is the parameter that controls the overall compression ratio. The ad-hoc relation between ω_i and QP_{mb} in Eq. (3.1) has two main components: the quadratic averaging of the pixel weights within the macroblock and the logarithmic mapping between that average and QP_{mb}. It takes into account the logarithmic scale of QP in H.264/AVC, which doubles the quantization step for each increment of 6 in QP [46].

3.3.3 ***Optimum Mapping between Weights and QPs***

The objective of this work is to optimize the ad-hoc relation between ω_i and QP_{mb} in Eq. (3.1). In order to achieve this optimization, the relation is initially derived by considering that the estimated value $\hat{p_s}$ of a pixel in the synthesized image can be expressed as the following linear combination of the pixels in the several video views:

$$\hat{p_s} = \sum_{i=1}^{|\mathbb{P}|} \omega_i \hat{p_i} \tag{3.2}$$

where $\mathbb{P}$ is the set of views that contribute to the formation of the estimated pixel $\hat{p_s}$. The set $\mathbb{P}$ and the weights ω_i are obtained by using the view synthesis algorithm briefly described in Section 3.3.1.

As the assumption is that each macroblock is actually quantized, the pixels p_i from the several views that will compose the final synthesized pixel are quantized with different QPs. Therefore, the objective is to find the set of QPs that minimizes the distortion of the actual synthesized pixel p_s. Considering that all pixels within the macroblock are quantized with the same QP_{mb}, the rate distortion function can be reduced to the following optimization:

$$R(D) = \underbrace{min}_{\sum_{i=1}^{|\mathbb{P}|} D_i(QP_{mb})} \sum_{i=1}^{|\mathbb{P}|} max\left\{R_i(QP_{mb}), 0\right\} \tag{3.3}$$

where $D_i(QP_{mb})$ is the mean square distortion of the macroblock under consideration.

Using Lagrange multipliers, the optimization can be performed by

$$J(D) = \sum_{i=1}^{|\mathbb{P}|} R_i(QP_{mb}) + \lambda \sum_{i=1}^{|\mathbb{P}|} D_i(QP_{mb}). \tag{3.4}$$

Assuming that all pixels within the same macroblock have the same weight[1] $\omega_i = \omega_{mb}$, Eq. (3.4) can be combined with Eq. (3.2) and be rewritten as

$$J(D) = \sum_{i=1}^{|\mathbb{P}|} R_i(QP_{mb}) + \lambda \sum_{i=1}^{|\mathbb{P}|} \omega_i^2 (\hat{p_i} - p_i)^2 \tag{3.5}$$

$$J(D) = \sum_{i=1}^{|\mathbb{P}|} R_i(QP_{mb}) + \lambda \sum_{i=1}^{|\mathbb{P}|} \omega_i^2 D_i(QP_{mb}) \tag{3.6}$$

Differentiating Eq. (3.6) with respect to $D(QP_{mb})$ and setting equal to 0, the obtained optimality condition is

$$\partial R(QP_{mb}) = -\lambda \, \partial \omega_{mb}^2 \, D(QP_{mb}), \tag{3.7}$$

or alternatively,

$$\frac{\omega_{mb}^2 \, \partial D(QP_{mb})}{\partial R(QP_{mb})} = constant. \tag{3.8}$$

Eq. (3.8) indicates that the derivative of the rate distortion curve scaled by ω_{mb}^2 should be constant across all macroblocks. It gives the intuition behind the optimization, but does not show a straightforward way to choose the best QP for each macroblock. In order to obtain the optimum set of QPs, the initial assumption that the variable being quantized is a Gaussian variable is made. For Gaussian sources $\mathbb{N}(0, \sigma^2)$ with squared error distortion, it is possible to express the distortion in terms of the rate as [47]:

$$D(R) = \sigma^2 2^{-2R} \tag{3.9}$$

As can be deduced from Eq. (3.9), for Gaussian sources, the distortion reduces by 6 dB for each increment of 1 bit in the rate, or equivalently, each halving of quantization step. In the logarithmic scale of QP in H.264/AVC, each halving of the quantization step corresponds to a

[1]The question of how properly average the weights within a macroblock is studied in Section 3.3.4.

reduction of 6 in QP. Therefore,

$$D(QP_{mb} - 6) = \frac{1}{2} D(QP_{mb}), \tag{3.10}$$

and the following relation can be derived:

$$\frac{D_{mb1}}{D_{mb0}} = 2^{\frac{(QP_{mb1} - QP_{mb0})}{6}} \tag{3.11}$$

Through the combination of Eq. (3.9) and Eq. (3.8), the optimality condition for a Gaussian variable is derived:

$$\frac{\omega_{mb}^2 \, \sigma^2 \, \partial \left(2^{-2R(QP_{mb})}\right)}{\partial R(QP_{mb})} = constant \tag{3.12}$$

$$\omega_{mb}^2 \, \sigma^2 \underbrace{\sigma^2 \, 2^{-2R(QP_{mb})}}_{D(QP_{mb})} \underbrace{ln(2)(-2)}_{constant} = constant \tag{3.13}$$

$$\omega_{mb}^2 \, D(QP_{mb}) = constant \tag{3.14}$$

which means that $\omega_{mb0}^2 \, D(QP_{mb0}) = \omega_{mb1}^2 \, D(QP_{mb1})$, for any macroblocks named 0 and 1 with the same variance σ^2.

Assuming that all pixels in macroblock 0 have the same weight $\omega_{mb0} = 1$, the combination of Eq. (3.11) with Eq. (3.14) results

$$QP_{mb1} = QP_{mb0} - 6 \, log_2 \omega_{mb1}^2. \tag{3.15}$$

One can note that Eq. (3.15) is very similar to Eq. (3.1) used in [42]. This similarity comes to the conclusion that the mapping between ω_i and QP_{mb} in Eq. (3.1) is quasi-optimum for Gaussian sources. However, since typical images do not fit well a Gaussian distribution, the exploration of a more appropriate typical RxD curve is proposed.

The RxD curve for the *Breakdancers* sequence shown in Fig. 3.5 was computed with the original H.264/AVC codec and plotted in Fig. 3.8. The distortion can also be plotted as a function of the quantization parameter, as in Fig. 3.9. In both plots, 100 frames of the sequence were considered and the average data among all 8 views was computed.

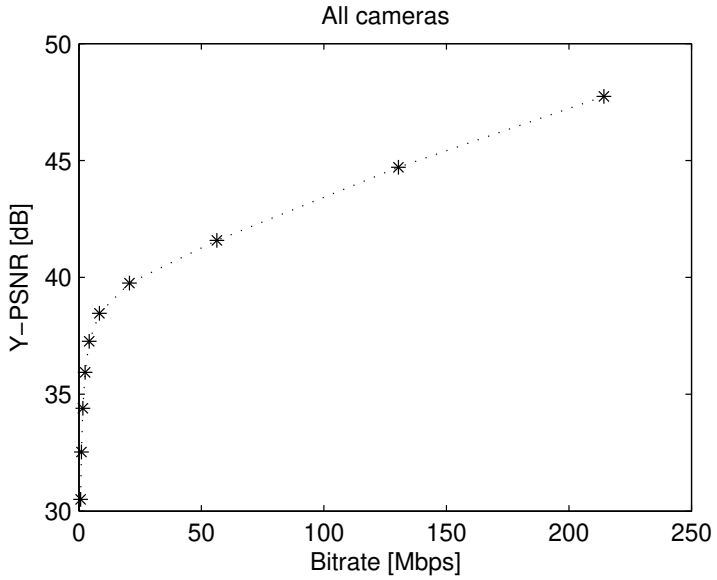


Figure 3.8: Y-PSNR x rate curve for the *Breakdancers* sequence.

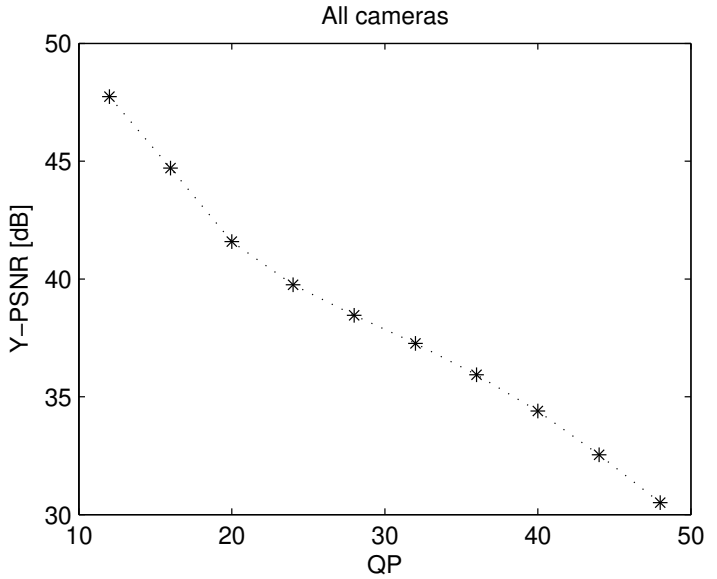


Figure 3.9: Y-PSNR x QP curve for the *Breakdancers* sequence.

As indicated by Eq. (3.8), the derivative of the rate distortion curve scaled by ω_{mb}^2 should be constant. Therefore, the rate distortion curve is piecewise linearized in a way that all line pieces have the same derivative, in other words, the same slope (or gradient).

To start the process of obtaining the optimum QP_{mb}, the QP_{base} is mapped to the corresponding $PSNR_{base}$ in Fig. 3.9. It is important to emphasize that QP_{base} controls the overall compression ratio. Following, the scaled PSNR is related to its line piece in Fig. 3.8 and the

corresponding bit-rate is obtained. Finally, this bit-rate is related to the optimum QP_{mb}. Therefore, instead of applying Eq. (3.15), the piecewise linearized rate distortion curve of the particular sequence is now being used to achieve the best mapping between ω_i and QP_{mb}.

A few points are worthy mentioning here. Firstly, although, for simplicity, the whole sequence was used to obtain the RxD curve, it is expected that performance should be approximately as good as if only past frames are used. Secondly, better fitting of the RxD curve, should further improve results. For example, different curves could be used for I, P and B frames, or, in the extreme case, the optimization could be done at the macroblock level by computing several encoding possibilities, which is one of the options in many H.264/AVC encoders.

3.3.4 ***Averaging the Weights***

The derivation in the previous section assumed all pixels within the macroblock were used in the final image with the same weight ω_{mb}. This is not usually the case. However the whole macroblock must be quantized with the same QP. Therefore, the best way to obtain a representative (i.e., average) weight for representing the whole block is studied here.

The final contribution of each pixel $\hat{p}_i$ is given by $\omega_i \hat{p}_i$, as defined in Eq. (3.2). Under the assumption that each pixel is independent and has the same contribution to the number of bits required to encode the macroblock, the rate distortion relation could be optimized by simply writing the distortion equation for a single pixel, and deriving it in relation to the rate, exactly as performed between MBs in the previous section. However, herein the pixels are highly correlated, and the cross terms can not be set to zero.

In [42], the quadratic averaging of the weights is used as representative for the macroblock, i.e.,

$$\omega_{mb} = \sqrt{\frac{1}{|\mathbb{M}|} \sum_{i=1}^{|\mathbb{M}|} \omega_i^2}. \tag{3.16}$$

Therefore, instead of deriving a complex mathematical model, the simple choice of the norm in the average is verified. More specifically, the mean-square norm averaging in Eq. (3.16) is generalized

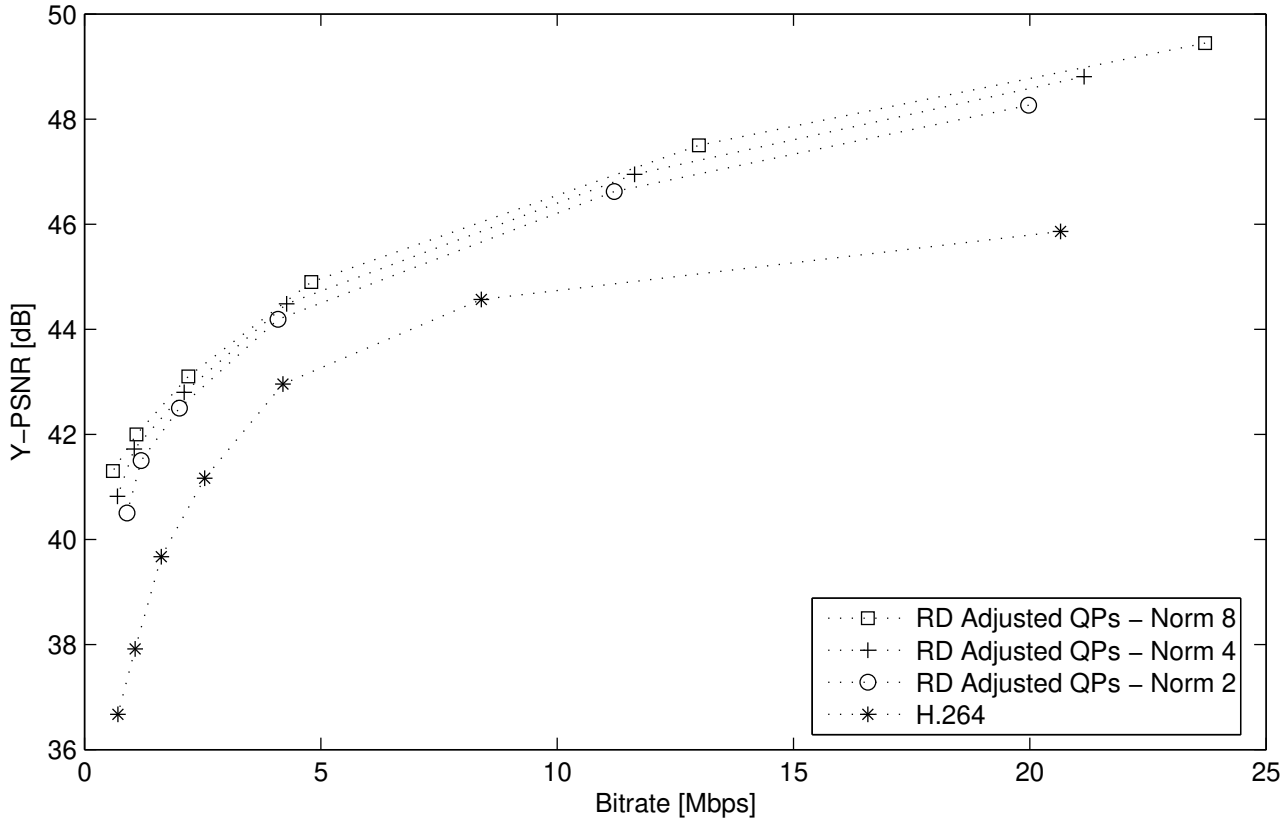


Figure 3.10: Y-PSNR x rate curves for the *Breakdancers* sequence. The PSNR is computed between the synthetic image using uncompressed frames and the synthetic images using each of the compression methods. The lower curve is standard H.264/AVC. The upper curves refer to the optimized QP, when using norms of 2, 4, and 8 to average the weights within a macroblock. The best results were obtained using L=8.

to the L-norm

$$\omega_{mb} = \left(\frac{1}{|\mathbb{M}|} \sum_{i=1}^{|\mathbb{M}|} \omega_i^L \right)^{1/L} \tag{3.17}$$

and the choices of norm $L = 2$, 4 and 8 are experimentally compared. The results are discussed in next section.

3.4 Results

The results of the optimum mapping for the QP proposed in Sections 3.3.3 and 3.3.4 is shown for the *Breakdancers* sequence in Fig. 3.10. Using the same approach as in [42], the PSNR is

computed between the image synthesized using the compressed frames and the image synthesized using the uncompressed frames. One can note in Fig. 3.10 that the improvement over the standard H.264/AVC is on the order of 2dB, or, equivalently, about 50% of the rate can be saved. Early comparison with the non-optimized QP mapping do not show significant difference. One possible explanation is that the larger contribution to the rate is introduced by the few macroblocks where the weights are high, and thus, the influence of the blocks quantized with higher QPs is small. Another possibility is the non-optimized averaging of weights within a macroblock, which may in turn affect the optimality of the QP mapping.

One can also note in Fig. 3.10 that the choice of L_{norm} affects the results. By moving from L_2 norm to an L_8 norm, an improvement of around 0.3 dB can be observed.

3.5 Conclusions

A recent paper introduced the idea of optimizing the encoding of multiview video based on the expected contribution of each macroblock on the final image [42]. In this work, the choice of allocation of rate between macroblocks on that proposal was analyzed. It was found that the proposed QP allocation in [42] is quasi-optimum for Gaussian sources and when the pixels within a macroblock have the same weight in the final image. Results for the generic case of a given RxD curve were derived and a few choices for averaging the weights within a macroblock were analyzed.

Chapter 4

Local Adaptive Scanning for JPEG XR

4.1 Introduction

In a block transform image coder, coefficient scanning is the process of sorting transform coefficients in a "descending on the average" order before the entropy coding stage. The coefficient scanning stage of the coding process was already exploited in Chapter 2, where optimized scanning orders were developed for the several 3-D transforms implemented in FEVC. In this chapter, a new *adaptive* 2-D scanning order will be proposed for the JPEG XR image codec [48–50], conversely to the optimized, but *fixed*, 3-D scanning orders presented in Chapter 2. Another difference is that JPEG XR applies various prediction methods to reduce the coefficient entropy before the coefficient scanning stage. Therefore, the *coefficients* scanned are actually the *post-prediction residuals*, once they are the differences between the transform coefficients and their predicted values.

JPEG XR is the first standard image codec that implements an *adaptive* scanning order. The preceding image codecs performed the coefficients scanning by a fixed order (such as zig-zag scan or the optimized fixed orders proposed in Chapter 2) or by selecting one out of a collection of precomputed scanning orders and then encoding the selection [51–53].

The purpose of this work is to exploit whether there is room for improvement in the JPEG XR adaptive scanning order. This exploration was motivated by the computation of an optimistic loose bound on the performance of JPEG XR. It was found that when all transform coefficients are scanned in an exactly descending order and the reordering permutation is not encoded (thus disregarding its corresponding entropy), the compression rate in JPEG XR can be improved from 24% for relatively smooth images to 10% for images rich in details, for encoding rates from 0.5 to 4 bpp, as shown in Fig. 4.1. Intuitively, for a given compression rate, the compression gain

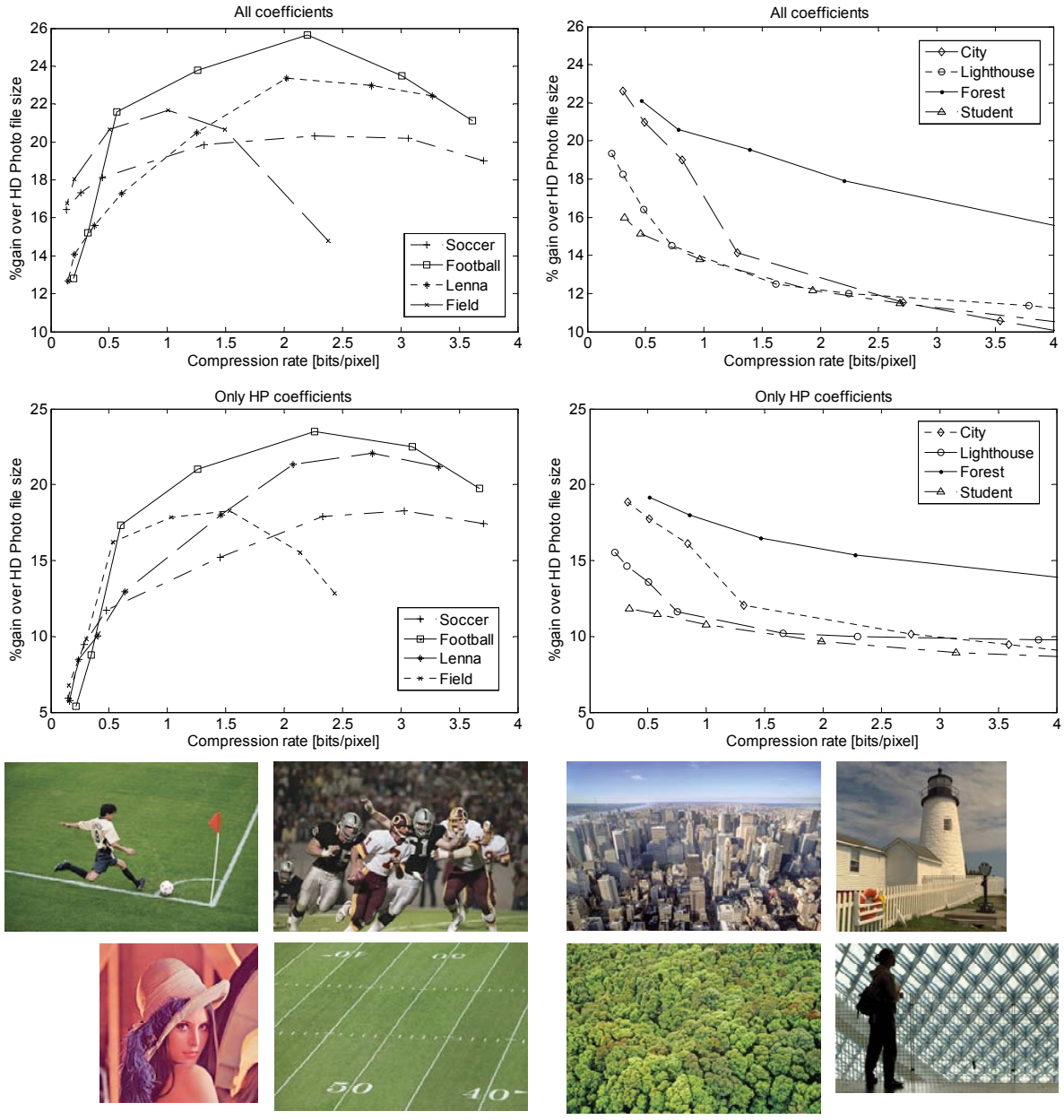


Figure 4.1: Improvement in compression ratio enforced over the original JPEG XR for the cases when all or HP-only coefficients are sorted in exactly descending order. The highpass (HP) coefficients are identified in Section 4.2.2.

from reordering reduces as the coefficient prediction efficiency increases. One can see that PSNR is not affected, as coefficient scanning does not introduce any additional data loss beyond that from quantization.

This chapter initially describes the JPEG XR image codec in Section 4.2. Specially, the JPEG XR coefficient scanning process is detailed at Section 4.2.5. The new adaptive scanning orders proposed here for the JPEG XR are obtained by localized and hybrid methods and are all

described in Section 4.3. These orders do not require changes in either the other coding/decoding stages or in the bitstream definition and have been partially published in [54]. The Sections 4.4 and 4.5 present the implementation results and conclude the chapter.

4.2 The JPEG XR Image Codec

JPEG XR (Joint Photographic Experts Group Extended Range) is a new international standard for image coding based on a Microsoft technology known as HD Photo [48–50]. HD Photo is a still image file format that offers PSNR performance comparable to JPEG 2000 with computational and memory performance more closely comparable to JPEG standard [55, 56].

4.2.1 ***General Characteristics***

One advantage of JPEG XR over previous standards is that it provides High Dynamic Range (HDR) image encoding while requiring only integer operations (without any division). Therefore, it supports both lossy and lossless compression. Other important advantages are that it offers a low-complexity solution for the scalable coding of high-resolution images and a tile structure support that is feasible, in terms of compressed file size overhead, even for tiles of size 64x64 pixels. This JPEG XR feature makes it suitable for the efficient design of hardware implementation, memory buffers and also for region-of-interest (ROI) coding applications. Tiles are image regions that can be decoded independently. The tile support feature is also present in other image codecs, such as JPEG 2000. However, the file size overhead for JPEG 2000 is significantly higher than for JPEG XR. In particular, the smaller the tile size, the more a JPEG 2000 encoder acts as a block-based encoder.

Examples of ROI coding applications include video surveillance and mobile client-server networks. In the first application [57], to address privacy concerns, face regions are detected in the image frames and subsequently scrambled in the transform domain. In the second application [58], a user with a mobile device may, for instance, communicate with the server as follows. First, after having selected a particular coding format, the user may request a low-resolution image that fits the limited display resolution of the device used. This request can be implemented by extracting a low resolution version of the selected image (i.e., by exploiting JPEG XR spatial scalability). Next, the user may request a high-quality version of a semantically meaningful region (ROI) that fits the resolution of the device at a high quality level. Determining a ROI can be done

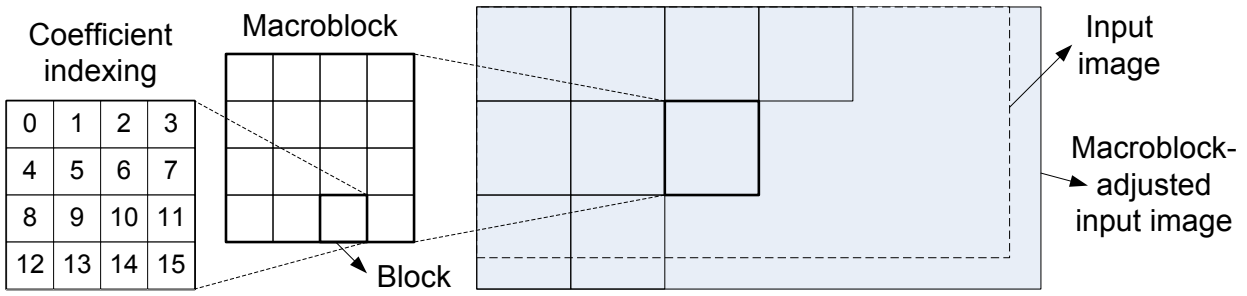


Figure 4.2: Image structure hierarchy and intra-block coefficient indexing.

either manually, with user intervention, or automatically, by making use of a content-aware image resizing tool.

The compression efficiency is highly impacted by the tiles size. The smaller the tile size, the finer the interactive ROI selection, but the lower the coding efficiency. Some guidelines regarding how an image should be tiled to enable fast local access by aligning the ROI access with the tile structure are proposed in [59].

4.2.2 ***Photo Overlap and Core Transforms***

JPEG XR employs a two-stage integer lapped biorthogonal transform composed of the optional pre-filter Photo Overlap Transform (POT) followed by the Photo Core Transform (PCT).

If the input image is being coded with only one tile, initially it is entirely adjusted into macroblocks, which are matrices of 4x4 blocks, each of them a 4x4 pixel matrix, as shown in Fig. 4.2. Then, the POT is optionally applied to 4x4 areas evenly straddling blocks. The pre-filter is also applied to 2x4 or 4x2 boundary areas. Only the four 2x2 corners are not filtered. Following, the PCT is applied to all 4x4 blocks and this completes the first transform stage. The PCT is inspired by the 4x4 DCT, yet it is fundamentally different. Conversely to the DCT, the PCT is a nonlinear, non-separable, integer and lossless transform [60].

The result of the first transform stage is illustrated in Fig. 4.3 by the green squares in the macroblock structure. These squares represent the 15 highpass coefficients of each of the 16 blocks and constitute the *HP (highpass) band*. The DC coefficients of each of these 16 blocks are then grouped together into a 4x4 structure and the first operation of the second transform stage is the optional application of the POT transform to this structure. The PCT is then applied again and the second transform stage is completed. The result of the second transform stage is illustrated

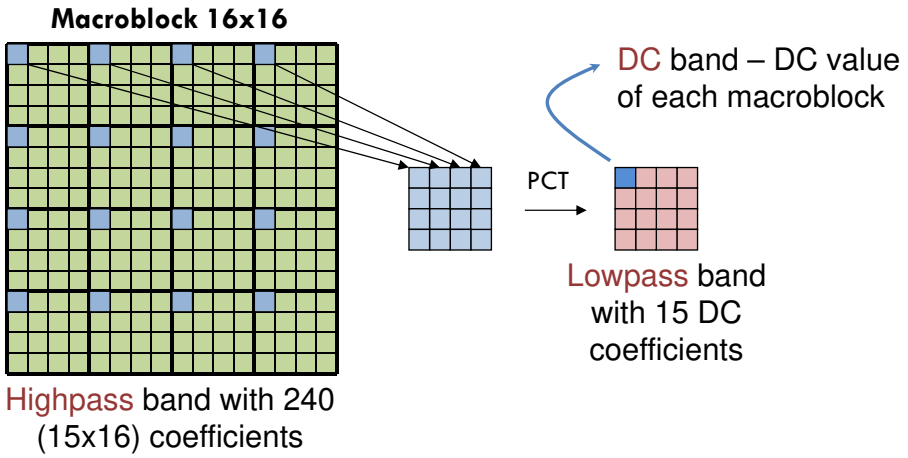


Figure 4.3: PCT coefficient bands.

in Fig. 4.3 by the 15 pink squares that constitute the *LP (lowpass) band*. Finally, the main DC coefficient constitutes the *DC band*.

The employment mode of the optional POT must be tuned with the application. If the application requires fast encoding/decoding and low compression ratio, the POT is not necessary. In this case, the JPEG XR transform is not lapped, once only the PCT is applied twice. However, if the POT is not applied, blocking artifacts can potentially occur, even at high bit-rates for some images. The POT can then be applied in the first transform stage and this mode achieves the best R-D performance for most of the applications. Only if the application requires low bit-rates, the POT can be applied both in the first and in the second stage, at the cost of increased encoding/decoding complexity and possible ringing artifacts in the vicinity of the blocks.

All macroblocks resulting coefficients can be organized in the final JPEG XR bitstream grouped by macroblocks or by bands. In the *spatial mode*, the bits pertinent to each macroblock are located together and the macroblocks are scanned in raster order (left to right and top to bottom). In the *frequency mode*, the DC band carries information of the DC value of each macroblock scanned in raster order. The same principle is applied to the HP and LP bands. There is an additional band named *Flexbits* that carries information regarding the low order bits of the HP band particularly for lossless and low loss cases. For medium and high bit-rates scenarios, the *Flexbits* band is often empty.

Only the frequency mode enables the JPEG XR quality and spatial scalability features. A trivial form of quality scalability can be performed by removing entirely or part of the *Flexbits* band from the bitstream. Spatial scalability is supported by additionally removing the LP and

HP bands, each time resulting in a reduction of the spatial resolution by a factor of four along the horizontal and vertical directions.

4.2.3 ***Quantization***

JPEG XR employs a scalar quantization in which the quantization step size is selected by a quantization parameter (QP). QPs indices range from 1 to 255 and each value is mapped to a particular QP in which the QP is chosen to be an integer from a harmonic scale. Therefore, only shifts operations are necessary to perform the division operations.

One set of QPs is supported for the DC band, up to 16 distinct sets of QPs for the LP band and up to 16 distinct sets of QPs for the HP band. The QPs are the same for the DC band and for all LP and HP coefficients within a macroblock, but can change at every macroblock.

4.2.4 ***Prediction***

The JPEG XR prediction method is applied in all bands. Four modes are allowed for the prediction of the DC band of a macroblock: from left macroblock, from top, from both left and top and no prediction. All these modes are illustrated in Fig. 4.4 which also illustrates the prediction of the LP band. The LP band prediction mode is determined by the DC band prediction mode together with the QPs of the current macroblock and of the macroblock from which DC is predicted. This rule ensures that the prediction of the LP band does not take place across macroblocks with different QPs.

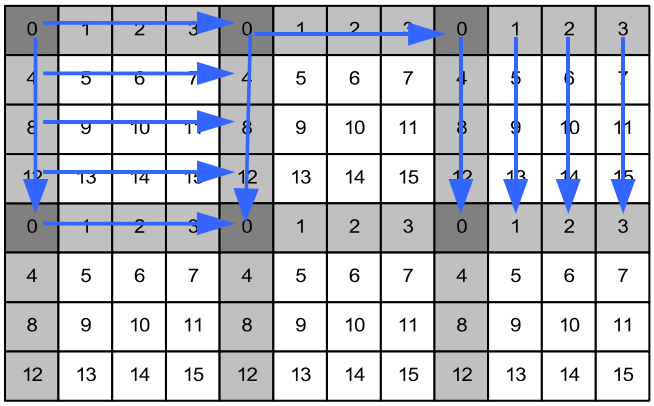


Figure 4.4: DC coefficients in dark grey predicted by: *top left coefficient)* no prediction, *top center and right coefficients)* left mode, *bottom left and right coefficients)* top mode, *bottom center coefficient)* both top and left mode. LP band in light grey predicted by: *top center block)* left mode, *bottom right block)* top mode, *other blocks)* no prediction.

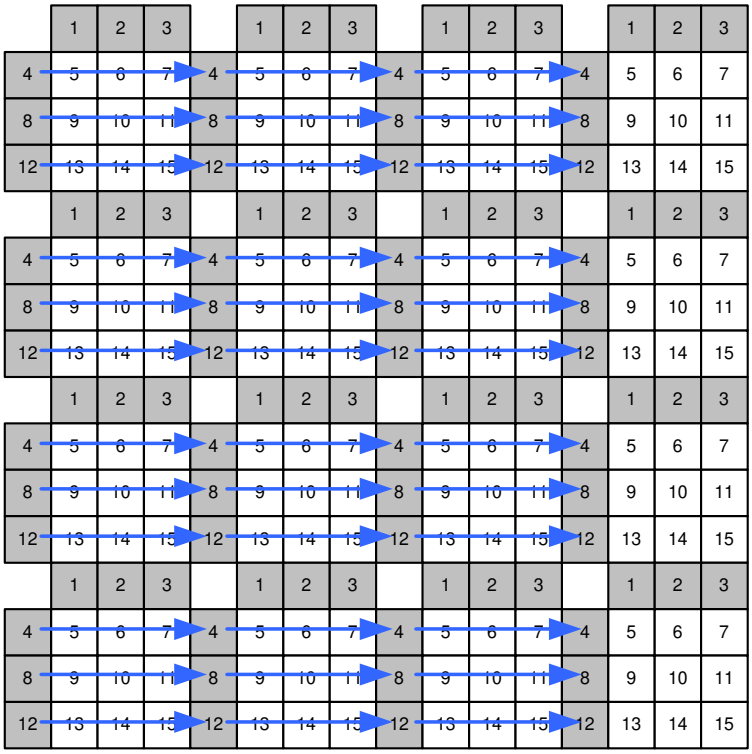


Figure 4.5: Left prediction mode of one macroblock HP band.

The prediction for the HP band is only performed within the macroblock and the same mode is used for all blocks of the macroblock. The allowed prediction modes are: from the left block, from the top block and no prediction. The mode is chosen according to the dominant direction indicated by the LP band. Fig. 4.5 shows the macroblock HP band predicted in the left mode. Prediction in the top mode is similar with the pattern of arrows transposed to point downwards.

4.2.5 ***Coefficient Scanning***

Initially, the DC band is scanned in raster order across all color planes in sequence. The LP coefficients are then scanned per color plane using an adaptive order. Before scanning each color plane of the HP band, each macroblock is divided into four 8x8 sub-macroblocks. Within each sub-macroblock, the encompassed 4 blocks are scanned in raster order and their HP coefficients are also scanned using an adaptive order.

The scanning orders for the LP and HP bands are adaptive, which means that are allowed changing over the course of the image/tile. The causal adaptation rule is based on the latest global probability of nonzero scanned coefficients. Two arrays are employed:

- ***order***$[i], i = 1 \ldots 15$, contains the current scanning order, i.e., ***order***$[3] = 5$ means that the coefficient indexed 5 should be scanned third, and
- ***totals***$[i], i = 1 \ldots 15$, contains the number of nonzero coefficients scanned prior to the current block, i.e., ***totals***$[3] = 24$ means that prior to scanning this block 24 nonzero coefficients have been found at the coefficient indexed 5 (note: ***order***$[3] = 5$).

When scanning the ***order***[i]-th coefficient, if and only if it is nonzero, the associated ***totals***[i] is incremented by one. After scanning a full block of coefficients, ***totals*** is sorted in decreasing order using a single-pass bubble-sort, and the elements of ***order*** are correspondingly reordered to reflect this sorting. Since, there exist two sets of coefficients adaptively scanned, HP and LP, there exist two pairs of corresponding $\{\boldsymbol{order}_x, \boldsymbol{totals}_x\}$ arrays, where $x \in \{\mathrm{HP}, \mathrm{LP}\}$.

The arrays are initialized at the start (top left macroblock) of the image/tile. The ***totals*** arrays, both for the HP and LP band, are always initialized with a constant array of descending values $\boldsymbol{t}_0 \equiv \{28, 26, 24, \ldots, 0\}$. Three predefined scanning patterns are used for initializing the ***order*** arrays. The $\boldsymbol{order}_{HP}$ is set depending on current macroblock dominant orientation computed as vertical, horizontal, or neutral at the DC prediction step. The HP scanning pattern for a vertically dominant macroblock is $\boldsymbol{o}_{\mathbb{V}} \equiv \{10, 2, 12, 5, 9, 4, 8, 1, 13, 6, 15, 14, 3, 11, 7\}$, where the indexing is performed according to the schedule shown in Fig. 4.2, and the HP scanning pattern for the other two macroblocks (horizontally dominant and with no dominant orientation) is $\boldsymbol{o}_{\mathbb{H}} \equiv \{5, 10, 12, 1, 2, 8, 4, 6, 9, 3, 14, 13, 7, 11, 15\}$. The $\boldsymbol{order}_{LP}$ array is also always initiated with $\boldsymbol{o}_{\mathbb{H}}$.

As expected, neither of the scanning patterns include the DC coefficient. The scanning patterns are invariant across macroblock color planes. The ***totals*** arrays are reset at every eight macroblocks and at the start of every tile, while the ***order*** arrays are reset only at the start of tiles. If the image is untiled, the ***order*** arrays are never reset. It is important to note that the periodic resets of ***totals*** are the only action taken in JPEG XR coefficient scanning process to address the content locality for the encoded image.

A global scheme for adaptive scanning of coefficients should consider the average energy of coefficients at each index. Fig. 4.6 shows the average energy of HP band coefficients depending upon their index. One can see that coefficients with indices 10, 5, 12, 1, 2, and 8 are dominant, and thus they should be scanned first. The energy distribution is relatively insensitive to the

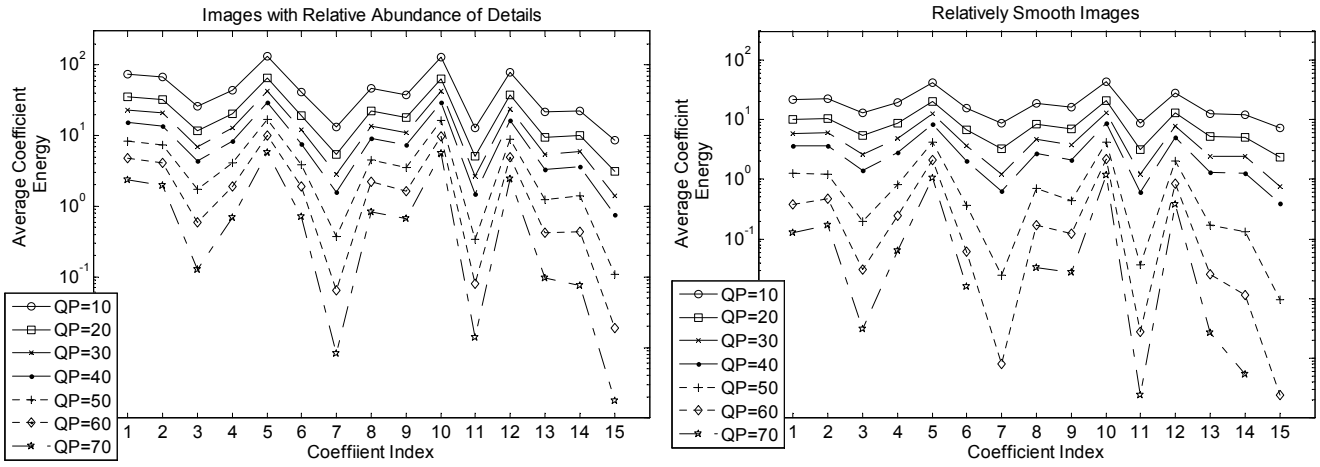


Figure 4.6: The average HP band coefficient energy for the two groups of images classified in Figure 4.1, with QP varying from 10 to 70.

level of detail present in the image. The results obtained with this experiment are in-line with the predefined scanning patterns used by JPEG XR for initializing the ***order*** arrays.

4.2.6 ***Entropy Coding***

Before the entropy coding, an adaptive coefficient normalization is applied by JPEG XR. The normalization is performed by tracking a statistical measure of variance of transform coefficients. Based on this measure, the current transform coefficient is regrouped into dyadic bins. Instead of encoding the transform coefficient, its bin id is sent using an efficient entropy code. Subsequently, an in-bin address locating the transform coefficient within its corresponding bin is sent with a fixed length code. The sign information is also sent when necessary.

Adaptive coefficient normalization is used for all frequency bands and the fixed length in-bin addresses of the HP band are referred to as *FlexBits*.

4.3 Proposed Localized Scanning Orders

Recent work has addressed improvements to JPEG XR, especially to the lapped transforms [60–65], but the efficiency of the coefficient scanning stage has not been exploited in the open literature yet. Therefore, as previously mentioned, the purpose of this work is to exploit if there is room for improvement in the JPEG XR adaptive scanning order.

The JPEG XR coefficient scanning process defined in Section 4.2.5 uses a simple adaptive scanning order for all blocks in the image/tile and an adaptation rule that classifies coefficients

simply as zero/nonzero. This heuristic is global in its construction, and thus it might not handle efficiently abrupt changes in the image content. Therefore, the primary objective is to analyze scanning order heuristics that are locally adaptable to the image content. Towards that goal, several per-block scanning order heuristics were created. These heuristics exploit the inter-block correlation, for block sizes as small as 4x4. The main guideline for all considered heuristics was the premise that the coefficient magnitudes in one block are likely similar to the coefficient magnitudes at the same position in neighboring blocks. In other words, the energy spreading patterns of neighboring blocks should be similar.

Several kinds of averaging filters were tried to affect the scanning order based upon localized heuristics. Each filter is used to reorder blocks as follows. In general, a filter $\mathcal{F}$ is defined for each coefficient $c_i(b)$ of the current block b as

$$\mathcal{F} \equiv \left\{ f_i(b) = \frac{1}{|\mathbb{N}(b)|} \sum_{x \in \mathbb{N}(b)} w(x) c_i(x), i = 1, \ldots, 15 \right\}, \tag{4.1}$$

where $\mathbb{N}(b)$ is a set of blocks neighboring to b considered for averaging and $w(x), x \in \mathbb{N}(b)$, is an arbitrary real scalar applied as weight to each coefficient individually. Coefficients coming from each block x from b neighborhood can be weighted distinctly. In order to be able to recover the ordering at the decoder, only blocks that were already parsed by JPEG XR raster order block scanning are considered for $\mathbb{N}(b)$. After a specific filter is applied, the resulting vector $\mathcal{F}(b) \equiv \{f_1(b), \ldots, f_{15}(b)\}$ is sorted in decreasing order with a resulting permutation π_b. This permutation is applied to sorting the coefficients of b and the result $\pi_b(b)$ is passed to JPEG XR run-length encoder. Clearly, in order to recover π_b, the decoder must compute $\mathcal{F}(b)$ from already decoded blocks, and then to establish the correct order of coefficients as $b = \pi_b^{-1}(\pi_b(b))$.

The six averaging filters that performed well in experiments are shown in Fig. 4.7. Each filter was developed to address blocks with specific correlation patterns. The filters differ in the encompassed "block neighborhood" and in the coefficients weighting. The gray levels in Fig. 4.7 indicate the coefficient weights. The darker the color, the higher the weight. Each increase in gray level indicates a doubling in weight. The "block neighborhood" encloses up to four blocks for the average filters and up to 12 blocks for the expanded average filters $\mathcal{F}_3$ and $\mathcal{F}_6$. That depends

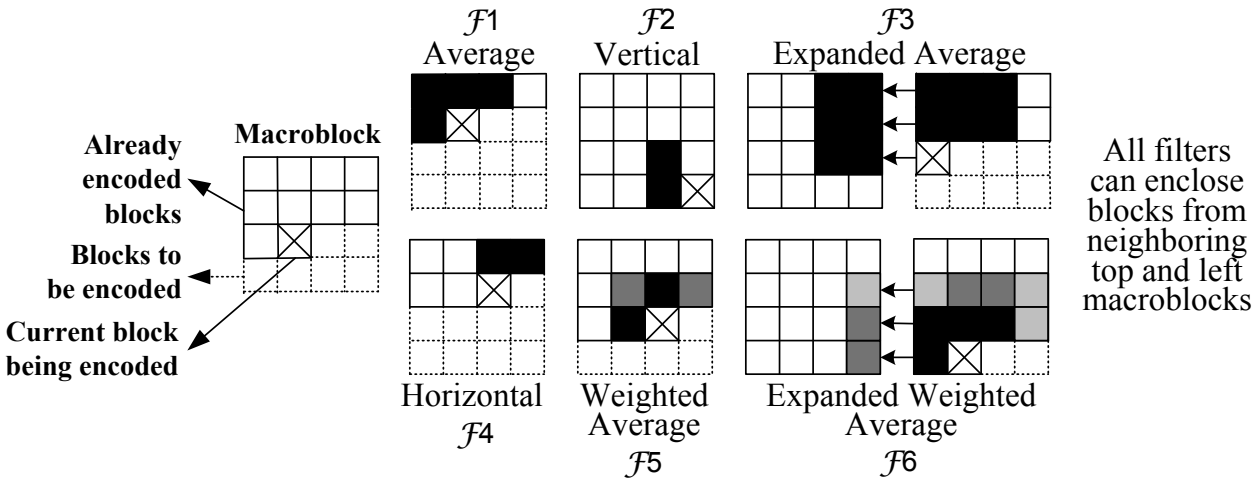


Figure 4.7: A collection of six localized averaging filters used for scanning coefficients in JPEG XR HP band.

on the block position, as the macroblocks and blocks are encoded in raster order. The averaging filters were applied only to the HP band for two reasons:

1) Fig. 4.1 shows that approximately 90% of the potential gains in the compression ratio are due only to the HP band.

2) The coefficients of the HP band span over smaller localities, which raises the conjecture that localized scan-order heuristics may be more applicable to this band.

It is worthy to note that the proposed coefficient ordering techniques would marginally increase the computational complexity of the overall codec, both at encoding and decoding.

4.3.1 ***Hybrid Techniques***

Generally, blocks located in highly correlated neighborhoods (relative to the imposed QP) are usually well predicted and result in coefficient residuals of low energy. As a consequence, these coefficients can usually be considered too randomized, and thus JPEG XR original scanning order with its global adaptation rule performs relatively well for such blocks. On the other hand, blocks that still contain high energy coefficients even after the prediction step, are usually located in areas of the image with high frequency content. For such blocks, localized scanning order heuristics often outperform global ones. This trade-off is considered in more details in the following.

First, an order-difference metric will be defined as follows. For two orders, $\boldsymbol{x}$ and $\boldsymbol{y}$, of the same set of elements $\mathbb{Z}$, $\boldsymbol{x}= \pi_x(\mathbb{Z})$ and $\boldsymbol{y}= \pi_y(\mathbb{Z})$, the distance metric is

$$\Delta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\mathbb{Z}|} \omega_i |y_i - x_i| \tag{4.2}$$

where x_i and y_i denote the i-th element of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, and ω_i denotes a scaling factor designed to emphasize the importance of ordering high-energy coefficients appropriately. The scaling factors are set as:

$$\{\omega_i, i = 1, \ldots, 15\} = \{16, 16, 8, 8, 4, 4, 2, 2, 1, 1, 1, 1, 1, 1, 1\} \tag{4.3}$$

to reflect upon the exponential rate of coefficient attenuation in most sorted sources. The number of relevant high energy coefficients changes depending upon the enforced bit-rate. One can observe from Fig. 4.6 that the signal energy is concentrated in fewer coefficients for increasing values of QP. For QPs smaller than 40 (i.e., bit-rates higher than 1 bpp), the energy is basically all concentrated in the top eight highest-energy coefficients and for QPs higher than 60 (bit-rates smaller than 0.5 bpp), the energy is almost all concentrated only in the top 2 highest-energy coefficients. In fact, in this scenario, optimal scanning of these two coefficients in the top two order positions is usually enough to reach the performance obtained when all coefficients are scanned in an exact descending order (see Fig. 4.1). Therefore, the configuration of the scaling factor ω_i reflects these considerations and improves compression especially at low bit-rates.

Consider the following experiment: for each block b in a test image, the perfect descending order of its coefficients $\pi_S(b)$ is initially computed. Then, the scanning orders resulting from the application of all six filters introduced in Fig. 4.7, $\pi_1(b), \ldots, \pi_6(b)$, are also computed. When all these orders are compared among each other and to the original JPEG XR scanning order $\pi_7(b)$ the filter indexed as

$$j(b) = \arg\min_{i=1\ldots7} \Delta(\pi_i(b), \pi_S(b)) \tag{4.4}$$

is considered the "best performing filter."

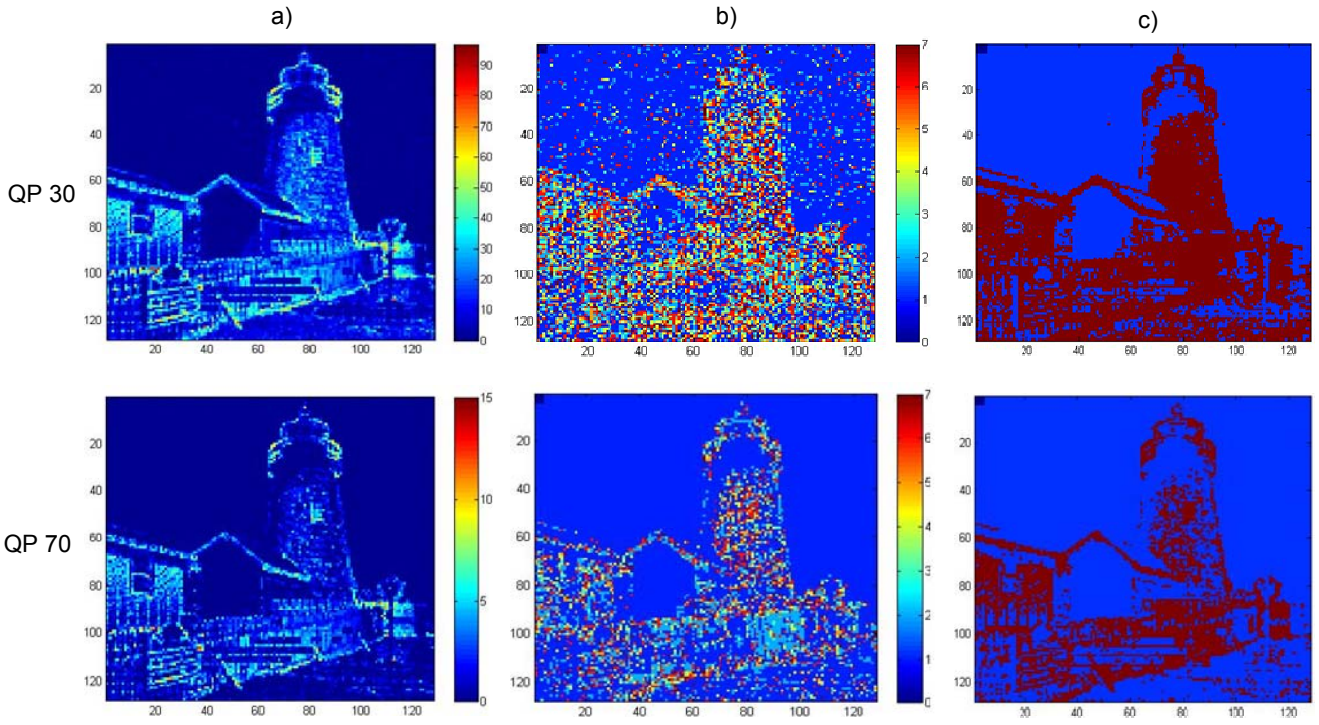


Figure 4.8: a) Energy amount per 4x4 block areas for the "Lighthouse" image, b) Best performing per-block filters denoted in figures' colorbars and indexed using the following color schedule: 1 - Global (original JPEG XR), 2 - Average, 3 - Weighted Average, 4 - Horizontal, 5 - Vertical, 6 - Expanded Average, 7 - Weighted Expanded Average, and c) Best performing per-block filters, where blue is the Global filter and red is the Expanded Average filter.

Fig. 4.8 illustrates for two different QP values of 30 and 70, applied to the "Lighthouse" image:

a) the block energy profile of the resulting coefficients;

b) the indexes of "winning" filters for each specific block.

In support of the data presented in Fig. 4.8(b), Table 4.1 presents the ratio of blocks where a specific filter performed the "best" (see Eqs. 4.2–4.4) among the proposed set of filters over all tested images presented in Fig. 4.1. Localized filters in the table are marked 1–6 according to Fig. 4.7.

One can see in Fig. 4.8 that for blocks with low energy, the global filter typically performs well, while the localized filters usually outperforms the global filter in all other cases. Based upon the results presented in Table 4.1, the expanded average filter $\mathcal{F}_3$ is selected as the "best" among the localized filters as it captures the majority of high-energy blocks. When compressing images from Fig. 4.1 at high bit-rates (QPs around 30), localized averaging filters outperform the global heuristic roughly half the time. As expected, this ratio is reduced in low bit-rate scenarios,

QP	Original	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$	$\mathcal{F}_5$	$\mathcal{F}_6$
30	52.9%	11.4%	5.7%	13.5%	6.3%	5.8%	4.4%
40	66.4%	8.3%	3.8%	9.8%	4.2%	4.4%	3.1%
50	71.7%	6.9%	3.0%	9.7%	3.3%	3.1%	2.3%
60	77.0%	5.4%	2.5%	9.4%	2.4%	2.1%	1.2%
70	84.5%	3.5%	1.3%	8.3%	1.2%	0.9%	0.3%

Table 4.1: Ratio of blocks where a specific filter performed as the "best" among the proposed set of filters over all images presented in Fig. 4.1. Localized filters are marked 1–6 according to Fig. 4.7.

but even at QPs around 70, approximately 15% of all blocks are scanned more efficiently with localized filters.

4.3.2 ***A Candidate Hybrid Coefficient Scanning Technique***

Motivated by the result illustrated in Fig. 4.8, the global and local scan orders were unified using a simple thresholding technique. Essentially, in the proposed construction, if the energy of a specific block is lower than E_T, then the original (global) JPEG XR scanning order is selected as the "winner" for the block. In the alternate case, the expanded average filter, identified as the "best" localized filter according to the data in Table 4.1, is selected as the block "winner". This is an example of a hybrid strategy that can be reproduced at the decoder.

It is important to note that this hybrid technique is necessary because the decision metric defined by Eq. (4.4) requires information that is not available at the decoder while decoding a block. The energy threshold E_T could be taken as a constant for all images and established as part of the codec design. Alternatively, it could be adaptively adjusted per image and per quantization parameter at encoding time. The encoder can randomly select a few block samples, empirically compute an adequate value for E_T, and encode it as part of the header of the resulting image file.

Experiments were conducted to evaluate the effectiveness of the hybrid scanning technique. Fig. 4.8(c) shows the "winning heuristics" when the filter choice was restricted to the original (global) filter or the expanded average filter $\mathcal{F}_3$ and the decision between them was based upon a precomputed and optimized energy threshold. One can observe that the expanded average filter is selected for many of the blocks coded with localized filters in Fig. 4.8(b) which suggests that the employed energy thresholds are appropriate. Even though the expanded average filter produces better performance than the original filter for these blocks, it does not cover satisfactorily all localized correlations.

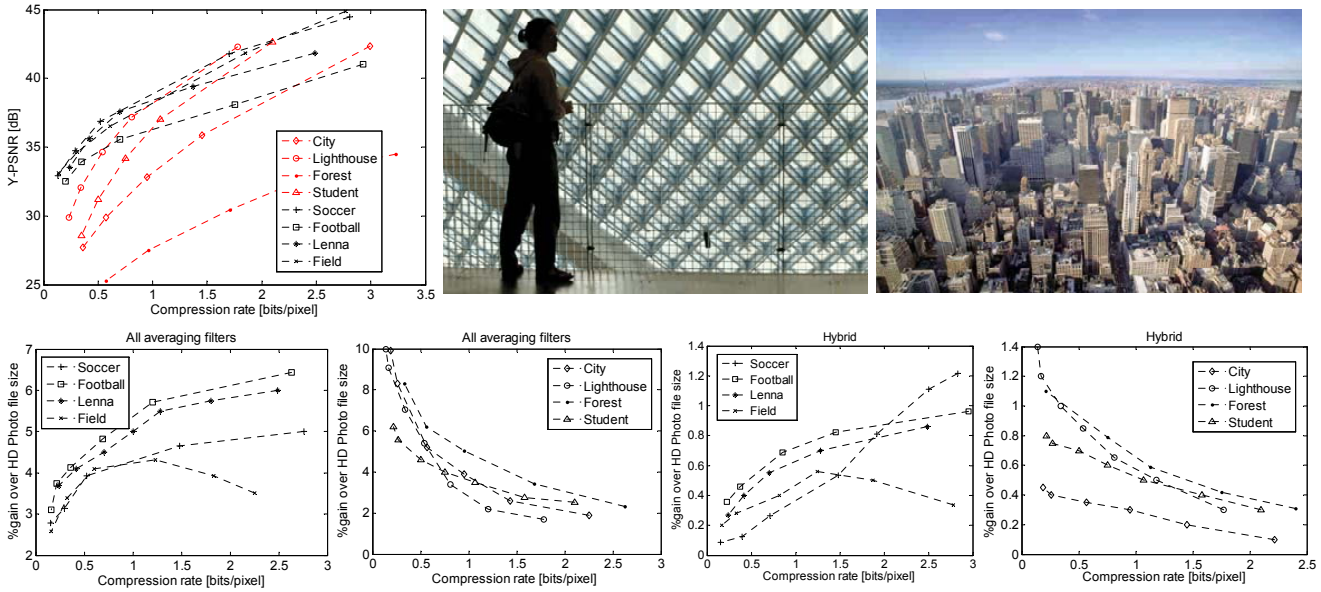


Figure 4.9: *top left*) Y-PSNR for different JPEG XR compression bit-rates, *top center and right*) resulting images compressed with JPEG XR at QP=70, *bottom row*) percentage of file size reduction due to the bound computed using best-of-7 filter and the proposed constructive hybrid technique at each block; results reported for smooth and high-frequency images.

4.4 Results

The hybrid coefficient scanning technique described in Sections 4.3.1 and 4.3.2 was evaluated using the Microsoft HD Photo Device Porting Kit [48] as a basis for implementation and several images selected from a large database of perceptually diverse content [66]. For the experiments, JPEG XR was parameterized as follows: no tiling, spatial mode, one-level of overlap in the transformation stage, no skipped subbands and no chroma sub-sampling. Because at relatively low bit-rates, most coefficients have low magnitudes, a better adaptive scanning order of chrominance planes is not worthy. Thus, the hybrid scanning technique was applied only to luminance planes and the chrominance planes were read using the original (global) JPEG XR scanning order.

Fig. 4.9 summarizes the results. The visual effects of the compression suite are visible in the top row of plots in the figure, which presents the Y-PSNR performance for all images in the benchmark, and two sample images from the database compressed at QP=70. The percentage improvement in compression rate is shown in the bottom row of four plots for both smooth (first and third) and high-frequency (second and fourth) images in the benchmark. The left two sets of curves refer to an optimistic bound in which for each block the "best-of-7" filter choice technique was applied to scan the coefficients (without imposing the overhead to encode the filter selection

in JPEG XR bitstream). The right two sets of curves refer to the constructive hybrid scheme described in Sections 4.3.1 and 4.3.2. In the "best-of-7" case, the obtained optimistic bound shows that images could be compressed up to 10% better. However, the operational (constructive) hybrid scheme resulted in about 1% improvement in the effective compression rate. Therefore, the Y-PSNR performance of the hybrid scheme is highly similar to the JPEG XR performance, shown at the top row of plots in Fig. 4.9.

The level of detail (high frequency content) in images impacts the compression gains significantly. With smooth images the gains tend to rise as the bit-rate increases. The opposite behavior is verified in images with relatively abundant detail. A possible explanation for this is that with smooth images and higher bit-rates, the filter that is based on localized statistics tends to perform better than the original JPEG XR filter. Thus, improved coefficient scanning schemes can produce higher gains in the resulting compression rates. Nevertheless, the relative gains obtained by scanning coefficients using the proposed hybrid heuristic are not significant. Therefore, the global heuristic present in JPEG XR proves to be highly efficient, considering its low computational complexity and the performance obtained using the investigated localized methods.

4.5 Conclusions

Coefficient scanning is typically the last stage of processing a compressed signal in a transform coder, before it is fed to the final entropy encoding stage. In modern high-efficiency coders, coefficients are transformed by sophisticated prediction techniques, which decorrelate (whiten) and reduce the variance of prediction residuals, thus reducing the opportunity for specialized scanning to improve compression performance.

The investigation presented in this chapter showed that although optimistic bounds may indicate a potential for performance gains, in a constructive scanning method that balanced localized averaging filters with global statistics, only about 1% improvement in compression rates was achieved, across a wide range of effective bit-rates. This helps in validating the high efficiency of the approach to coefficient scanning used in JPEG XR.

Chapter 5

HSDT - Hierarchical Signal Dependent Transform

5.1 Introduction

Hierarchical transforms, such as wavelets, have found applications since the multiresolution theory was firstly unified in 1987 [67]. Multiresolution theory successfully incorporates several techniques, such as subband coding, quadrature mirror filtering and pyramidal image processing. Since one of the applications is to represent images at more than one resolution, the usage of hierarchical transforms for progressive image coding has been straightforward.

One of the characteristics of hierarchical transforms is that after its application, although the dependency among the subbands in the same level is highly reduced, there is still dependency among the subbands across different levels, as illustrated by the white squares and arrows in Fig. 5.1. This dependency has been successfully exploited at the entropy coding codec stage since the *zerotrees* concept was introduced by the Embedded Zerotrees Wavelet (EZW) algorithm [23] in 1993. However, this dependency has not yet been effectively exploited at the transform codec stage. Therefore, the main contribution presented in this chapter is a new energy compaction technique, i.e., a transform, that can also exploit these cross-level structural similarities.

The new transform presented here is hierarchical, even though it is applied in a block-based fashion on the coefficients of each level. It is also adaptive and signal dependent, since the results of a number of functions applied to the lower resolution of the signal in each level is included into the transform basis. Based on these features, the transform was named Hierarchical Signal Dependent Transform (HSDT).

The characteristics of the HSDT provide several advantages to this transform. As it is hierarchical, both spatial and quality scalability are naturally provided. Although, unlike other

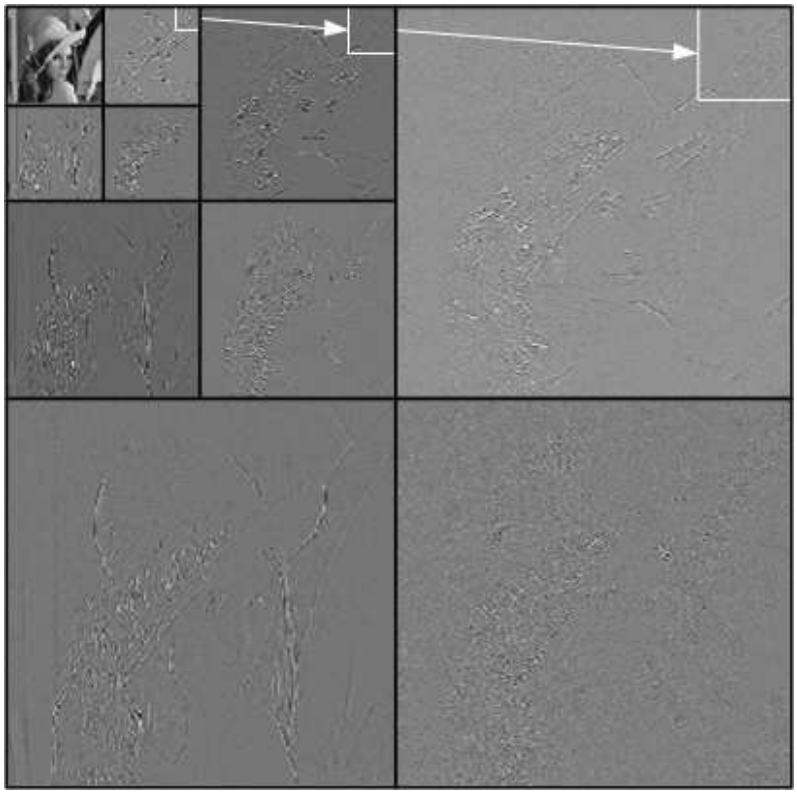


Figure 5.1: Three-level wavelet decomposition of the Lenna image.

hierarchical transforms, there is no lack of localized correlated data access or increased complexity, because the HSDT is applied in a block-based fashion. The common blocking artifacts resulting from the applications of block transforms, specially at low bit-rates, are less perceptible as they are spread among the several levels. As the bit-rate decreases, the distortion takes the form of a gradually increasing blurriness or loss of detail. Additionally, the adaptation scheme exploits the signal dependency between levels by taking advantage of both directionality and neighborhood correlation properties of most images.

As expected by the adaptation process, each block in each level will be coded with a different and optimized basis. The results of a set of adaptive functions applied to the lower resolution signal will compose this optimized basis. The *same* set of adaptive functions is applied to all lower resolution blocks, however the results, i.e., the basis components, are inherently *different* for each block. Therefore, HSDT can not be categorized as a matching pursuit [68] coding scheme, although a parallel with this technique can be made.

In basis pursuit [69], there is a very large number of basis functions and the goal is to find a good fit of the given signal as a linear combination of a small number of the basis functions. In

this context, the function family is linearly dependent and can be referred to as an *overcomplete basis* or *dictionary*. *Sparse coding* seeks the best representation of the input signal through a much smaller basis selected from this overcomplete basis formed through a dictionary learning process. One common approach to basis pursuit is the matching pursuit technique which applies a greedy iterative process. At each iteration, the residual (initially, the whole signal) is projected on all basis vectors and the basis vector that gives the maximum projection is chosen. The residual is then updated and the process is repeated by projecting the new residual and picking the new best basis vector in each iteration until a pre-defined stopping criteria is reached.

In the HSDT, there is a number of adaptive functions that are all applied to the lower resolution blocks. The goal is to order the results of these functions as basis vectors in a way that the results of the best functions, i.e., the functions that give the maximum projections, are placed in the initial basis positions. To order the functions appropriately, several statistical measures, such as standard deviation and local correlation, are initially computed for the block itself and its neighborhood. These measures define block classes, e.g., uniform blocks in directional neighborhoods. The classes are then associated with the best functions to represent blocks with the specific characteristics described by the classes. The association between classes and best functions is defined by a previous extensive machine learning process employing several blocks from several images.

Like the basis pursuit coding scheme, the set of adaptive functions in the HSDT can be seen as a dictionary formed and ordered by a learning process. However, the results of the adaptive functions are the basis vectors, not the functions. Therefore, the basis is inherently different for each block. Unlike the basis pursuit scheme, all basis vectors are orthonormalized, resulting in a complete basis, not an overcomplete basis. Additionally, all coefficients resulting from the application of the basis to the block are scanned and entropy coded. So, there is no sparse coding.

This chapter will initially detail the HSDT characteristics in Section 5.2. Then, the adaptive functions will be described in Section 5.3 and the basis vectors ordering is presented in Section 5.4. A full image codec was built to evaluate the performance of the HSDT, which is compared with JPEG, JPEG 2000 and with the current state of the art scalable image codec using block transforms, JPEG XR. These comparisons are shown in Section 5.6.3. The performance of the HSDT in terms of energy concentration and transform coding gain is also computed and shown in Sections 5.6.2 and 5.6.1.

5.2 Hierarchical Block-based Signal Dependent Transform

In the multiresolution theory, a *scaling* function is used to create a series of approximations of a function, each differing by a factor of 2 in resolution from its nearest neighboring approximations. Additional functions, called *wavelets*, are then used to encode the difference in information between adjacent approximations. For two-dimensional functions, like images, one two-dimensional scaling function $\varphi(x, y)$ and three two-dimensional wavelets $\psi^H(x, y)$, $\psi^V(x, y)$, and $\psi^D(x, y)$ are required. These wavelet measure intensity variations along different directions: ψ^H measure variations along columns (e.g., horizontal edges), ψ^V measure variations along rows (e.g., vertical edges) and ψ^D corresponds to variations along diagonals. The scaling function and the three wavelets are usually arranged as the subbands in Fig. 5.2. As a practical example, the different intensity variations detected by each wavelet are clearly perceptible in Fig. 5.1.

Instead of being applied to the whole image, as wavelet transforms are traditionally designed, the hierarchical transform will be applied in this work in a block-based fashion with blocks of fixed size 8x8. In other words, a block transform will be considered as if it were an one-level wavelet decomposition, in the same way as the arrangement shown in Fig. 5.2. After the transform application, all 4x4 blocks resulting from the two-dimensional scaling function $\varphi(x, y)$ will be joined together to form the lower resolution image to be transformed in the next decomposition level. This procedure is graphically illustrated in the form of a pyramid in Fig. 5.3.

As can be seen in Fig. 5.3, the base of the pyramid contains the image with its full resolution and the apex contains the image with its lowest resolution (8x8). At each level increase, both size and resolution decrease by a factor of 2 in height and width. Although the resolution decreases at each level increase, the transform size is always 8x8 and the 4x4 blocks resulting from the scaling function are always grouped together to be decomposed at the next level. The application of the transform is shown in the right side of Fig. 5.3 for the first level and in the left side of Fig. 5.3 for

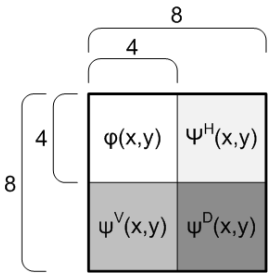


Figure 5.2: Subbands arrangement after one-level decomposition by a wavelet transform.

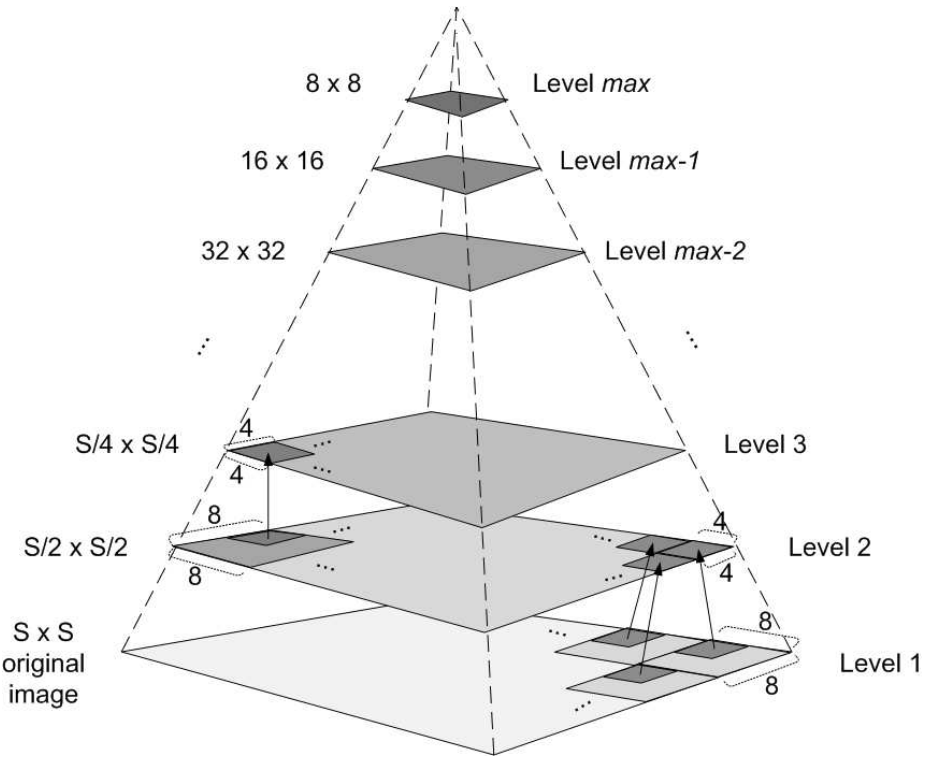


Figure 5.3: 8x8 transform application process arranged in the form of a level pyramid.

the second level. The same process is applied to all blocks in each level and to all higher levels until the maximum decomposition level is reached, where

$$\text{Level } max = log_2(S) - 2 \tag{5.1}$$

and S is the original square image side size.

The wavelet coefficients will, on average, be greater in the higher levels than in the lower ones, once each coefficient represents a larger spatial area of the image at each level increase. Therefore, the energy concentration increases with the level.

Once the transform size is fixed as 8x8, the full image represented with all its coefficients at any decomposition level is said to have dimensionality 64. The image represented with the same size of the full image, but only with the coefficients resulting from the two-dimensional 4x4 scaling function, from now on referred to as *approximation* coefficients, is said to have dimensionality 16. And the image represented only with the coefficients resulting from the wavelet functions, from now on referred to as *residual* coefficients, is said to have dimensionality 48. As an example, Fig. 5.4 shows all these images computed for the luminance plane of the *Palm leaf* picture

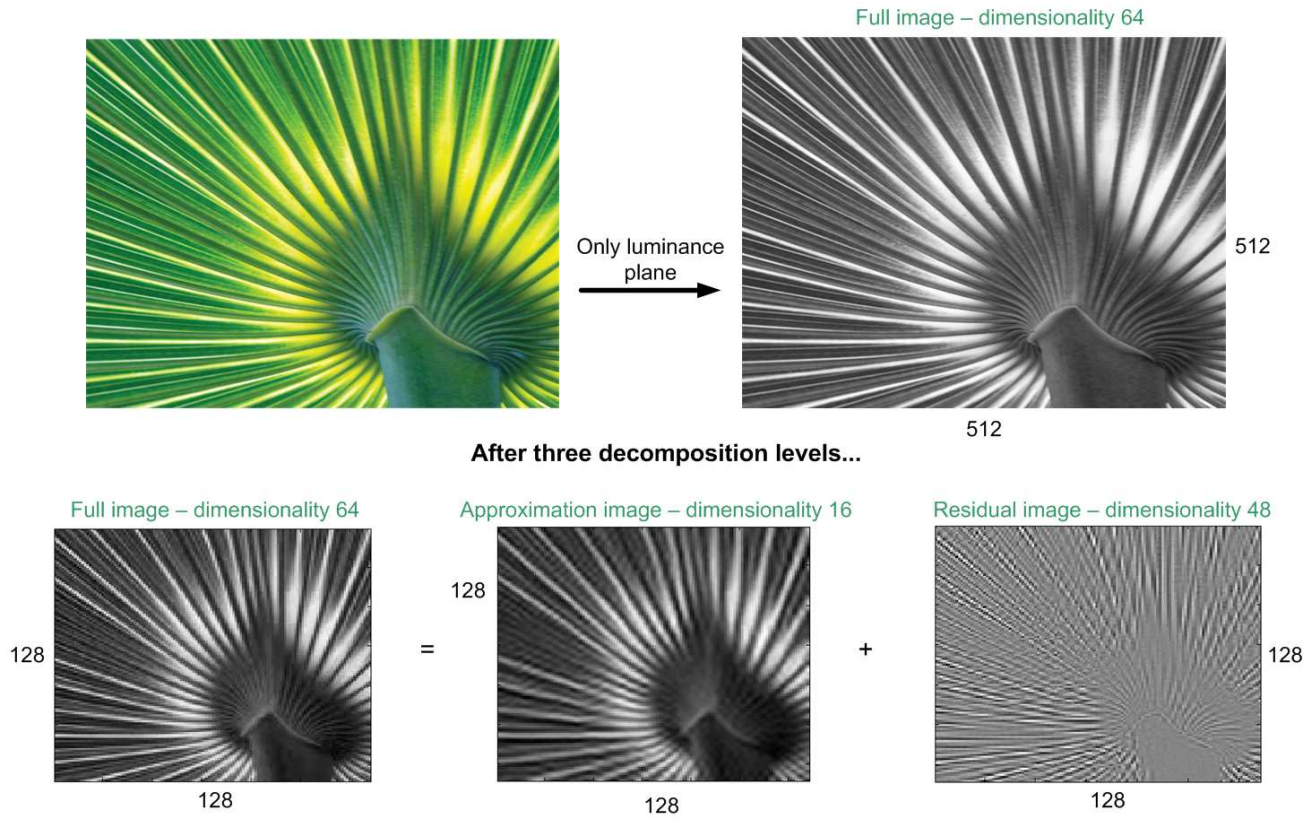


Figure 5.4: Full, approximation and residual images for the luminance plane of the *Palm leaf* picture after three decomposition levels.

after three decomposition levels. For display purposes, the image coefficients are shifted by 128 hereafter. One can see that, while the approximation image is the lower resolution version of the full image, the residual image carries all the missing details.

As mentioned in Section 5.1, the main purpose of the HSDT is to exploit the images cross-level structural similarities through an adaptive and signal dependent transform. The way this is performed by HSDT is detailed now with the help of the previously introduced concepts and the images in Fig. 5.5.

Initially, all decomposition levels are computed with a default hierarchical transform[1] applied in a block-based fashion with blocks of size 8x8. Then, the adaptation process of the HSDT starts at the highest decomposition level in a tentative to code the residual blocks down the pyramid structure with a more appropriate basis derived from the corresponding lower resolution block of the previous level. Fig. 5.5 illustrates this adaptation process when the residual blocks of the luminance plane of the *Palm leaf* picture are being coded with HSDT in level 3. The full image

[1]The default hierarchical transform employed by the HSDT is described at Section 5.2.1.

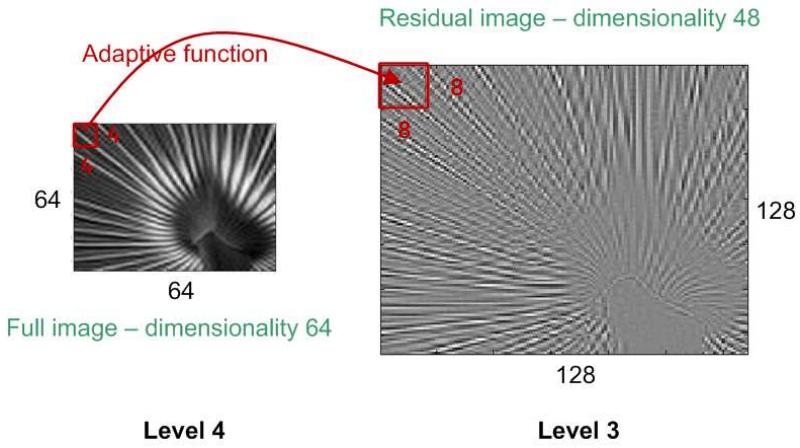


Figure 5.5: Adaptive function applied to a 4x4 full block of the lower resolution version of the luminance plane of the *Palm leaf* picture. The result should be as similar as possible to the 8x8 full block of the current level.

of the previous level, which is level 4 in this case, is available both at the encoding and decoding processes. A set of adaptive functions[2] is applied to each 4x4 full block of the lower resolution image. These functions should produce 8x8 full blocks as similar as possible to each 8x8 full block of the current level, which is level 3. If this similarity is successfully achieved, the representation of these produced 8x8 full blocks with dimensionality 48 will match with the residual block being coded. Therefore, an optimized basis for this residual block can be obtained by including these 8x8 produced blocks with dimensionality 48 as basis vectors in the transform. Consequently, most of the energy of the residual block will be concentrated in very few coefficients. This generation process of the HSDT for each residual block will be detailed in the following section through a linear algebra view. At this point, a few comments are worth to be mentioned:

1) The new adaptive basis is generated only for the residual blocks. Therefore, in the wavelet terminology, the wavelet functions are adaptive, while the scaling functions are never modified.

2) The results of the adaptive functions applied to the lower resolution blocks are used to code the residual blocks of the next resolution level. Therefore, the transform is signal dependent and exploits the cross-level dependencies.

3) The transform is not separable, i.e., it can not be applied independently to the columns and rows of the residual blocks. All basis vectors will be applied to the whole residual

[2]The adaptive functions employed by the HSDT are described at Section 5.3.

block. Therefore, the correlation and directionality properties of the blocks can be properly exploited.

5.2.1 ***A Linear Algebra View of Hierarchical Transform***

When a wavelet transform is applied, the resulting coefficients are usually arranged in the scheme shown in Fig. 5.2. Each coefficient in the approximation subband, for instance, is the result of the correlation of the whole image and a given two-dimensional scaling function. Therefore, there are 16 two-dimensional scaling functions to produce the 4x4 approximation subband in Fig. 5.2. The coefficients in the other subbands are also produced by the same procedure, but the correlation is calculated between the whole image and one two-dimensional wavelet function. Therefore, there are 48 two-dimensional wavelet functions to produce the three residual subbands, 16 functions for each subband. If each function is read by columns in order to form a function vector of size 1x64, all functions can be arranged as a transform matrix as shown in Fig. 5.6. This figure illustrates the 64x64 matrix form of a wavelet transform of size 8x8.

As previously described, before the adaptation process of the HSDT, all decomposition levels are computed with a default hierarchical transform applied in a block-based fashion with blocks of size 8x8. In this work, the default hierarchical transform is chosen as the multilevel DCT. Both the basis patterns and the matrix form of the two-dimensional 8x8 DCT are shown in Fig. 5.7. As described for the wavelet transform, each function of the DCT is also read by columns in order to form a function vector of size 1x64. This function vector is then positioned at the transform matrix according to the following rules:

1) The two-dimensional DCT is divided into subbands in a wavelet-fashion. The blue lines in Fig. 5.7 represent this division. The top-left 4x4 basis patterns are considered the scaling functions and the coefficients resulting from the application of these functions are the approximation coefficients. The function vectors of the scaling functions are positioned at the first 16 rows of the transform matrix.

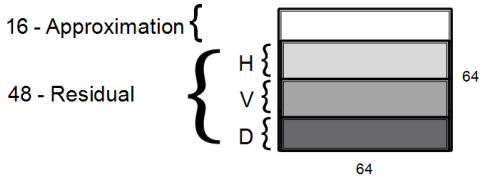
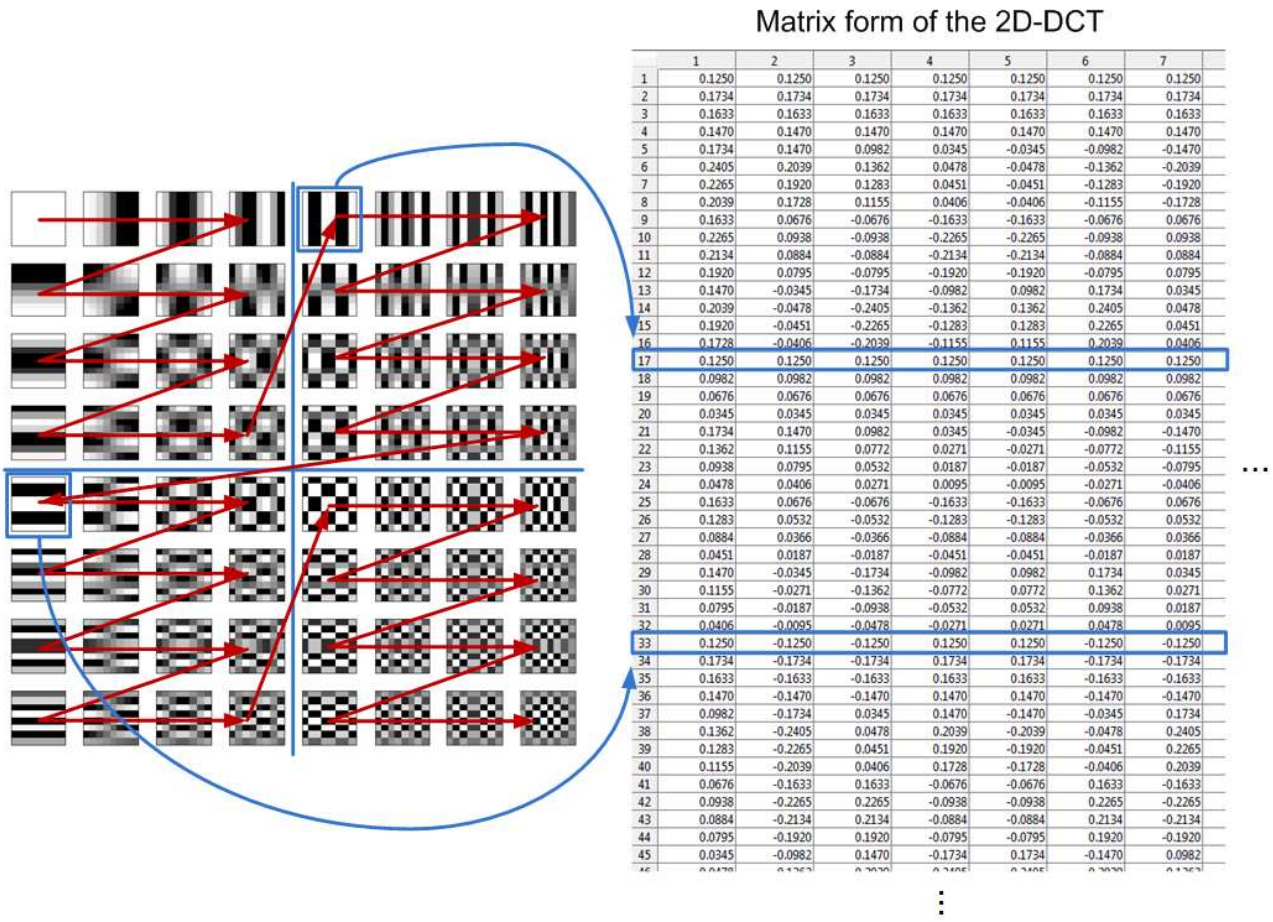


Figure 5.6: Matrix form of a wavelet transform of size 8x8.



	1	2	3	4	5	6	7
1	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.1734	0.1734	0.1734	0.1734	0.1734	0.1734	0.1734
3	0.1633	0.1633	0.1633	0.1633	0.1633	0.1633	0.1633
4	0.1470	0.1470	0.1470	0.1470	0.1470	0.1470	0.1470
5	0.1734	0.1470	0.0982	0.0345	-0.0345	-0.0982	-0.1470
6	0.2405	0.2039	0.1362	0.0478	-0.0478	-0.1362	-0.2039
7	0.2265	0.1920	0.1283	0.0451	-0.0451	-0.1283	-0.1920
8	0.2039	0.1728	0.1155	0.0406	-0.0406	-0.1155	-0.1728
9	0.1633	0.0676	-0.0676	-0.1633	-0.1633	-0.0676	0.0676
10	0.2265	0.0938	-0.0938	-0.2265	-0.2265	-0.0938	0.0938
11	0.2134	0.0884	-0.0884	-0.2134	-0.2134	-0.0884	0.0884
12	0.1920	0.0795	-0.0795	-0.1920	-0.1920	-0.0795	0.0795
13	0.1470	-0.0345	-0.1734	-0.0982	0.0982	0.1734	0.0345
14	0.2039	-0.0478	-0.2405	-0.1362	0.1362	0.2405	0.0478
15	0.1920	-0.0451	-0.2265	-0.1283	0.1283	0.2265	0.0451
16	0.1728	-0.0406	-0.2039	-0.1155	0.1155	0.2039	0.0406
17	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
18	0.0982	0.0982	0.0982	0.0982	0.0982	0.0982	0.0982
19	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676
20	0.0345	0.0345	0.0345	0.0345	0.0345	0.0345	0.0345
21	0.1734	0.1470	0.0982	0.0345	-0.0345	-0.0982	-0.1470
22	0.1362	0.1155	0.0772	0.0271	-0.0271	-0.0772	-0.1155
23	0.0938	0.0795	0.0532	0.0187	-0.0187	-0.0532	-0.0795
24	0.0478	0.0406	0.0271	0.0095	-0.0095	-0.0271	-0.0406
25	0.1633	0.0676	-0.0676	-0.1633	-0.1633	-0.0676	0.0676
26	0.1283	0.0532	-0.0532	-0.1283	-0.1283	-0.0532	0.0532
27	0.0884	0.0366	-0.0366	-0.0884	-0.0884	-0.0366	0.0366
28	0.0451	0.0187	-0.0187	-0.0451	-0.0451	-0.0187	0.0187
29	0.1470	-0.0345	-0.1734	-0.0982	0.0982	0.1734	0.0345
30	0.1155	-0.0271	-0.1362	-0.0772	0.0772	0.1362	0.0271
31	0.0795	-0.0187	-0.0938	-0.0532	0.0532	0.0938	0.0187
32	0.0406	-0.0095	-0.0478	-0.0271	0.0271	0.0478	0.0095
33	0.1250	-0.1250	-0.1250	0.1250	0.1250	-0.1250	-0.1250
34	0.1734	-0.1734	-0.1734	0.1734	0.1734	-0.1734	-0.1734
35	0.1633	-0.1633	-0.1633	0.1633	0.1633	-0.1633	-0.1633
36	0.1470	-0.1470	-0.1470	0.1470	0.1470	-0.1470	-0.1470
37	0.0982	-0.1734	0.0345	0.1470	-0.1470	-0.0345	0.1734
38	0.1362	-0.2405	0.0478	0.2039	-0.2039	-0.0478	0.2405
39	0.1283	-0.2265	0.0451	0.1920	-0.1920	-0.0451	0.2265
40	0.1155	-0.2039	0.0406	0.1728	-0.1728	-0.0406	0.2039
41	0.0676	-0.1633	0.1633	-0.0676	-0.0676	0.1633	-0.1633
42	0.0938	-0.2265	0.2265	-0.0938	-0.0938	0.2265	-0.2265
43	0.0884	-0.2134	0.2134	-0.0884	-0.0884	0.2134	-0.2134
44	0.0795	-0.1920	0.1920	-0.0795	-0.0795	0.1920	-0.1920
45	0.0345	-0.0982	0.1470	-0.1734	0.1734	-0.1470	0.0982

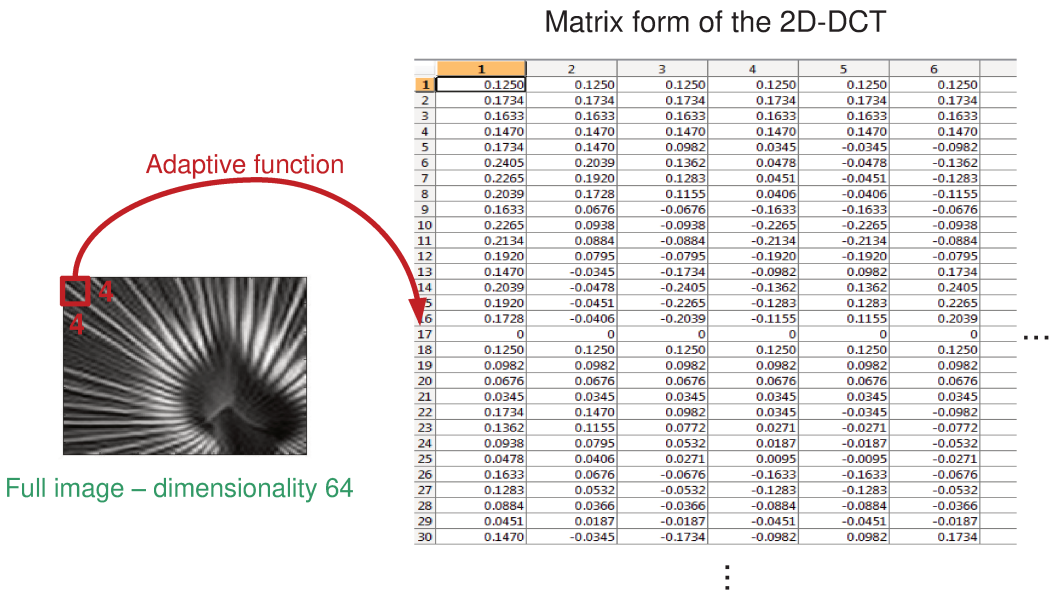
Figure 5.7: 8x8 DCT basis patterns and matrix form.

2) All other basis patterns are considered as the wavelet functions and the resulting coefficients are the residual coefficients. The position of the function vectors of the wavelet functions in the transform matrix follows the scheme shown in Fig. 5.6.

3) Inside each subband, the basis patterns are read according to the raster order illustrated by the red arrows in Fig. 5.7.

It is important to note here that the transform matrix of the two-dimensional DCT is not frequency-ordered. As highlighted by the blue arrows in Fig. 5.7, basis with low frequency patterns are positioned far from each other, for example, in the 17^{th} and 33^{rd} matrix positions. This point will be important for some considerations in Section 5.3.

As the HSDT is not separable and all basis vectors are applied to the whole residual block, the HSDT also must be arranged in its matrix form. As previously emphasized, the adaptive basis in the HSDT is generated only to code the residual blocks, which means that, the first 16 rows



	1	2	3	4	5	6
1	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
2	0.1734	0.1734	0.1734	0.1734	0.1734	0.1734
3	0.1633	0.1633	0.1633	0.1633	0.1633	0.1633
4	0.1470	0.1470	0.1470	0.1470	0.1470	0.1470
5	0.1734	0.1470	0.0982	0.0345	-0.0345	-0.0982
6	0.2405	0.2039	0.1362	0.0478	-0.0478	-0.1362
7	0.2265	0.1920	0.1283	0.0451	-0.0451	-0.1283
8	0.2039	0.1728	0.1155	0.0406	-0.0406	-0.1155
9	0.1633	0.0676	-0.0676	-0.1633	-0.1633	-0.0676
10	0.2265	0.0938	-0.0938	-0.2265	-0.2265	-0.0938
11	0.2134	0.0884	-0.0884	-0.2134	-0.2134	-0.0884
12	0.1920	0.0795	-0.0795	-0.1920	-0.1920	-0.0795
13	0.1470	-0.0345	-0.1734	-0.0982	0.0982	0.1734
14	0.2039	-0.0478	-0.2405	-0.1362	0.1362	0.2405
15	0.1920	-0.0451	-0.2265	-0.1283	0.1283	0.2265
16	0.1728	-0.0406	-0.2039	-0.1155	0.1155	0.2039
17	0	0	0	0	0	0
18	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
19	0.0982	0.0982	0.0982	0.0982	0.0982	0.0982
20	0.0676	0.0676	0.0676	0.0676	0.0676	0.0676
21	0.0345	0.0345	0.0345	0.0345	0.0345	0.0345
22	0.1734	0.1470	0.0982	0.0345	-0.0345	-0.0982
23	0.1362	0.1155	0.0772	0.0271	-0.0271	-0.0772
24	0.0938	0.0795	0.0532	0.0187	-0.0187	-0.0532
25	0.0478	0.0406	0.0271	0.0095	-0.0095	-0.0271
26	0.1633	0.0676	-0.0676	-0.1633	-0.1633	-0.0676
27	0.1283	0.0532	-0.0532	-0.1283	-0.1283	-0.0532
28	0.0884	0.0366	-0.0366	-0.0884	-0.0884	-0.0366
29	0.0451	0.0187	-0.0187	-0.0451	-0.0451	-0.0187
30	0.1470	-0.0345	-0.1734	-0.0982	0.0982	0.1734

Figure 5.8: Result of an adaptive function applied to a 4x4 full block included as a basis vector in the transform matrix.

of the transform matrix are never modified, while the other 48 rows are adapted. As the default hierarchical transform chosen for the HSDT is the multilevel DCT, the first 16 rows of the HSDT transform matrix are exactly the scaling functions of the multilevel DCT. The adaptation process of the other 48 rows is illustrated by Fig. 5.8. In this figure, a basis vector, resulting from the application of one adaptive function to a 4x4 full block of the lower resolution image, is included at the 17^{th} matrix position. One can see that the matrix is shifted down by one row. Therefore the new basis vector is indeed included in the matrix and does not replace the original rows from the multilevel DCT. Not only one, but a set of adaptive functions is applied to the 4x4 full block. The basis vectors resulting from the application of all these adaptive functions are then included sequentially in the 18^{th}, 19^{th}, 20^{th}, etc, matrix positions.

As previously described and shown in Fig. 5.5, the set of adaptive functions applied to the 4x4 full blocks of the lower resolution image produce 8x8 full blocks. Therefore, the basis vector included in the transform matrix is the 8x8 full block (with dimensionality 64) read by columns. However, as this basis vector will code the residual block, it must be represented with dimensionality 48. This representation will be achieved through a Gram-Schmidt orthonormalization with respect to the first 16 rows of the matrix.

The first 16 rows of the matrix are already linear independent, since they are always the original rows from the multilevel DCT. However, the basis vectors resulting from the adaptive

functions are linear dependent. Therefore, each new basis vector included in the matrix must be orthonormalized with respect to the first 16 rows and to the other basis vectors previously included into the matrix.

After the orthonormalization of all included basis vectors, the residual subbands rows from the original multilevel DCT that were shifted and kept in the matrix are also orthonormalized with respect to the included basis vectors. This orthonormalization process is performed row-by-row until the basis is complete. Consequently, at the end of the HSDT matrix generation process, a complete orthonormal optimized basis is obtained.

It is important to note that the same basis can be generated both at the encoding and decoding processes, once the adaptation depends only on the lower resolution image computed at the previous decomposition level.

In the linear algebra view of the HSDT application, the residual block is initially read by columns in order to form a block vector. Then, the inner product of this block vector and the basis vectors is calculated to obtain the resulting coefficients. The approximation coefficients (first 16) will be null, once the residual block has dimensionality 48, and the first residual coefficient (17^{th} coefficient) will concentrate most of the residual block energy, if the adaptive functions are successfully designed.

5.3 Adaptive Functions

The performance of the HSDT is highly dependent on the performance of the adaptive functions. If the adaptive functions can produce basis vectors that, after the orthonormalization process, are similar to the residual block, the transform will achieve its goal of high energy concentration. In order to pursue this similarity, two natural image properties will be exploited by the adaptive functions as described in the following sections: directionality and correlation.

5.3.1 ***Directional Adaptive Functions***

Separable two-dimensional transforms applied horizontally and vertically capture the information in these directions very well. However, in natural images, most of the information is not along these two directions. Besides, the Karhunen-Loève Transform (KLT) is optimal given the assumptions on the signal correlation matrix that is characterized by isotropically and strongly correlated pixels [70]. But, for most images, the correlation changes locally and is more likely stronger along a certain direction, not necessarily being horizontal or vertical. Based on these

considerations, several directional transforms have been proposed [71–76]. These transforms can be classified into two categories. In the first category, new transforms are introduced to incorporate directional bases, like curvelets [71] and dual tree complex wavelets [72]. The transforms of the second category are modified from existing transforms, such as the directional wavelet transform [73], the directional DCT transform [74, 75] and the directional lapped transform [76].

The scheme proposed to design directional adaptive functions in this work is from the first category and consists of two steps. All adaptive functions are applied to the 4x4 full blocks of the lower resolution image and must produce 8x8 full blocks. Therefore, in the first step, the dominant direction of the 4x4 block is identified, provided that the block has a strong correlation along a certain direction. This step is referred to as *direction detection*. In the second step, the block is interpolated in the dominant direction to generate the 8x8 block. This step is, then, referred to as *directional interpolation*.

The combination of direction detection and directional interpolation techniques is suitable to produce directional adaptive functions for the HSDT based on the assumption that lower resolution blocks with strong directional patterns show the same strong directional patterns in the corresponding residual block. As all blocks of the full lower resolution image of the previous decomposition level are available both at coding and decoding processes of the current decomposition level, the 4x4 block is expanded to a 8x8 block in order to reduce the blocking artifacts. Both the direction detection and directional interpolation procedures are then applied to the expanded 8x8 blocks and 16x16 interpolated blocks are produced. The inner central 8x8 area of this interpolated block is finally read by columns to form the basis vector. In other words, the 8x8 central area is the result of the directional adaptive function that will be included in the transform matrix as a basis vector. An example of the results obtained with these techniques is shown in Fig. 5.9 for a 4x4 full block of the luminance plane of the *Palm leaf* picture.

Several direction detection and directional interpolation techniques are employed by HSDT and will be detailed in the sequel. Each combination of the techniques forms a directional adaptive function, resulting in a set of 30 different functions.

Direction Detection

The techniques employed by HSDT for direction detection can be separated into three categories: Hough transform, texture descriptors and orthogonal regression using principal components analysis.

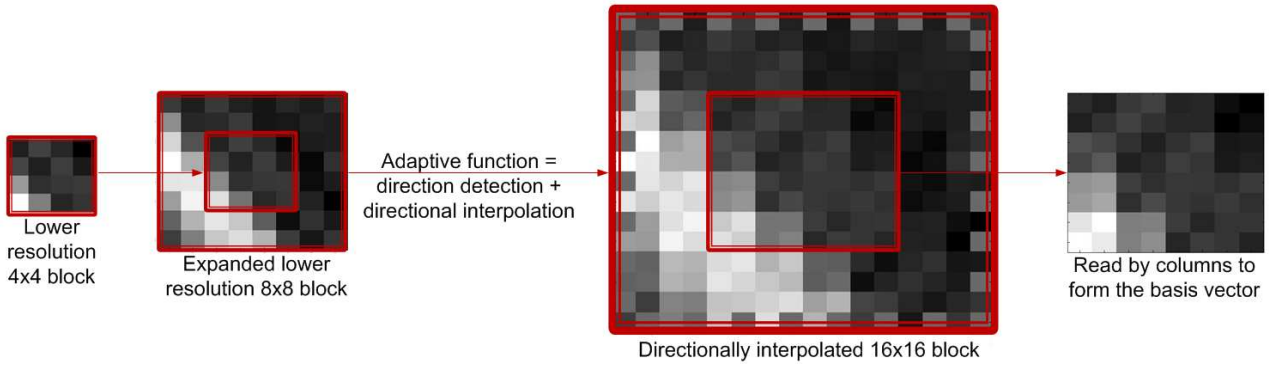


Figure 5.9: Application of a directional adaptive function to an expanded 8x8 lower resolution block in order to produce a basis vector.

The Hough transform using the principle of template matching [77] is applied to detect lines in the lower resolution 8x8 blocks. The result of the transform is the projection of the block intensity along a radial line oriented at a specific angle. If the transform is computed for angles varying from 0° to 179°, the result is a matrix which components are the projected values. The maximum value gives the block dominant direction. Previously to the transform, an edge detector algorithm can also be applied to the block, such as the Prewitt operator. The image resulting from the edge detection process is then the Hough transform input.

Texture descriptors characterize image textures or regions. The descriptors aim to describe a texture through the computation of several statistical measures, many of them related with directional features. Therefore, two texture descriptors that compute spatial activity in specific directions are employed by HSDT: LAS and SASI.

The Local Activity Spectrum (LAS) descriptor [78] computes four simple measures for each pixel of a 3x3 block and detects the block spatial activity in only four directions: 0°, 45°, 90° and 135°. In order to be applied to the expanded 8x8 blocks, the following variations of the original LAS are developed in this work:

1) The 3x3 mask block is shifted pixel-by-pixel inside the 8x8 block and the 4 original LAS measures are computed per shift. The measure with the smallest value per shift gives the dominant direction for each 3x3 area. After all 8x8 block is covered, the most frequent dominant direction is considered the final dominant direction of the 8x8 block. This version of the LAS descriptor will be referred to in this work as *LAS - 4 Directions*.

2) The 3x3 mask block is expanded to a 5x5 mask block and each measure is adapted to enclose more pixels from the block. Therefore, 12 directions can now be detected: from 0° to 165°, with intervals of 15°. Additionally, less shifts are required to cover the whole 8x8 block. The decision procedure of choosing the most frequent local dominant direction as the final dominant direction is kept. This version will be referred to as *LAS - 12 Directions*.

3) The 5x5 mask block is finally expanded to a 7x7 mask block, the measures enclosure is also expanded and 24 different directions can then be detected: from 0° to 172.5°, with intervals of 7.5°. This version will be referred to as *LAS - 24 Directions*.

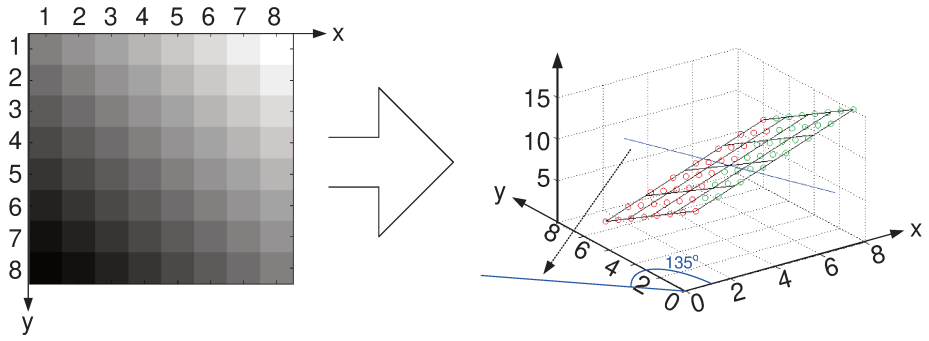
The Statistical Analysis of Structural Information (SASI) [79, 80] descriptor is based on second order statistics of clique autocorrelation coefficients, which are the coefficients over a set of directional moving windows. The moving windows are designed with various sizes and shapes to describe the characteristics of textures in different granularity. However, the directions are limited to only four possibilities: 0°, 45°, 90° and 135°. In this work, windows of size 3x3 are used and positioned in the expanded 8x8 blocks in the following ways:

1) The window is centralized in the 8x8 block and the autocorrelation coefficients are computed for each of the four directions. The largest autocorrelation coefficient gives the dominant direction of the 8x8 block. This version of the SASI descriptor will be referred in this work as *SASI - 1 value*.

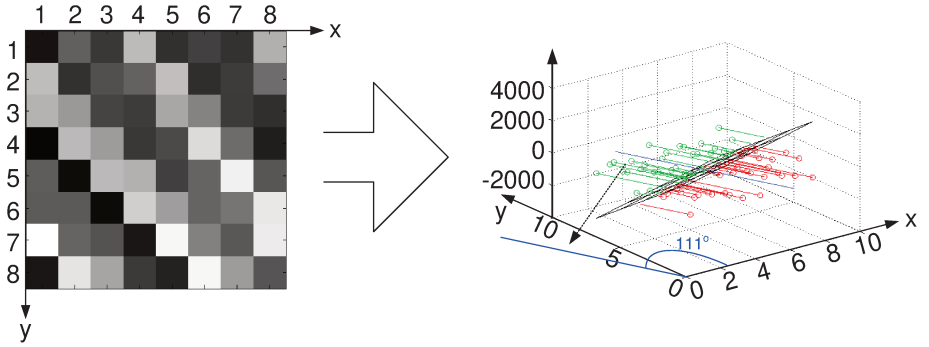
2) The window is shifted over four central positions in the 8x8 block and the autocorrelation coefficients are computed for each direction and shift. The means of the autocorrelation coefficients of the same direction are then computed and the largest mean gives the dominant direction of the 8x8 block. This version of the SASI descriptor will be referred to as *SASI - 4 values*.

3) The same procedure in item *2)* is performed, but the window is shifted over eight positions in the 8x8 block. This version will be referred to as *SASI - 8 values*.

Finally, the last technique employed by the HSDT for direction detection is the orthogonal regression using Principal Components Analysis (PCA). The PCA designates a class of methods for signal analysis and is particularly used here to fit a linear regression that minimizes the perpendicular distances from the values in the expanded 8x8 block to the fitted plane. This linear regression process is known as orthogonal regression or total least squares and the fitted plane



(a)



(b)

Figure 5.10: Best-fit regression planes computed to find the dominant direction of: *a*) an artificially generated 8x8 block and *b*) a real 8x8 block.

is the best 2-D linear approximation to the 8x8 block. The normal vector of the fitted plane is projected onto the x-y plane and shifted to the origin. The angle that the projection makes with x-axis gives the dominant direction of the 8x8 block. This direction detection technique will be referred to in this work as *Fit Plane*. A variant of the technique is to consider the perpendicular direction of the dominant direction as the new dominant direction. This variant will be referred to as as *Fit Plane - Perpendicular*.

Fig. 5.10 illustrates the results obtained with the *Fit Plane* technique. A 8x8 block generated with a well-defined directionality of 135° is shown in Fig. 5.10(a) with its corresponding best-fit regression plane. The blue line is the normal vector of the plane. The dotted arrow indicates the projection of the normal vector onto the x-y plane and the shift to the origin. As expected, the projection makes exactly 135° with x-axis. A real example, with similar visual directionality of 135°, is shown in Fig. 5.10(b). The green points around the fitted plane are above it and the red points are below. For this real 8x8 block, the technique returns a good approximation of 111° as the dominant block direction.

Directional Interpolation

The main task of all adaptive functions is to produce 8x8 blocks from the 4x4 blocks. Therefore, the missing block values must be generated from the available information. This process is often referred to as magnification. The task of magnification is an essential part of software zooming, for instance, and the main challenge is to preserve sharpness of image after resolution enhancement. The traditional magnification approaches are based on bilinear, bicubic or spline interpolation. These methods are fast, but usually generate blurred images. To avoid this problem, interpolation methods that try to preserve the image characteristics, such as the directionality, can be applied [81–83].

Three simple directional interpolation methods were specially designed for the HSDT: interpolation with directional weights enclosing a 12-point neighborhood, interpolation with directional weights enclosing a 16-point neighborhood and interpolation with directional and Euclidean weights enclosing a 16-point neighborhood. The direction of the dominant block is determined by one of the direction detection techniques previously described.

An illustrative example of the directional weight computation enclosing a small 4-point neighborhood is given by Fig. 5.11. In this example, the four black points are the original available points and the central red point is the point being interpolated. The directional interpolation should be performed at $\theta = 30°$. The dashed line passing through the red point, i.e., the origin, sets this direction. The perpendicular distances d_i of all original points P_i to the dashed line are computed and the inverse distances are used as weights ω_i. Therefore, the weight assigned to each neighbor point diminishes as the distance from the dashed line increases. The procedure can be defined as an inverse directional distance weighted interpolation.

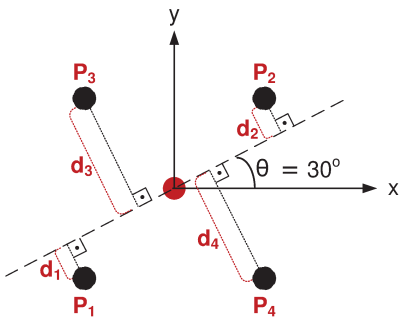


Figure 5.11: Directional distance computation enclosing 4-original points represented in black. The central red point is the point to be directionally interpolated at $\theta = 30°$.

Considering that the perpendicular distance between a point $P(x, y)$ and a line $ax+by+c = 0$ is given by

$$d = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}, \tag{5.2}$$

the perpendicular distances d_i between the points $P_i(x_i, y_i)$ in Fig. 5.11 and the dashed line oriented θ° from the x-axis can be expressed as

$$d_i = \frac{|tg\theta \cdot x_i - y_i|}{\sqrt{tg^2\theta + 1}}. \tag{5.3}$$

The weights are the inverse distances, i.e. $\omega_i = 1/d_i$, and must be normalized. Each weight is then divided by the sum of all weights.

After the directional weight computation is completed, each directionally interpolated point P can be computed as

$$P = \sum_{i=1}^{|\mathbb{P}|} \omega_i \cdot P_i \tag{5.4}$$

where $\mathbb{P}$ is the set of original points included in the interpolated point neighborhood.

In this work, neighborhoods of both 12 and 16 original points are considered and the arrangements are represented, respectively, in Figs. 5.12(a) and 5.12(b). The same directional weight computation procedure for the 4-point neighborhood is extended for enclosing a 12-point neighborhood and a 16-point neighborhood. All directional weights are pre-computed and stored to be used on-the-fly in the computation of the interpolated points.

In both arrangements of Fig. 5.12, the black points are the original available points and the red points are the points to be directionally interpolated. The arrangement in Fig. 5.12(a) has staggered original rows and columns. Therefore, every point to be interpolated has equidistant original neighbors, i.e., it is adjacent to four original points. The directional interpolation technique previously referred to as *interpolation with directional weights enclosing a 12-point neighborhood* employs this type of arrangement. The central point in Fig. 5.12(a), for instance, is directionally interpolated through Eq. (5.4), where $\mathbb{P}$ comprises all 12 original points in the arrangement.

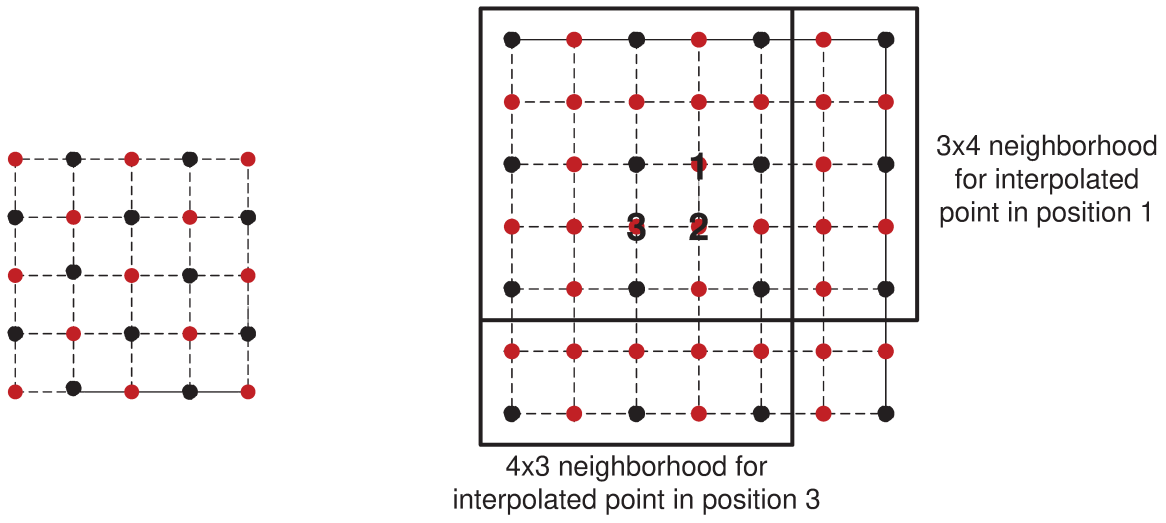


Figure 5.12: Arrangements for interpolation with directional weights enclosing a: *a*) 12-point neighborhood and *b*) 16-point neighborhood.

Unlike the arrangement in Fig. 5.12(a), the arrangement in Fig. 5.12(b) has a regular pattern where points at rows and columns positions multiple to pel (0,2,4,..., for pel=2) must be original points and the points half-way between the original points must be directionally interpolated. These points are usually known as "half-pel points". Each half-pel point that is adjacent to two original points, such as the points in positions **1** and **3** in Fig. 5.12(b), is interpolated through Eq. (5.4), where $\mathbb{P}$ comprises all 12 original points in the highlighted neighborhoods: 3x4 neighborhood for interpolated points in position **1** and 4x3 neighborhood for interpolated points in position **3**. For the half-pel points in position **2**, $\mathbb{P}$ comprises all 16 original points in the arrangement. This is the directional interpolation technique previously referred to as *interpolation with directional weights enclosing a 16-point neighborhood.*

In some cases, the computation of the directional weights can not assure that the value of the directionally interpolated point will be properly influenced by each of its neighbors. Fig. 5.13 shows the case of a point being interpolated in position **1** of the arrangement shown in Fig. 5.12(b). The directional interpolation should be performed at $\theta = 90°$. As can be noted, all perpendicular distances from the original points 1 to 6 to the dashed line oriented 90° from the x-axis are equal, which means that all these points are weighted equally in Eq. (5.4). However, the closest points, 1 and 2, should have more influence on the final value of the point being interpolated than all other

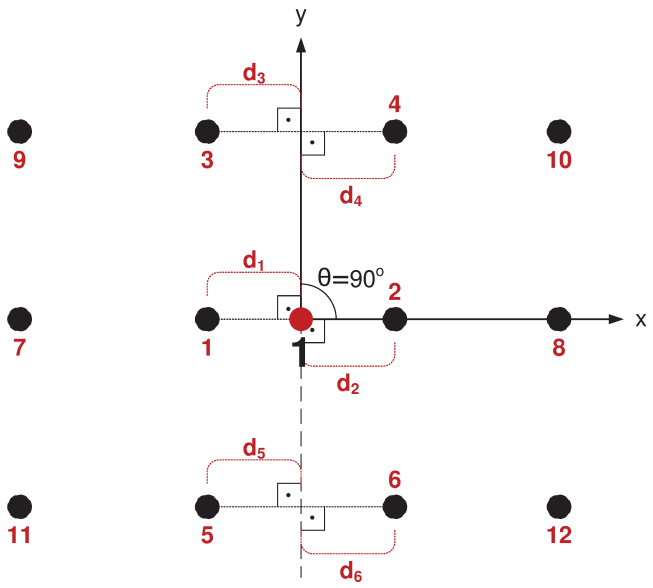


Figure 5.13: Red central point being directionally interpolated at $\theta = 90°$ in the position **1** of the arrangement shown in Fig. 5.12(b).

points. In order to solve this problem, Euclidean distances between all original points and the point being interpolated are also pre-computed, stored and turned into normalized weights. These Euclidean weights are then combined with the directional weights in the directional interpolation technique previously referred to as *interpolation with directional and Euclidean weights enclosing a 16-point neighborhood*.

5.3.2 ***Block Matching Adaptive Functions***

While the directional adaptive functions exploit the natural blocks directionality, the block matching adaptive functions exploit the natural local and non-local blocks correlations. It is well recognized that pixels in images usually have a strong correlation to their neighboring pixels, known as local correlation. In addition, across an image, there are potentially similar structures and repeated patterns, exhibiting strong non-local correlations.

The scheme proposed to design block matching adaptive functions in this work takes advantage of the correlation among the 4x4 full blocks of the lower resolution image to obtain the basis vector to be included in the transform matrix. This scheme is illustrated by Fig. 5.14. The block marked with X in the 128x128 image is the current block being encoded/decoded. The approximation block, i.e., the representation of the block with dimensionality 16, is available from the 64x64 final image of the previous decomposition level. Therefore, encoding/decoding of the current

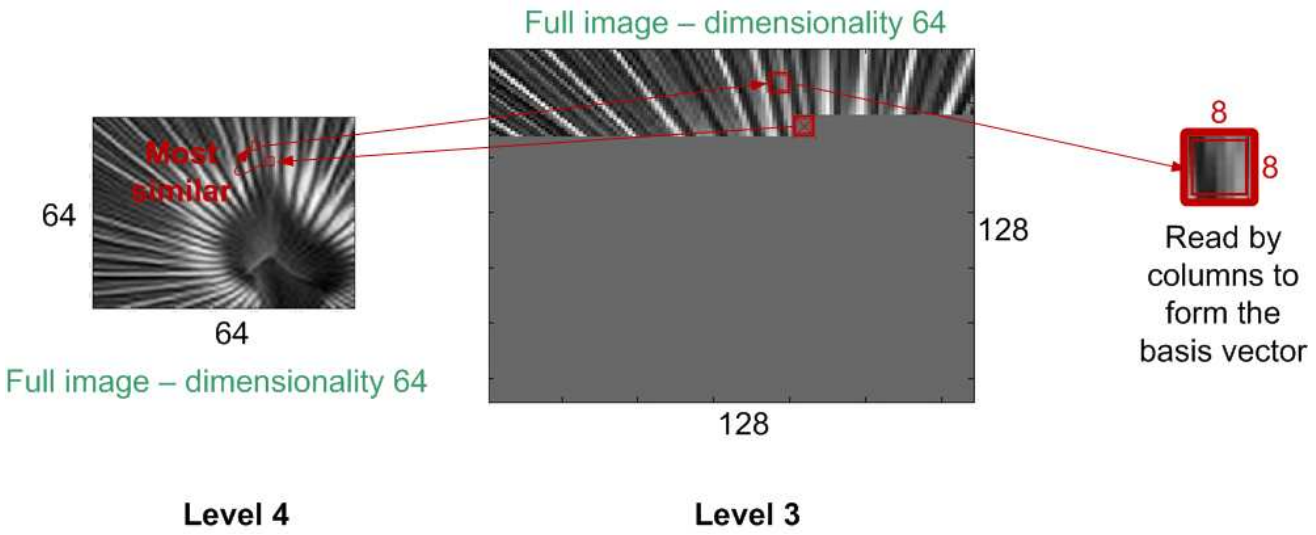


Figure 5.14: Application of a block matching adaptive function to a 4x4 lower resolution block in order to produce a basis vector.

block actually means encoding/decoding of the residual block in order to complete the full block representation and the transform will be adaptively optimized to code this residual block. Once the blocks encoding/decoding processes are performed in raster order, all blocks in previous rows and previous columns of the current row are available with dimensionality 64.

As indicated by Fig. 5.14, the block matching algorithm is applied only to the full blocks of the lower resolution image. Initially, the corresponding lower resolution block of the current block being encoded/decoded is localized. Then, the most similar block to the lower resolution block is identified. Finally, the corresponding full 8x8 block of this most similar block is localized and read by columns to form the basis vector. It is important to note that, unlike performed by the directional adaptive functions, the blocks read as columns to form the basis vectors are not produced by directional interpolation. These blocks are directly extracted from the already encoded/decoded blocks of the current full image.

The block matching algorithm applied to the full blocks of the lower resolution image always employs a small window search of 3x5 blocks, i.e., 12x20 values. As the matching computations take considerable computation time, the window search size is kept small. In another effort to reduce the computation time, the matching computations can be performed in two stages as shown in Fig. 5.15. In the first stage, referred to as *5-point comparison*, only the central 4x4 areas of connected blocks, surrounded by the red boxes, are compared with the current block, marked with X. The best match between the current block and the 5 areas is obtained and the

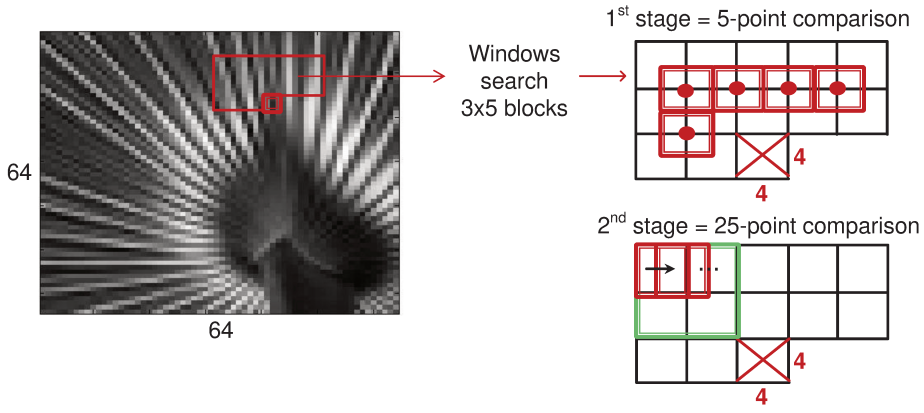


Figure 5.15: 2-stage block matching process.

second stage is performed according to this result. Only the best match area is expanded to a 8x8 area and new comparisons are performed pixel-by-pixel inside this 8x8 area in a full search procedure. The second stage is then referred to as *25-point comparison*. In the example shown in Fig. 5.15, the top-left 4x4 area is the "winner" in the first stage and the green box in the second stage represents the expanded 8x8 area.

The simplest distance metric employed to evaluate the similarity among the blocks, in order to identify the best match block, is the Sum of Absolute Differences (SAD). The computation of the block matching with the 2-stage process and SAD as the distance metric will be referred in this work as *Two-step*. If the block matching is computed with a pixel-by-pixel full search procedure in the whole 3x5 window search and SAD is employed as the distance metric, the procedure will be referred to as *One-step*.

It is assumed that the distance metric labels the best match block. However, this assumption does not always hold and blocks with higher similarity than the chosen as the best match block are ignored. Therefore, another version of the *One-step* procedure considers not only one best match block, but the 5 best match blocks. The 5 corresponding 8x8 blocks are localized at the current full image, the mean of these 5 blocks is computed and read by columns to form the basis vector. This version of the *One-step* procedure is referred to as *One-step mean*.

Like designed for the directional adaptive functions, the block matching adaptive functions also employ texture descriptors. The difference is that, unlike the directional functions, here all the values computed for each descriptor, not only the largest autocorrelation coefficient, are taken into account. The descriptors are composed of a set of values and several distance metrics are

employed in order to measure the similarity among the descriptors computed for each block. The set of values in the descriptors can be normalized or non-normalized. Another difference is that not only descriptors that detect directional features in the blocks are computed, such as LAS and SASI described in Section 5.3.1. Texture descriptors that describe the neighborhood of each value inside the block are also employed, such as the LBP. The Local Binary Pattern (LBP) [84] descriptor is defined as a gray-scale invariant texture measure which labels the pixels of an image by thresholding the neighborhood of each pixel and considering the result as a binary number.

Several combinations of techniques can be made for the block matching adaptive functions employing texture descriptors. For the functions employing SASI, for instance, the versions in Section 5.3.1 referred to as *SASI - 1 value*, *SASI - 4 values* and *SASI - 8 values* are all employed here with normalized or non-normalized descriptors and with Euclidean distances or the similarity measure defined in [79] as distance metrics. Considering all possible combinations for the functions employing SASI and all other functions described in this Section, a total of 18 different blocking matching adaptive functions arises.

5.3.3 ***Performance Evaluation***

As 30 different directional adaptive functions resulted from the combination of the techniques detailed in Section 5.3.1 and more 18 different block matching adaptive functions resulted from the methods described in Section 5.3.2, a diverse set of 48 adaptive functions is achieved.

In an effort to evaluate the average performance of these adaptive functions, the following procedure is adopted:

1) Several images are selected from a large database of perceptually diverse content [66].

2) All adaptive functions are computed to all residual blocks from the several selected images.

3) The result of each adaptive function is included in the 17^{th} transform matrix position, as illustrated by Fig. 5.8. The transform is orthonormalized and used to encode the residual block.

4) After the residual block is encoded with the transform produced by each adaptive function, the energy percentage represented only by the first residual coefficient, i.e., the 17^{th} coefficient, is computed with respect to the total residual block energy.

5) The average energy percentage per adaptive function is computed over all encoded blocks from all selected images.

Considering the size of the window search (12x20 values) employed by the block matching adaptive functions and also the fact that after a certain decomposition level, the coefficient values are too randomized for being exploited with correlation and directionality techniques, the HSDT is only computed for decomposition levels where images have sizes greater or equal than 64x64. It means, for instance, that for all images with size 512x512 shown in the results of Section 5.6, the HSDT will be computed for only 4 decomposition levels: levels 4, 3, 2, and 1. It is worthy to refer here to Fig. 5.3 for an illustration of the levels arrangement.

As the HSDT is not computed for all decomposition levels, the blocks in the levels where the transform is not computed are encoded/decoded with the default multilevel DCT. However, as mentioned in Section 5.2.1 and shown in Fig. 5.7, the transform matrix of the 2D-DCT is not frequency-ordered. In an effort to achieve a better order for the 2D-DCT, the performance of each "residual" 2D-DCT is evaluated. This evaluation is performed in this work in the same way as the adaptive functions were evaluated by the 1-5 steps procedure previously described.

Fig. 5.16 shows some performance evaluations for the 48 adaptive functions and the 48 residual DCT functions. The evaluations were performed with several QPs (quantization parameters)[3] and the results obtained with QP 20 and QP 60 are illustrated, respectively, by the left and right graphs. Several conclusions can be taken from these graphs, such as:

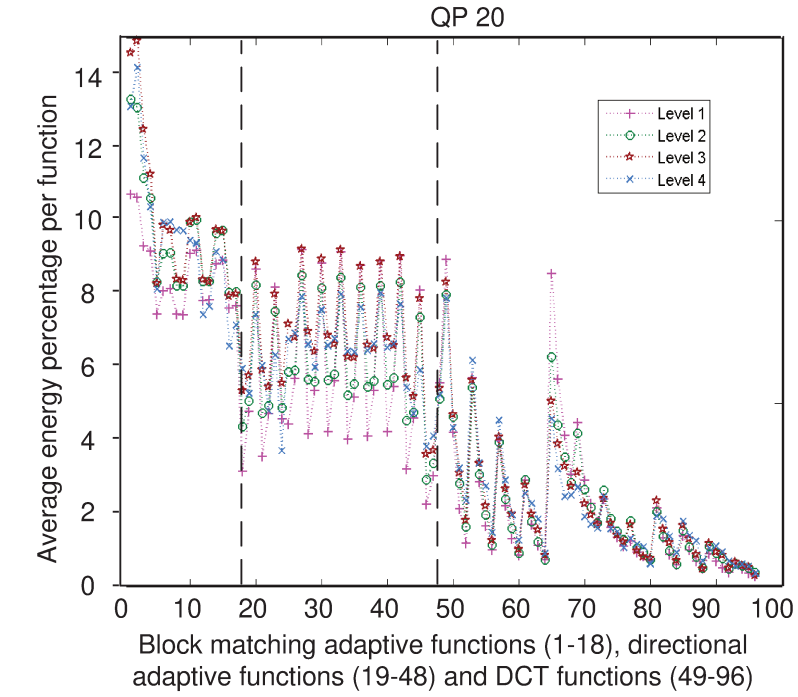
1) The adaptive functions achieve more energy concentration in higher levels (3 and 4), while the DCT functions achieve their best energy concentration result in lower levels (1 and 2). This result was expected once the overall energy exploited by the adaptive functions is higher at the higher levels.

2) The performance of the block matching adaptive functions is better, in general, than the performance of the directional adaptive functions. However, several directional adaptive functions achieve almost the same energy concentration in all levels, specially at the low bit-rate scenario with QP 60. The same regular behavior is not observed for the block matching adaptive functions.

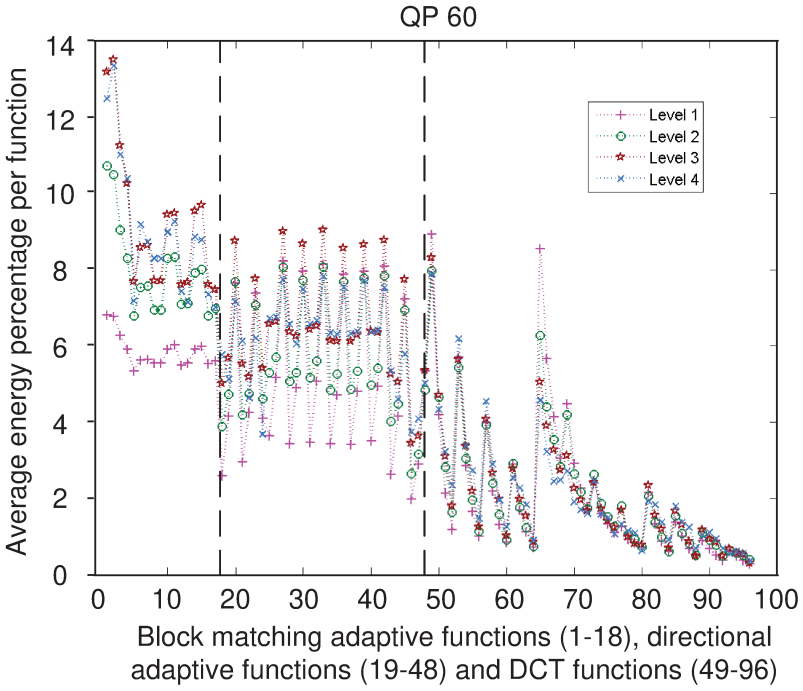
3) The overall performance of the adaptive functions is better at the high bit-rate scenario with QP 20 than at the low bit-rate scenario. The performance at low bit-rate scenario is specially affected in lower levels (1 and 2).

4) The overall performance of the DCT functions is insensitive to the bit-rate variation.

[3]The quantization method employed by the HSDT is described at Section 5.5.1.



(a)



(b)

Figure 5.16: Performance evaluation for adaptive and DCT functions in terms of average energy percentage per function with *a*) QP 20 and *b*) QP 60.

5) Six directional adaptive functions have very similar performance. That happens because the direction detection techniques are combined with the directional interpolation techniques and the resulting functions are organized in the graphs following a repeated pattern. Therefore, all the best six directional adaptive functions use *interpolation with directional and Euclidean weights enclosing a 16-point neighborhood* (the last directional interpolation method described in Section 5.3.1). It means that the performance of directional adaptive functions is more affected by the directional interpolation method than by the direction detection chosen method.

A special result can be taken from the graphs in Fig. 5.16 and the other graphs obtained with different QPs: the best-in-the-average order for the 2D-DCT functions. The numbers in the basis patterns of Fig. 5.17 indicate the basis order in the matrix transform shown in Fig. 5.7. These functions could be ordered with the standard zig-zag pattern drawn in Fig. 5.17, but a best-in-the-average order can be extracted from the performance evaluation conducted for the DCT functions. Both orders are shown in Fig. 5.17.

Therefore, an additional result obtained with the HSDT is a best-in-the-average scanning order for the 2D-DCT. An interesting aspect of this result is that the first 10 functions read according to the best-in-the-average order are exactly the lower-frequency functions of the residual subbands named H and V in Fig. 5.2. These 10 functions are surrounded with red boxes in Fig. 5.17.

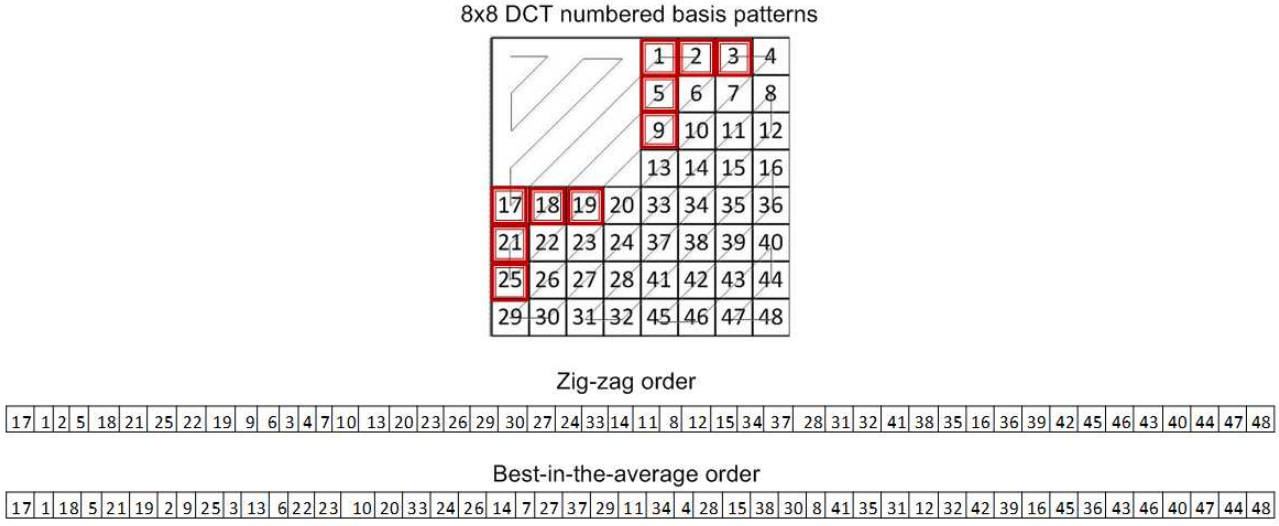


Figure 5.17: Zig-zag order and best-in-the-average order for the 2D-DCT.

Besides the 10 best functions identified for the DCT, the following 15 best adaptive functions were also selected (the first 8 are block matching adaptive functions and the remaining are directional adaptive functions - all directional functions employ *interpolation with directional and Euclidean weights enclosing a 16-point neighborhood*):

1) One-step
2) One-step mean
3) Two-step
4) LBP
5) SASI - 1 Value - non-normalized - similarity distance metric
6) SASI - 4 Values - non-normalized - similarity distance metric
7) SASI - 8 Values - non-normalized - similarity distance metric
8) SASI - 8 Values - normalized - Euclidean distance metric
9) Hough transform
10) LAS - 4 Directions
11) LAS - 12 Directions
12) LAS - 24 Directions
13) SASI - 4 Value
14) SASI - 8 Values
15) Fit plane

It is important to note that the performance shown in Fig. 5.16 is the average performance. Individual performance per block can achieve much better results, such as illustrated by Figs. 5.18 and 5.19. Both figures show the results of adaptive functions applied to code the same residual block, which was extracted from the image shown in Fig. 5.5. In Fig. 5.18, the residual block is coded with the result of the block matching adaptive function *One-step mean* and in Fig. 5.19 the residual block is coded with the result of the directional adaptive function *LAS - 24 Directions*. It can be seen in Fig. 5.18 that the full 8x8 corresponding block is indeed very similar to the full 8x8 block being encoded/decoded. Therefore, the basis vector, i.e., the corresponding block read by columns and orthonormalized, is also very similar to the residual block and is able to represent 78% of the total residual block energy. In Fig. 5.19, the directionally interpolated block is not so similar to the full 8x8 block being encoded/decoded. Consequently, the resulting basis vector is able to represent only 22% of the total residual block energy. Although the performance

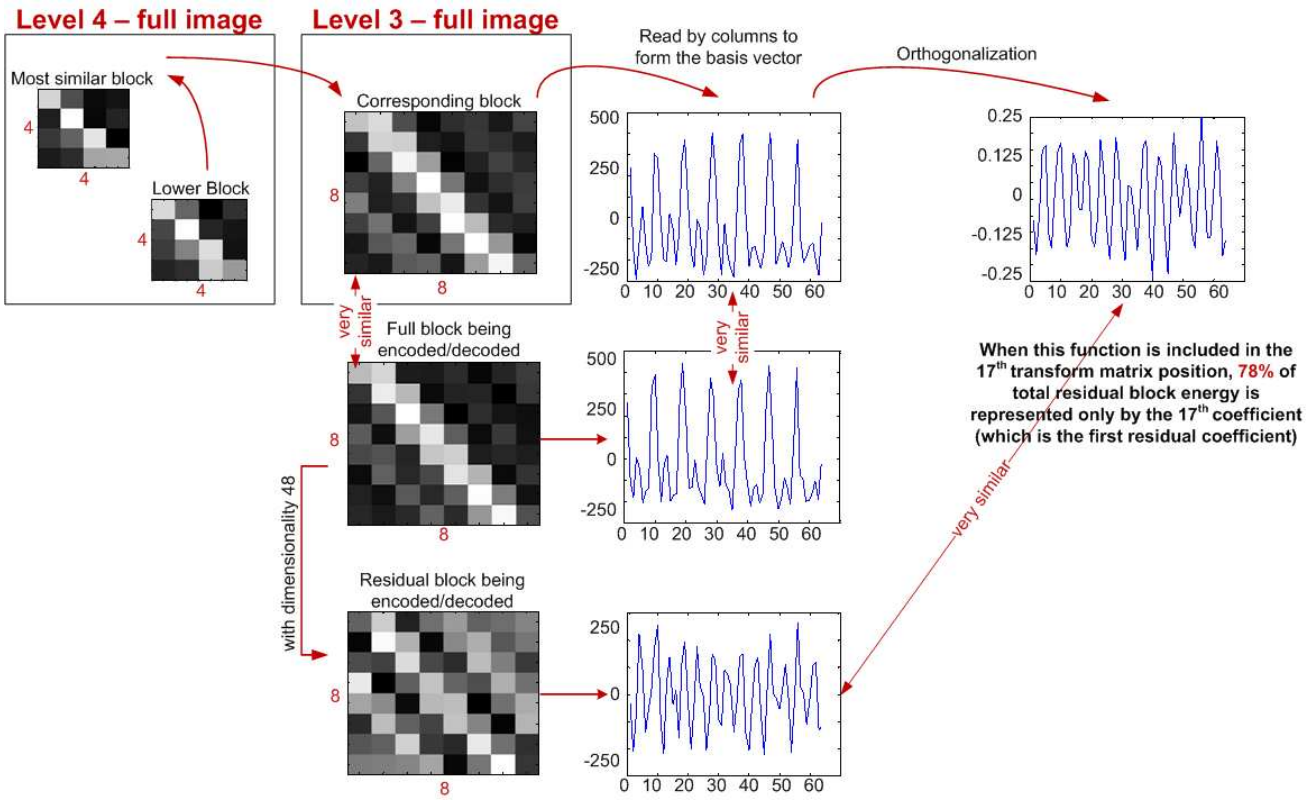


Figure 5.18: Encoding/decoding process employing a block matching adaptive function.

of the directional adaptive function is almost 4 times worse than the performance of the block matching adaptive function for this residual block, it is still almost 3 times better than the average performance of the directional adaptive functions shown in Fig. 5.16.

5.4 Basis Vectors Ordering

The different performance results presented in the previous section raised the need to identify the "goodness" of each adaptive function with respect to the specific characteristics of each residual block. For instance, an uniform residual block localized in a correlated image area could be successfully coded with a block matching adaptive function. On the other hand, a directional residual block localized in a non-correlated area probably would be better coded with a directional adaptive function. And finally, a non-directional residual block in a non-uniform area would not benefit much from adaptive functions and should be coded with the standard 2D-DCT.

Therefore, it is clear that the characteristics of each residual block must be measured and the best adaptive or DCT functions to code each type of residual block must be identified. This process was performed in two steps in this work. In the first step, statistical measures considering

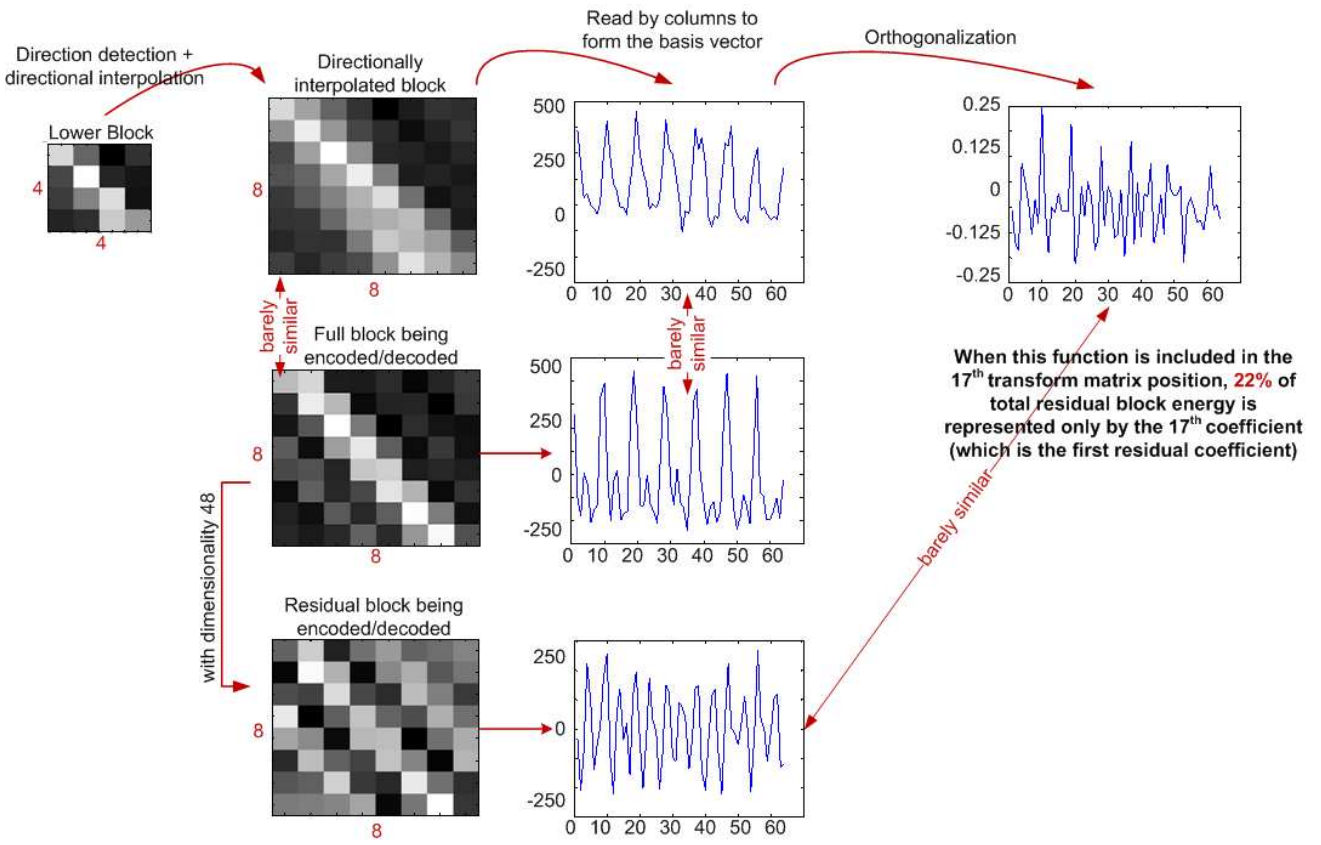


Figure 5.19: Encoding/decoding process employing a directional adaptive function.

not only the residual block itself but also its neighborhood were computed. In the second step, a training process was performed following the same procedure described at the beginning of Section 5.3.3, except that the procedure was also adopted for the pre-identified best 10 DCT functions, it was only adopted for the pre-identified best 15 adaptive functions and the average energy percentage per adaptive function was not computed in the end of the training process. Instead of that, the individual energy concentration performance of each considered adaptive and DCT function per block was stored. Multiple linear regression was then applied to determine hyperplanes of dimension m for each adaptive and DCT function, where m is the number of statistical measures computed per block. These two steps are detailed in the sequel.

Five statistical measures were computed, all for the expanded lower resolution 8x8 full block, which is available both at encoding and decoding processes. Two measures exploit the block correlation, two other measures exploit the neighborhood correlation and the last measure is the block standard deviation.

The two measures exploiting block correlation employ the LAS descriptor with 24 directions described in Section 5.3.1. This descriptor provides spatial activities measures computed in different 24 directions inside the block. If the spatial activity is small in some direction, probably the block has strong correlation in that direction. Then, the smallest measure is kept as the first statistical block correlation measure. There is then an inverse relation, small spatial activity indicates strong correlation. But this measure alone does not define if the block is correlated in only one direction, i.e., directional, or if it is correlated in all directions, i.e., uniform. Therefore, the other kept statistical correlation measure is the standard deviation of the 10 smallest spatial activities measures. If the deviation is small, probably the block is uniform. Conversely, the block is directional.

The two measures exploiting neighborhood correlation employ the same *One-step* method described in Section 5.3.2, except that the distance metric (SAD, in this case) result is divided by the block standard deviation here. Like performed for the block correlation statistical measures, the smallest SAD measure is kept as the first statistical neighborhood correlation measure and the standard deviation of the 10 smallest SAD is kept as the second statistical neighborhood correlation measure. These are also inverse measures, a small SAD measure means that there is, at least, one neighbor block highly correlated to the current block. If the deviation is also small, the neighborhood is probably uniform and several blocks are correlated to the current block.

Summarizing, the five statistical measures computed and stored per block are:

1) Standard deviation

2) Smallest spatial activity (highest block correlation)

3) Standard deviation of the 10 smallest spatial activities

4) Smallest SAD (highest neighborhood correlation)

5) Standard deviation of the 10 smallest SADs

The relation between the energy concentration performance of each adaptive/DCT function and the statistical measures per block will be defined through multiple linear regression. For all blocks of the several selected images, the five statistical measures were stored and the performance of the 10 DCT functions and of the 15 adaptive functions were also stored. At the end of the training process, the multiple linear regression determines the hyperplanes that minimize the sum of the squares of the distances (where the distances are parallel to the y-axis) between the hyperplanes and the statistical measures. One hyperplane is determined for each of the 10 selected

DCT functions and each of the 15 selected adaptive functions. The equation of the hyperplane is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 \tag{5.5}$$

where y is the *dependent* variable, x_{1-5} are the *independent* variables or *predictors* and β_{1-5} are the *coefficient regressors*. The *coefficient regressors* are the main result of the training process. Therefore, they are stored to be used on-the-fly in the encoding/decoding processes of each block. In this work, x_{1-5} are the 5 statistical measures and y is the energy concentration percentage. As Eq. (5.5) is computed for all selected adaptive and DCT functions, y gives the approximate energy concentration achieved by each function for a block with characteristics defined by the statistical measures x_{1-5}. Clearly, the chosen function to encode/decode the block is the function which returns the highest y value.

The visual representation of the hyperplanes can not be done, but for illustration purposes, the regression planes were computed considering only 2 statistical measures: the *Smallest spatial activity* and the *Standard deviation of the 10 smallest spatial activities*. The planes are shown for four functions in Fig. 5.20: block matching adaptive function *One-step*, directional adaptive function *SASI-8 values*, DCT function number 1 in Fig. 5.17 and DCT function number 17 also in Fig. 5.17. It can be seen, for instance, that blocks with high deviation and small spatial activity, which are probably directional blocks as previously observed, would be better coded with adaptive functions. On the other hand, blocks with small deviation and high spatial activity are not correlated and would be better coded with the DCT functions. The graphs in Fig. 5.20 do not consider neighborhood correlation statistical measures, only block correlation statistical measures. Therefore, only a partial evaluation of the function performance can be done based on them.

A very important aspect which was not handled yet in this work is that, not only the best function for the first transform matrix position must be identified, but also the best functions for all other matrix positions. If the first basis vector represents 50% of the total residual energy, but the other 50% of the energy is spread across the remaining 47 basis vectors or even concentrated in the last basis vectors, the overall performance of the transform will not be satisfactory. The average energy percentage obtained through the regression hyperplanes can not be used to order

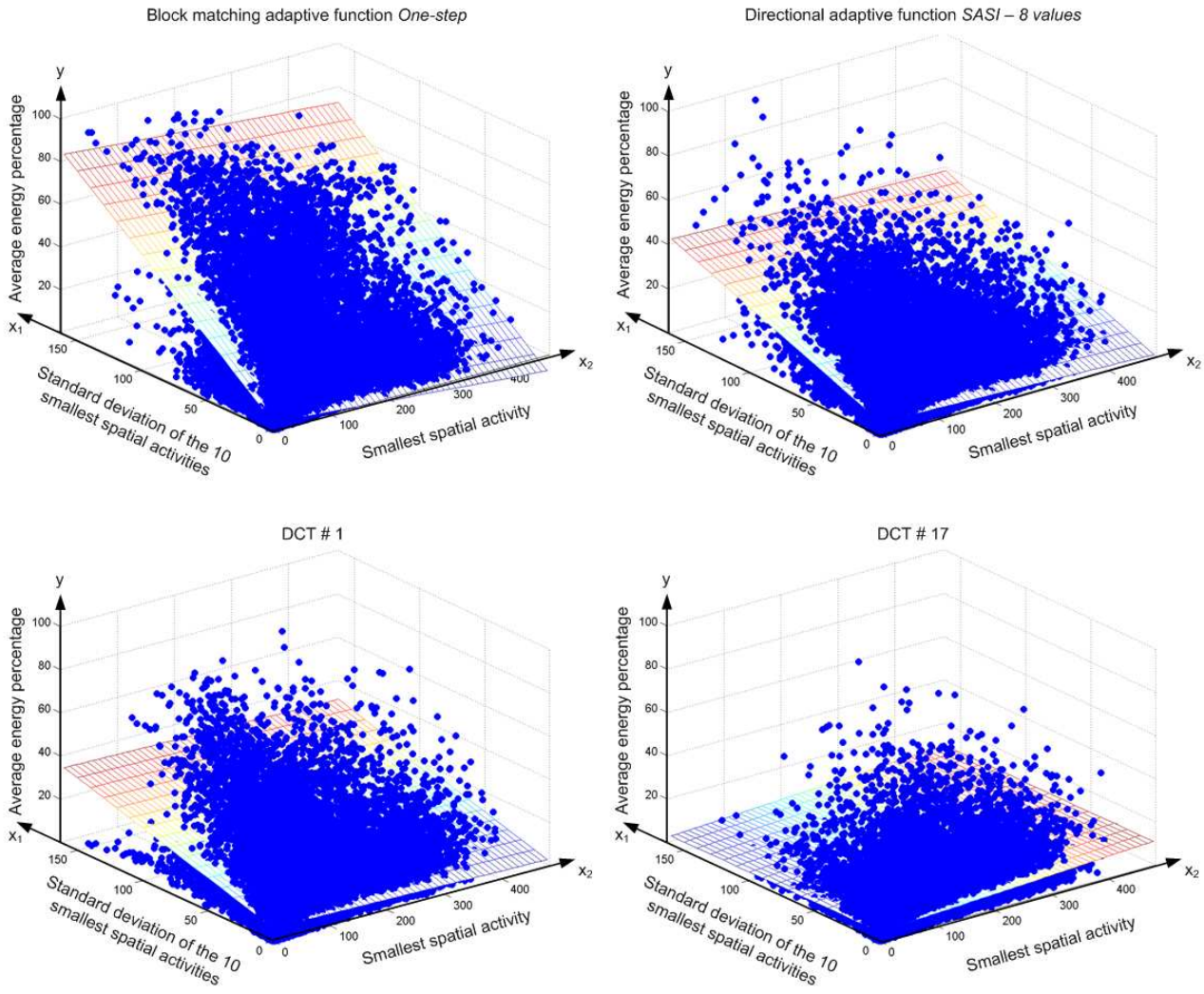


Figure 5.20: Regression planes of two adaptive and two DCT functions. The DCT function numbers refer to the numbers in Fig. 5.17.

the adaptive and DCT functions in the remaining basis positions because similar functions exploit similar characteristics of the blocks and produce similar basis vectors. Therefore, if the first and the second basis vectors are very similar, after the orthonormalization, the second basis vectors will become too randomized and most of the energy it could represent would have already been represented by the first basis vector.

The necessary ordering of the basis vectors is performed in this work by two developed methods: regression hyperplanes and KLT. In first method, regression hyperplanes were also computed for the second transform matrix position in the same way the hyperplanes were computed for the first transform matrix position, but considering which specific function was chosen as the best

first one. Therefore, the whole training process was performed for each case, i.e., for each function in the first position, all other remaining functions were tried at the second position and all performance results were stored. In the end, the hyperplanes for all combinations were computed, resulting in 24 coefficient regressors sets for each of the 25 possible functions selected for the first position (15 adaptive functions plus 10 DCT functions).

As this training process is computationally expensive, the coefficient regressors sets were not computed for the third position, the fourth position and so on. The performance of the functions in the third position was also measured following the same previous procedure, but now considering which specific functions were chosen as the best ones for the first and second matrix positions. All performance results for each pair of functions in the initial positions were stored and at the end of the training process, the average energy percentages were just used to order the remaining functions, without the computation of new coefficient regressor sets for each case. Summarizing, coefficient regressor sets were computed and stored for the first and second matrix positions and fixed orders were computed and stored for the remaining positions.

The other possible method applied in this work to order the basis vectors is the KLT. The possible set of best adaptive functions to code the current residual block is identified and the KLT is applied to these functions. The KLT decomposes these functions as a linear combination of the *principal components* of the functions. Therefore, the functions are uncorrelated and the common good characteristics of the best functions are reinforced, while the uncommon undesirable characteristics are weakened.

Before the KLT is applied, the set of best adaptive functions must be determined. The coefficient regressor sets computed to identify the best function in the first matrix position, as explained in Section 5.3.3, are applied for this task. Eq. (5.5) is computed only for the adaptive functions, not for the DCTs. After the computation, only the adaptive functions that achieve, at least, 10% of energy concentration are selected. Following, the selected functions are orthogonalized, but not normalized. The energy concentration results are also used as weights to differentiate the selected functions. If a selected function achieves 80% of energy concentration, for instance, its content should be more represented by the *principal components* than another function that achieves 13%. After this process is completed, a set of weighted orthogonal best adaptive functions is determined and the KLT can be applied to this set.

Initially, the covariance matrix is computed and the eigendecomposition is performed on it. Following, the eigenvalues are sorted in descending order and the corresponding eigenvectors are arranged to form the KLT matrix. Finally, only the initial eigenvectors of the KLT matrix are kept in order to form the final basis vectors of the HSDT transform matrix. The number of kept eigenvectors is equal to the number of selected adaptive functions in the set. This number is variable, since it depends on the performance of the functions for the specific block being coded. The remaining positions of the HSDT transform matrix are filled with the default 2D-DCT ordered according to the best-in-the-average order shown in Fig. 5.17. The only difference is that the initial 10 best DCT functions are also ordered according to the energy concentration results obtained with Eq. (5.5) for these functions. The same 2D-DCT ordering procedure is applied to code the residual block only with the 2D-DCT when any adaptive function meets the requirement of achieving, at least, 10% of energy concentration.

5.5 Full Image Codec

As a proof of concept, a full image codec was implemented to evaluate the potential gain from the HSDT. For simplicity, only the image luminance component is considered. In order to create zero-mean images, the value of 128 is subtracted from the original luminance components. The quantization and entropy coding stages of the image codec will be detailed in this section.

5.5.1 ***Quantization***

The design of the quantization process, in particular regarding the optimal quantization step size and dead zoning/quantization rounding control mechanisms, has been widely studied and one of the most important observations was that coefficients distribution followed, in most cases, a Laplacian distribution. This assumption is followed within the design of many codecs, including H.264/AVC, in an attempt to improve the rate distortion performance.

The quantization process applied by the H.264/AVC [26,85] is illustrated in Fig. 5.21 and can be seen as a *deadzone with variable width uniform scalar quantizer*. This is the process adopted also in this work to quantize the residual coefficients resulting from the transform application. The quantization of a coefficient equal to W is performed as

$$WQ = \left\lfloor \frac{|W| + f}{\Delta} \right\rfloor \cdot sgn(W) \tag{5.6}$$

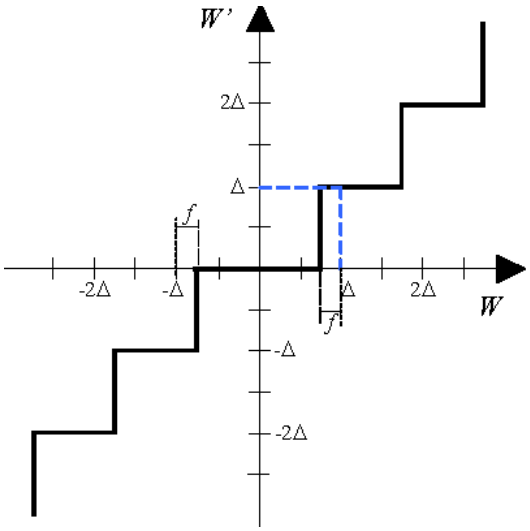


Figure 5.21: Deadzone with variable width uniform scalar quantizer.

where WQ is the quantized coefficient, Δ is the quantization step size and f is the rounding parameter. In this work, $f = \Delta/6$ in an attempt to approximate the most probable coefficients Laplacian distribution [10]. The addition of a positive rounding parameter results in a smaller deadzone. The reconstructed coefficient $W^{'}$ is computed as

$$W^{'} = \Delta \cdot WQ. \tag{5.7}$$

The following methods were developed in this work to define the quantization step size Δ, from now on referred to as Quantization Parameter (QP):

1) Store the maximum absolute coefficient value $|W|$ per decomposition level l and compute QP as $\left\lceil \frac{|W|}{2^{l+1}} \right\rceil$. This is an implicit quantization method, where the formula computes the quantization parameters according to the coefficients values. By this method, the step size decreases at each decomposition level, while the number of quantization levels increases by an approximate factor of 2. The computed QPs (one per decomposition level) are stored at the header of the coded image to be used by the decoder.

2) Each QP computed per level can be used to quantize all residual coefficients of each block in the level or secondary QPs can be derived from this base QP to code each specific residual coefficient with a more appropriate QP. As the initial residual coefficients tend to concentrate more energy than the final ones, the possibility of coding each coefficient with a different QP may positively influence the final performance of the HSDT.

3) Define the quantization parameter as a fixed codec parameter. This an explicit quantization method, where the QP is pre-defined and used to code all residual coefficients from blocks in all decomposition levels.

The QP definition methods presented in items *1-3)* are all implemented at the full image codec. However, as the main purpose of it is to evaluate the HSDT, not the whole encoding process, the simplest method specified in item *3)* is chosen to show the transform results in Section 5.6.

5.5.2 ***Entropy coding***

The entropy coding stage of the full image codec developed in this work employs an arithmetic coder. Arithmetic coding provides a practical alternative to variable-length codes that can more closely approach theoretical maximum compression ratios [86]. An arithmetic coder converts a sequence of data symbols into a single fractional number and can approach the optimal fractional number of bits required to represent each symbol.

Before the arithmetic coder is applied, the residual coefficients of all blocks must be read per level and turned into symbols. Two methods were implemented here to read the coefficients: sequential and progressive. In the sequential method, all 48 residual coefficients of the same block are read in sequence and encoded together. On the other hand, in the progressive method, similar-positioned coefficients of all blocks are read and encoded together, followed by the next similar-positioned coefficients of all blocks, and so on. Therefore, initially the first residual coefficient (the 17^{th} coefficient) of all blocks in the same level is read, then the second residual coefficient is read, etc. It is important to note that the residual coefficients of blocks with total energy zero after the quantization stage are not read and coded. An extra bit is used to identify these blocks. As after the transform application, the residual coefficients should be arranged in decreasing in-the-average order, generally the progressive scanning provides better results than the sequential scanning. Therefore, the progressive scanning is chosen to show the transform results in Section 5.6.

The next step is to define a statistical model for the residual coefficients. The model adopted is not optimized and starts with a uniform distribution for each decomposition level. The maximum bounds of the model can be identified through the theoretical or signal dependent modes described in the sequel.

The theoretical mode considers that, once all functions at HSDT are normalized and the original luminance image is zero-mean, the range for the coefficients at the first decomposition

level is $8 \times (-127) \ldots 8 \times (128)$. Therefore, the maximum absolute coefficient value, from now on referred to as V_{max}, would be 8×128. At each decomposition level there is an additional gain of 2 for the lower resolution image. So, at same level l, $V_{max}(l)$ is $8 \times 2^{l-1} \times 128$.

The signal dependent mode considers that, once all functions at HSDT are, not only normalized, but also orthogonal, the total energy is conserved according to the Parseval relation. Therefore, in this mode, the total energy of the residual coefficients of all blocks is computed per level before the HSDT, when all decomposition levels are coded with the 2D-DCT, and only the maximum total energy value per level is kept. After the application of the HSDT, relying on the energy-preserving property, $V_{max}(l)$ will be the root square of the maximum total energy value per level.

Both the theoretical and the signal dependent modes identify $V_{max}(l)$. Therefore, the maximum bound of the model is $2 \times V_{max}(l)$. As the theoretical bounds can be unnecessarily higher than the signal dependent bounds, these are the chosen for the results in Section 5.6. After defined, the maximum bound is quantized and used to compute the intervals between the values in the statistical model. A cumulative distribution function is then computed and the uniform resulting model, varying from 0 to 1, is finally scaled to $2^{14} - 1$.

Once the statistical model is defined, the next step is to turn the residual coefficients into symbols. In order to keep the approximate Laplacian distribution, the coefficients are interleaved according to the following rule:

$$symbol = \begin{cases} 0, & \text{if coefficient} \equiv 0 \\ |\text{coefficient}| \times 2 + 1, & \text{if coefficient} < 0 \\ |\text{coefficient}| \times 2, & \text{if coefficient} > 0 \end{cases} \tag{5.8}$$

Finally, the arithmetic coder converts each symbol into bits and updates the model. It is important to note that the initial uniform statistical model is updated only on-the-fly after each symbol encoding/decoding. As the main purpose of the full image codec is to evaluate the HSDT, the entropy coding stage, like the quantization stage, is kept as simple as possible.

After all levels are entropy coded, the final bitstream is produced, written in the coded file and the coding process is completed.

5.6 Results

The full image codec was implemented in *Matlab* language without fixed variables and in an object oriented fashion. Therefore, it is possible to modify several parts of the codec, such as to add a new adaptive function, to change the choice or the order of the best adaptive functions, to include new quantizers and entropy coders or even to change the size of the 8x8 HSDT.

This section shows the results obtained with the HSDT employing the coding options chosen and detailed in the previous sections of this chapter. The transform performance was evaluated in terms of transform coding gain, energy concentration, distortion versus bit-rate curves and visual performance. The results are shown for three images of size 512x512 selected from a large database of perceptually diverse content [66]. The first image is the *Palm leaf* picture, initially introduced in this work in Fig. 5.4. This image has a strong directional content and many details. The second image also contains many details combined with a smooth sky area. It is shown in Fig. 5.22(a) and portrays the *New York City skyline*. Conversely to the first two images, the third image, presented in Fig. 5.22(b), has a predominantly smooth and correlated content. An internal area of the *Guggenheim Museum* is covered in this image.

(a) (b)

Figure 5.22: Luminance planes of the images: *a*) New York City skyline and *b*) Guggenheim Museum interior.

As the HSDT was not optimized in terms of computational efficiency and most parts of the codec include tests to verify if the codec is running in training or coding mode, the encoding and decoding times were not measured. But an analysis was done through the *Matlab Profile Tool* to identify the most time-consuming coding and decoding stages. The analysis is shown in Table 5.1 and it can be seen that, as expected, the computation of the adaptive functions is the most expensive coding/decoding stage. The computation of the SASI descriptor with 8 values appears as particularly expensive, being responsible for almost 70% of the coding/decoding times. A new choice of adaptive functions or the replacement of SASI by another texture descriptor with similar efficiency should speed up the coding/decoding processes. Once the HSDT is a symmetric codec, i.e., the same operations performed at the coder to compute the transform are performed at the decoder, the coding and decoding times are equal.

5.6.1 ***Transform Coding Gain***

The main objective of transforms employed in compression schemes is to compact energy efficiently in few coefficients. The transform coding gain (TCG) is an appropriate measure for

Coding/decoding stages						**Times**
Transform	Adaptive functions	Block matching		SASI	8 values	69.8%
					4 values	20.2%
					1 value	3.8%
				LBP		4.0%
				One-step		0.1%
		Directional	Direction detection	SASI	8 values	0.3%
					4 values	0.2%
				LAS	24 directions	0.1%
			Direction interpolation	Directional and Euclidean weights - 16 points		0.3%
	Basis orthonormalization					0.3%
	Statistical measures computation					0.1%
Quantization						0.5%
Arithmetic coder						0.3%

Table 5.1: Average time percentage spent in each coding/decoding stage.

comparing energy concentration performance of various transforms. Figs. 5.23, 5.24 and 5.25 present the TCG versus QP curves, respectively, for the images *Palm leaf*, *New York City skyline* and *Guggenheim Museum*. The curves compare the performance of the HSDT, employing the two basis vectors ordering methods described in Section 5.4, with the performance of the 2D-DCT employing the best-in-the-average order shown in Fig. 5.17 and the DCT regression hyperplanes order. The first DCT order is referred to as *fixed order* in the figures, while the second is referred to as *adaptive order*.

For orthogonal transforms, the TCG can be computed by Eq. (2.6). For computing the TCG of an orthogonal hierarchical transform like HSDT, Eq. (2.6) can be rewritten as:

$$TCG = \frac{\frac{1}{N}\sum_{i=1}^{L} \sigma_i^2 C_i}{\left(\prod_{i=1}^{L} \left(\sigma_i^2\right)^{C_i}\right)^{\frac{1}{N}}} \tag{5.9}$$

where L is the total number of decomposition levels, C_i is the total number of coefficients in the i^{th} level and $N = \sum_{i=1}^{L} C_i$.

The TCG was computed after the quantization stage. Therefore, only the non-zero blocks were included in the computation. The number of non-zero blocks varies according to the image

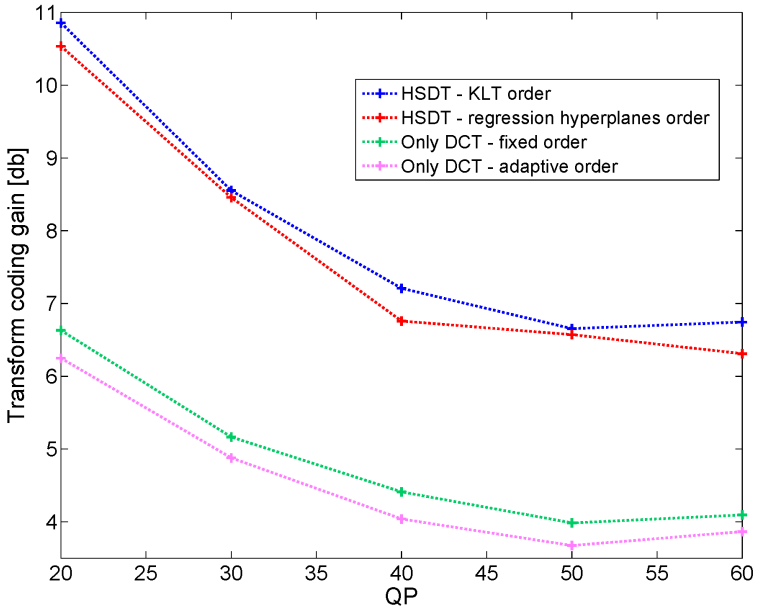


Figure 5.23: TCG versus QP for the *Palm leaf* image.

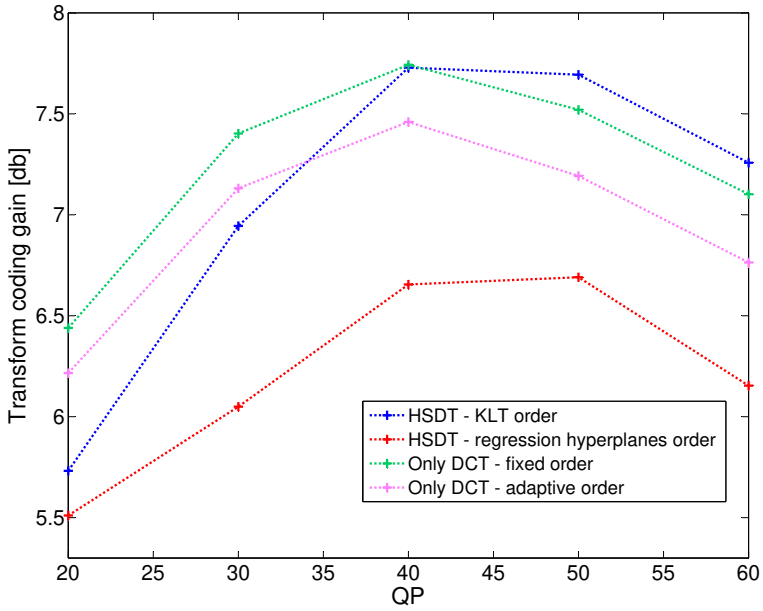


Figure 5.24: TCG versus QP for the *New York City skyline* image.

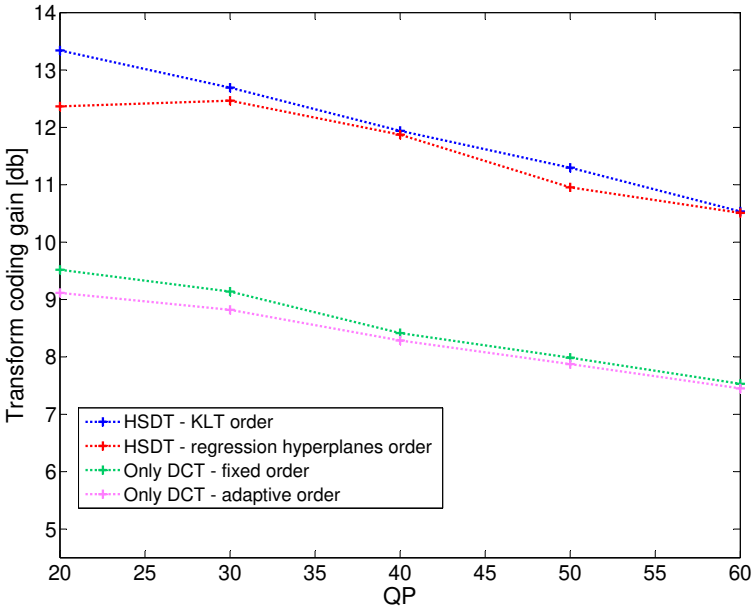


Figure 5.25: TCG versus QP for the *Guggenheim Museum* image.

content and, certainly, the quantization parameter. The uniform *Guggenheim Museum* image, for instance, has only 21.37% non-zero blocks after being quantized with QP 20. On the other hand, when the same low QP is set to quantize the rich in details *Palm leaf* and *New York City skyline* images, 62.61% and 65.20% of the blocks are non-zero, respectively.

As expected, the best HSDT performance in terms of TCG is obtained for the *Palm leaf* and *Guggenheim Museum* images. It can be seen in Figs. 5.23 and 5.25 that the HSDT outperforms the

DCT in 4 dB in the average. It can also be seen in these figures that, for the DCT, the *fixed order* outperforms the *adaptive order* by approximately 0.5 dB, and for the HSDT, the *KLT order* outperforms the *regression hyperplanes order* with different gains. The difference between the TCGs obtained for the HSDT with the two basis vector ordering methods is specially perceptible for the *New York City skyline* image in Fig. 5.24. As it is an image with very diverse content, probably several blocks are better coded only with the DCT and the usage of the *KLT order* enables this option.

An interesting parallel can be made between the computation of the TCG for the DCT in Section 2.3.2 and in this chapter. In Section 2.3.2, the one-dimensional TCG of the 4x4 DCT was computed for a stationary Gauss-Markov input with correlation coefficient $\rho = 0.9$ and the result was 5.38 dB. In this chapter, the results vary approximately from 3.5 dB to 9.5 dB, according to the image and QP. This difference in the TCGs arises because in Section 2.3.2, the ideal model of a high-correlated image was adopted and the coefficients variances were obtained from the autocorrelation coefficient matrix. In this chapter, the variances of the real residual coefficients c were computed as

$$\sigma_c^2(n) = E[c^2(n)] \quad for \quad n = 1 \ldots 48. \tag{5.10}$$

The equality in Eq. (5.10) is valid if $E[c(n)] = 0$, which holds true once the images are zero-mean and the transform is orthonormal. Another reason for the different TCGs is the transform size. In Section 2.3.2, the transform size was 4x4 and in this chapter, the transform size is 8x8. The higher the transform size, the higher the transform coding gain, specially for correlated images.

5.6.2 ***Energy Concentration***

After providing high energy concentration in few coefficients, it is desirable that the transform coefficients be ordered in a "decreasing in the average" order. The average percentage energy per coefficient is shown in Figs. 5.26, 5.27, and 5.28, respectively, for the images *Palm leaf*, *New York City skyline* and *Guggenheim Museum*. The curves presented in these figures were computed for enabling comparisons among the same HSDT and DCT methods described in Section 5.6.1. One difference is that, in these figures, the curves are not varying per QP. Instead of that, only the average result for QPs ranging from 20 to 60 is presented. The average result can represent

appropriately the transforms performance in terms of energy concentration because, although the absolute percentage energy per coefficient varies slightly according to the QP (specially for the first residual coefficients), the overall decay curve profile is almost insensitive to the QP variation.

For all three images presented here, the HSDT outperforms the DCT providing both smoother decay curves and higher energy concentrations in the first residual coefficients. For the *Palm leaf* image, approximately 55% of the energy is concentrated, in the average, only in the first residual coefficient. The HSDT performance is even better for the *Guggenheim Museum* image, where more than 70% of the energy is concentrated in the first residual coefficient. Although the HSDT performance for the *New York City skyline* is not so impressive, it still can also be considered satisfactory, once more than 25% of the energy is concentrated in the first residual coefficient.

An interesting observation here can be made for the *fixed* and *adaptive orders* of the DCT. As mentioned in Section 5.6.1, the *fixed order* outperforms the *adaptive order* in terms of TCG for all three images by approximately 0.5 dB. The reason of this can be found specially in Fig. 5.27. The *fixed order* for the DCT achieves higher disperse energy peaks than the *adaptive order*. As the TCG only evaluates energy compaction, not coefficients ordering, the TCG results for the DCT with *fixed order* will be consequently higher. However, as shown for all three images in Figs. 5.26, 5.27, and 5.28, the DCT with *adaptive order* compacts more energy in the first residual coefficients and provides smoother decay curves.

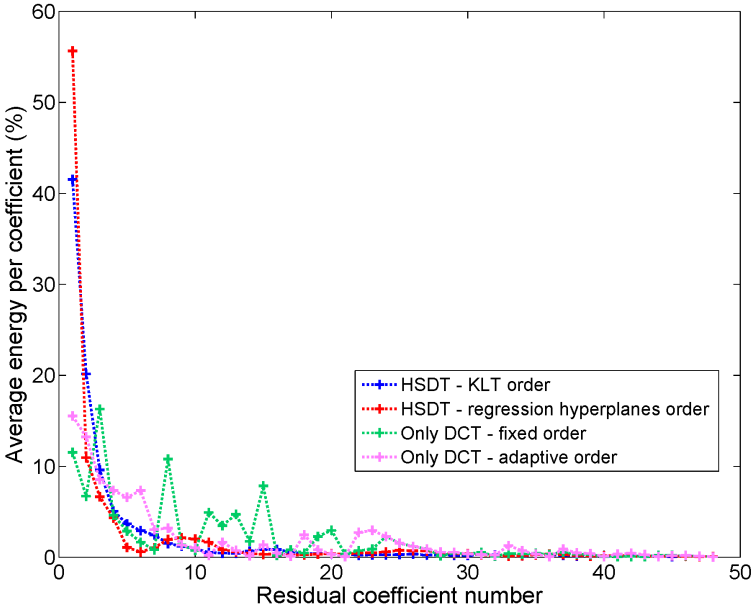


Figure 5.26: Residual coefficients energy concentration for the *Palm leaf* image.

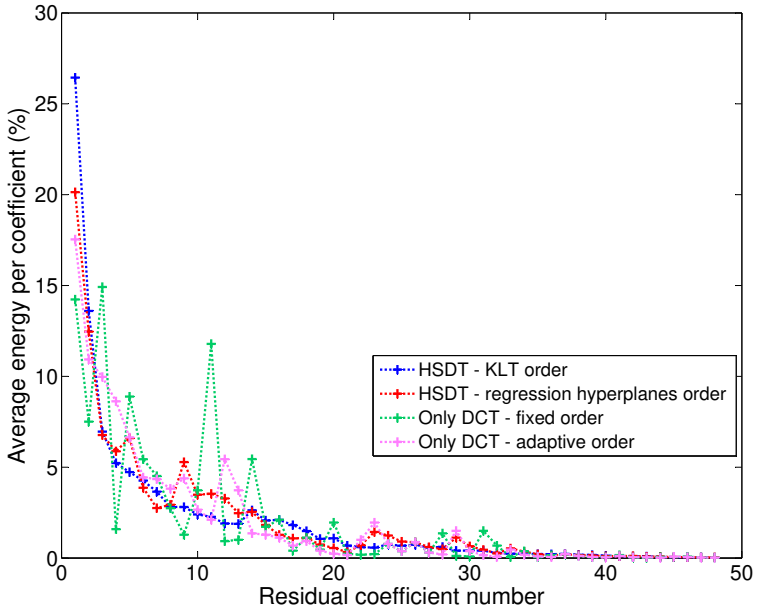


Figure 5.27: Residual coefficients energy concentration for the *New York City skyline* image.

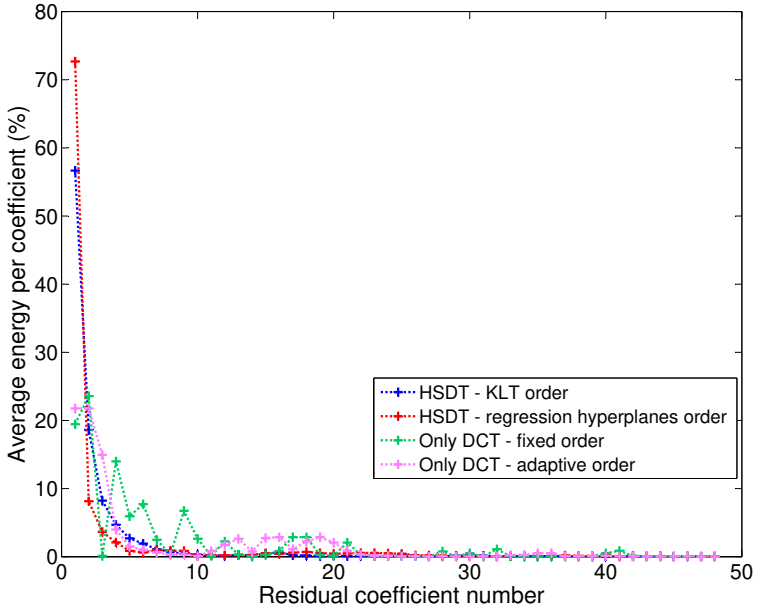


Figure 5.28: Residual coefficients energy concentration for the *Guggenheim Museum* image.

5.6.3 ***Distortion versus Bit-rate Curves***

The distortion versus bit-rate performance comparisons will be performed among JPEG, JPEG XR, JPEG 2000 and the full image codec built for the HSDT using the two basis vector ordering methods described in Section 5.4: KLT and regression hyperplanes. The software implementations for the JPEG and JPEG 2000 image codecs are the default implementations provided by the *Matlab* environment. For the JPEG XR, the software implementation obtained

in [48] is used. Compared with the three image codecs, the HSDT codec has more comparable characteristics with JPEG XR than with JPEG and JPEG 2000. Among other comparable features, JPEG XR also employs a block transform and its *frequency mode* enables quality and spatial scalability functions. Certainly, JPEG XR has several features not present at the HSDT codec, such as an optional second transform lapped stage to reduce blocking artifacts, a flexible quantization stage, a prediction stage after the quantization and an optimized entropy coding stage. However, as previously mentioned in Section 5.5, the main purpose of the full image codec implementation is to evaluate the HSDT performance and, therefore, the other codec stages should be kept as simple as possible. The results for the JPEG XR were obtained in *frequency mode* without the overlapped transform. The other configuration parameters were all set as "default", following the definitions in [48]. More details about the JPEG XR image codec were given in Section 4.2.

Y-PSNR versus bit-rate curves are shown in Figs. 5.29, 5.30, and 5.31, respectively, for the images *Palm leaf*, *New York City skyline* and *Guggenheim Museum*. The results for the *Palm leaf* image, in Fig. 5.29, show that even with the HSDT codec constraints, it outperforms JPEG-XR by approximately 1dB in the average. One can see that the performances of the HSDT with KLT and regression hyperplanes orders are quite comparable, while, as expected, JPEG 2000 outperforms all other image codecs. At low bit-rates, however, the performance difference between the HSDT and JPEG 2000 reaches less than 1 dB. These good results for the HSDT were expected for the *Palm leaf* image, once its strong high correlation and directionality characteristics are well exploited by the HSDT.

Like Fig. 5.29, the performances of the HSDT with KLT and regression hyperplane orders are quite comparable for the *New York City skyline* image in Fig. 5.30. Unlike Fig. 5.29, however, HSDT does not outperform JPEG-XR. The best results obtained with HSDT for the *New York City skyline* image are at low bit-rates and in this scenario, the HSDT performance achieves satisfactorily the JPEG XR performance. This result is satisfactory because the image is very rich in details, not correlated, and the previous sections showed reasonable TCG and energy concentration measures for it. Even though, in the low bit-rate scenario, the performance difference between the HSDT and JPEG 2000 reaches only approximately 0.5 dB.

For the *Guggenheim Museum* image, the results presented in Fig. 5.31 indicate that the HSDT performance is comparable to the JPEG XR performance. Based on the promising TCG and

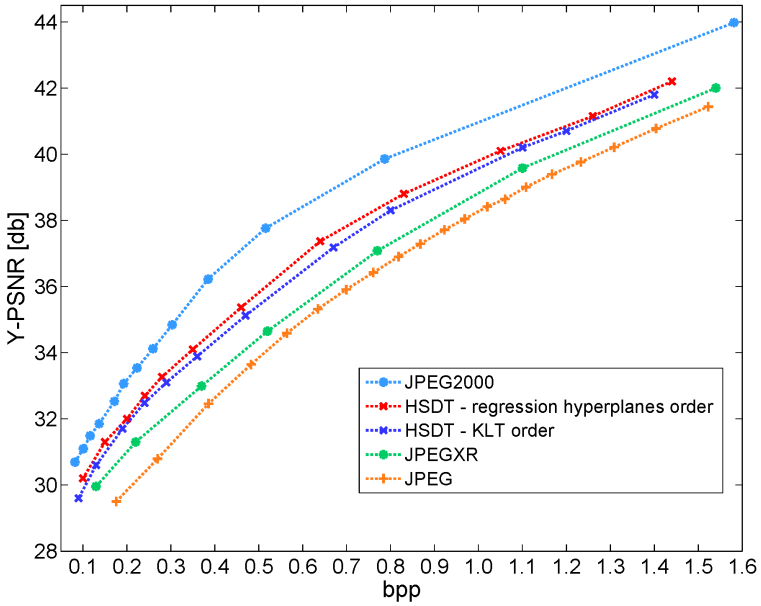


Figure 5.29: Y-PSNR versus bit-rate curves for the *Palm leaf* image.

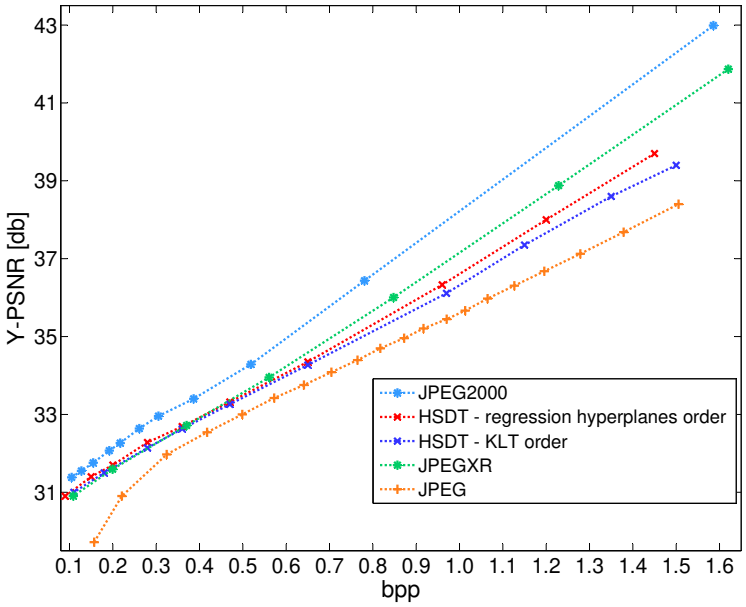


Figure 5.30: Y-PSNR versus bit-rate curves for the *New York City skyline* image.

energy concentration measures computed by the HSDT in the previous sections and also on the performance achieved for the *Palm leaf* image in Fig. 5.29, better results were expected for the *Guggenheim Museum* image. One possible explanation is that most blocks are zero after the quantization stage and, therefore, the number of remaining non-zero blocks that could benefit from an adaptive transform is small. Another possible explanation is that, once the entropy

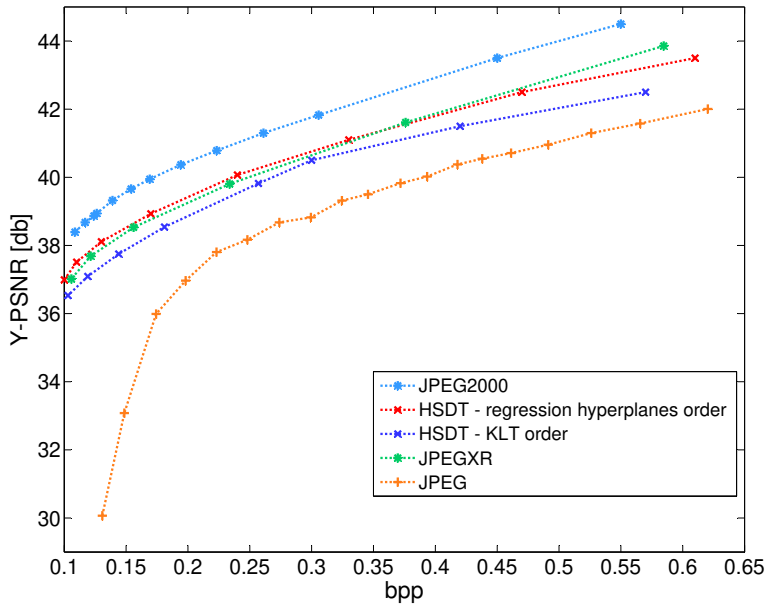


Figure 5.31: Y-PSNR versus bit-rate curves for the *Guggenheim Museum* image.

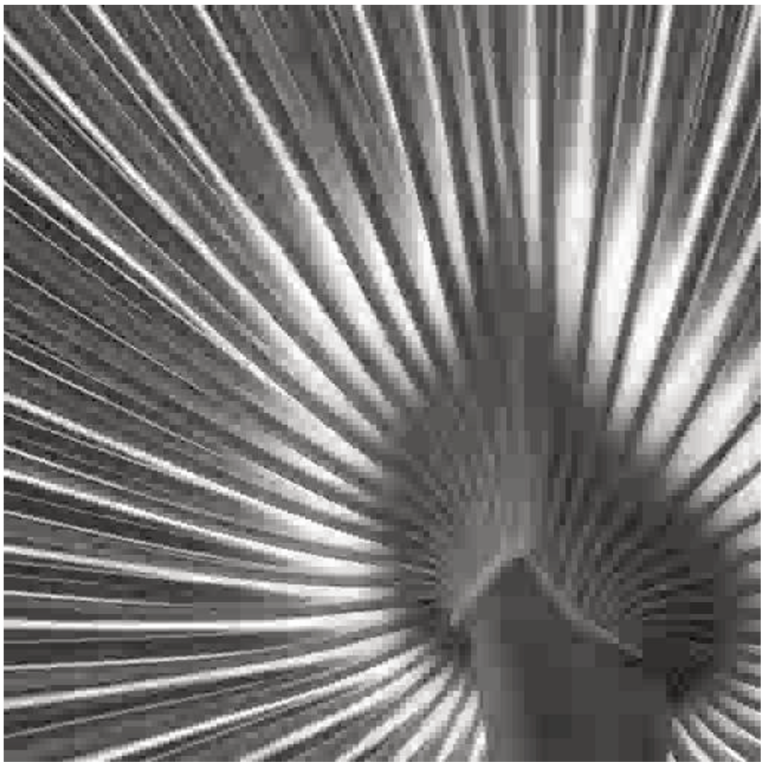
coder stage is not optimized, the codec could not capitalize on the high energy concentration achieved by HSDT for this image.

Visual comparisons are shown for reconstructed images at low bit-rates in Figs. 5.32, 5.33, and 5.34. In all images it is possible to note that, while JPEG XR introduces more visible blocking artifacts, HSDT produces more blurred images for comparable bit-rate scenarios. This is an interesting result obtained for the HSDT because, although it also employs a block transform as the JPEG XR, the blocking artifacts are weakened through the various decomposition levels. The visual different characteristics produced by each transform are specially perceptible for the bulb area of the *Palm leaf* image in Fig. 5.32, the sky of the *New York City skyline* image in Fig. 5.33 and the white large strip of a zoomed area of the *Guggenheim Museum* image in Fig. 5.34. It is important to note here that JPEG XR has as an optional second transform lapped stage to reduce blocking artifacts at low bit-rates.

5.7 Conclusions

This chapter presented a new transform, named HSDT, that is able to exploit the cross-level structural similarities commonly found in hierarchical coding schemes. The transform is adaptive, signal dependent, applied in a block-based fashion and relies on natural properties of most images, such as directionality and neighborhood correlation. A full symmetric image codec was developed

(a)



(b)

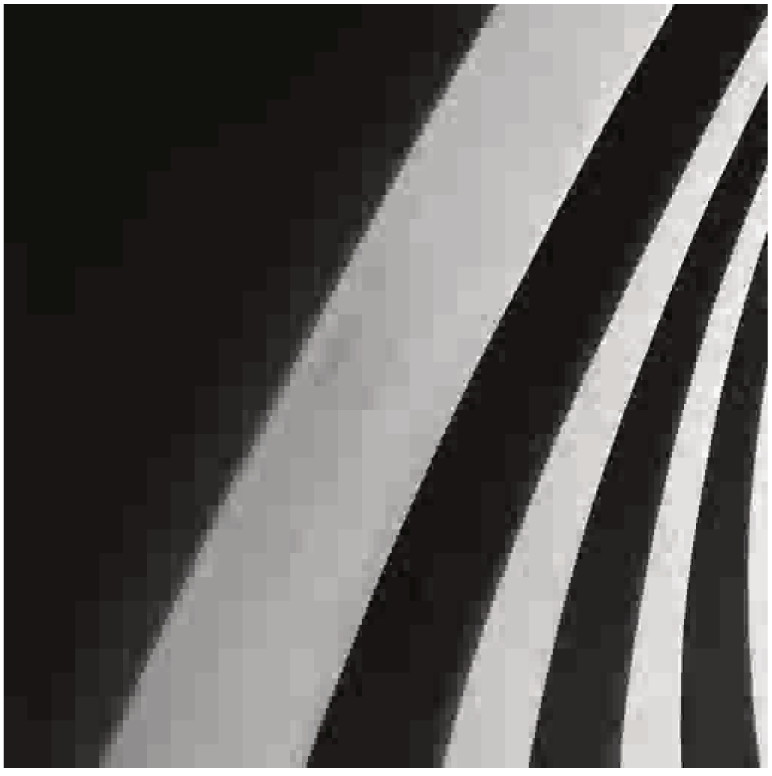
Figure 5.32: Reconstructed *Palm leaf* image previously encoded with: *a*) HSDT at 0.25 bpp (achieving Y-PSNR of 32.5 dB) and *b*) JPEG XR at 0.22 bpp (achieving Y-PSNR of 31 dB).

(a)

(b)

Figure 5.33: Reconstructed *New York City skyline* image previously encoded with: *a*) HSDT at 0.3 bpp (achieving Y-PSNR of 32.3 dB) and *b*) JPEG XR at 0.4 bpp (achieving Y-PSNR of 32.6 dB).

(a)



(b)

Figure 5.34: Reconstructed *Guggenheim Museum* image previously encoded with: *a)* HSDT at 0.1 bpp (achieving Y-PSNR of 36.5 dB) and *b)* JPEG XR at 0.09 bpp (achieving Y-PSNR of 36.3 dB).

to evaluate the HSDT performance. Besides being symmetric, there is no additional overhead in the coding stages since adaptation is based on previously coded blocks.

The results obtained with the HSDT codec are comparable with JPEG XR, when coding and decoding processes are performed under similar circumstances. Good results are also provided by the HSDT in terms of transform coding gain and energy concentration. The HSDT codec shows potential to provide much better results with the tuning of the quantization and entropy coding stages. Therefore, this work shows that there is still room for further improvement in image coding exploiting cross-level structural similarities.

Chapter 6

Future Works

As this thesis is a set of contributions on various topics related to image and video coding, each chapter can be read independently and is completed with its specific conclusions. Therefore, this final chapter aims to propose future works motivated by the several contributions of each chapter.

Chapter 2: It is interesting to further exploit the reduction of blocking artifacts with the use of lapped transforms. Additionally, other fast transforms, more efficiently matched to the signal covariances in the spatial and temporal dimensions could provide competitive results. The possibility of including some partial level of motion estimation and motion compensation could improve performance with limited impact on complexity.

Chapter 3: A better model that takes into account the dependence among pixels is necessary for the next stage of the rate allocation algorithm described in this chapter. The proposed optimization procedure can also be applied to the encoding of depth maps, resulting in increased coding efficiency. An additional benefit can be derived form a better fitting of the RxD curve, for instance, by computing different curves for I, P, and B frames.

Chapter 4: An attractive enhancement of the proposed method is its evolution towards the characterization of classes of block neighborhoods and the associated sets of average block filters. This modification would require a training stage and would have some impact on the system complexity.

Chapter 5: The computational load of the proposed adaptive transform scheme could be reduced by a preliminary computation of the standard deviation of the various blocks, so that trivial blocks would not be adaptively encoded. Improved performance could also be derived from various punctual modifications in the proposed system, such as, overlapped transforms, improved statistical modeling and training, and optimized quantization and entropy coding stages. Also promising is the application of the HSDT to video coding.

References

[1] Encoding.com Inc. Video Format Trends. *Available: www.encoding.com*, 2010.

[2] V. Testoni. *Sistema de Codificação de Vídeo Baseado em Transformadas Tridimensionais, Rápidas e Progressivas*. Dissertação de mestrado, Universidade Estadual de Campinas, 2007.

[3] V. Testoni and M. H. M. Costa. Fast Embedded 3D-Hadamard Color Video Codec. *Proceedings of the XXV Simpósio Brasileiro de Telecomunicações*, 2007.

[4] V. Testoni and M. H. M. Costa. 3D-Hadamard Coefficients Sequency Scan Order for a Fast Embedded Color Video Coded. *Proceedings of the International Conference on Signal Processing and Communication Systems*, 2007.

[5] V. Testoni and M. H. M. Costa. Three-Dimensional Transforms and Entropy Coders for a Fast Embedded Color Video Codec. *Proceeding of the Brazilian Symposium on Computer Graphics and Image Processing*, 2008.

[6] V. Testoni and M. H. M. Costa. Entropy Coders and 3-D Hadamard Coefficients Scan Order for a Fast Embedded Color Video Codec. *Elsevier Computers & Electrical Engineering*, 36(4):676–690, 2009.

[7] V. Testoni and M. H. M. Costa. Block-based 3-D Fast Transforms applied to an Embedded Color Video Codec. *Proceedings of the International Workshop on Telecommunicationsg*, 2011.

[8] G. Sullivan and S. Estrop. Video Rendering with 8-bit YUV Formats. Technical report, Microsoft Digital Media Division, 2003.

[9] A. K. Jain. *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ, USA, Prentice Hall, 1989.

[10] N. S. Jayant and P. Noll. *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ, USA, Prentice Hall, 1984.

[11] N. P. Sgouros, S. S. Athineos, and P. E. Mardaki. Use of an Adaptive 3D-DCT Scheme for coding Multiview Stereo Images. *Proceedings of the IEEE Symposium on Signal Processing and Information Technology*, pages 180–185, 2005.

[12] B. Furht, K. Gustafson, H. Huang, and O. Marques. An Adaptive Three-Dimensional DCT Compression Based on Motion Analysis. *Proceedings of the ACM Symposium on Applied Computing*, pages 765–768, 2003.

[13] Y. L. Chan and W. C. Siu. Variable Temporal-length 3D Discrete Cosine Transform Coding. *Proceedings of the IEEE Transactions on Image Processing*, 6(5):758–763, 1997.

[14] J. J. Koivusaari and J. H. Takla. Simplified Three-Dimensional Discrete Cosine Transform Based Video Codec. *Proceedings of the SPIE Multimedia on Mobile Devices*, pages 11–21, 2005.

[15] J. L. Takala, M. Gabbouj, and H. Chen. Variable Temporal Length 3D DCT-DWT Based Video Coding. *Proceedings of the IEEE International Symposium on Intelligent Signal, Processing and Communications*, pages 506–509, 2007.

[16] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-Complexity Transform and Quantization with 16-bit Arithmetic for H.26L. *Proceedings of the International Conference on Image Processing*, pages 489–492, 2002.

[17] R. K. W. Chan and M. C. Lee. 3D-DCT Quantization as a Compression Technique for Video Sequences. *Proceedings of the International Conference On Virtual Systems And Multimedia*, pages 188–196, 1997.

[18] R. K. W. Chan and M. C. Lee. Quantization of 3D-DCT Coefficients and Scan Order for Video Compression. *Journal of Visual Communication and Image Representation*, 8(4):405–422, 1997.

[19] J. G. Senecal, P. Lindstrom, M. A. Duchaineau, and K. I. Joy. An improved n-bit to n-bit reversible Haar-like transform. *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pages 371–380, 2004.

[20] F. C. Oliveira and M. H. M. Costa. Embedded DCT Image Encoding. *Proceedings of the International Telecommunications Symposium*, 2002.

[21] M. H. M. Costa and H. S. Malvar. Efficient Run-Length Encoding of Binary Sources with Unknown Statistics. *Proceedings of the Data Compression Conference*, pages 534–544, 2004.

[22] K. Shen and E. J. Delp. Wavelet Based Rate Scalable Video Compression. *Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, pages 109–122, 1999.

[23] J. M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 41(12):3445–3462, 1993.

[24] A. Said and W. A. Pearlman. A New Fast and Efficient Image Coded Based on Set Partitioning in Hierarchical Trees. *Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[25] B. Kim and W. A. Pearlman Z. Xiong. Low Bit Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT). *Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, pages 1374–1387, 2000.

[26] HHI Heinrich Hertz Institute. H.264/AVC reference sotware version JM 11.0. *Available: http://iphome.hhi.de/suehring/tml*, 2006.

[27] P. Merkle, K. Müller, and T. Wiegand. 3D video: Acquisition, Coding, and Display. *Proceedings of the International Conference on Consumer Electronics*, pages 127–128, 2007.

[28] H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. Goss, W. Culbertson, and T. Malzbender. The Coliseum Immersive Teleconferencing System. Technical report, HP Labs, 2002.

[29] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-Quality Video View Interpolation Using a Layered Representation. *Proceedings of the ACM SIGGRAPH*, 2004.

[30] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured Lumigraph Rendering. *Proceedings of the ACM SIGGRAPH*, 2001.

[31] C. Zhang and J. Li. Interactive Browsing of 3D Environment over the Internet. *Proceedings of the IEEE Visual Communications and Image Processing*, 2001.

[32] C. Zhang, Z. Yin, and D. Florêncio. Improving Depth Perception with Motion Parallax and Its Application in Teleconferencing. *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, 2009.

[33] ISO/IEC MPEG & ITU-T VCEG. Joint multiview video model (JMVM) 2.0. *JVT-U207*, 2006.

[34] ISO/IEC MPEG & ITU-T VCEG. Multi-view video plus depth (MVD) format for advanced 3D video systems. *JVT-W100*, 2007.

[35] G. Sullivan. Standards-based Approaches to 3D and Multiview Video Coding. *Proceedings of the SPIE Applications of Digital Image Processing*, 2009.

[36] C. Zhang and T. Chen. A Survey on Image-Based Rendering – Representation, Sampling and Compression. *EURASIP Signal Processing: Image Communication*, 19:1–28, 2004.

[37] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang. Multi-View Imaging and 3DTV. *IEEE Signal Processing Magazine*, 24(6):10–21, 2007.

[38] J. Zhang, M. M. Hannuksela, and H. Li. Joint Multiview Video Plus Depth Coding. *Proceedings of the International Conference on Image Processing*, pages 2865–2868, 2010.

[39] Q. Zhang, P. An, Y. Zhang, Q. Zhang, and Z. Zhang. Reduced Resolution Depth Compression for Multiview Video plus Depth Coding. *Proceedings of the International Conference on Image Processing*, pages 1145–1148, 2010.

[40] Y. Liu, Q. Huang, S. Ma, D. Zhao, W. Gao, S. Ci, and H. Tang. A Novel Rate Control Technique for Multiview Video Plus Depth Based 3D Video Coding. *IEEE Transactions on Broadcasting*, pages 562–571, 2011.

[41] J. Y. Lee, H. Wey, and D. Park. A Fast and Efficient Multiview Depth Image Coding Method Based on Temporal and Inter-View Correlations of Texture Images. *IEEE Transactions on Circuits and Systems for Video Technology*, 99, 2011.

[42] D. Florêncio and C. Zhang. View-Dependent Multiview Video Compression and Streaming for Immersive Tele-Conferencing. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[43] V. Testoni, D. Florêncio, and M. H. M. Costa. Optimizing QPs for Multiview Image Coding for Free Viewpoint Video. *Proceedings of XXVII Brazilian Symposium on Telecommunications*, 2009.

[44] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7), 2003.

[45] P. Merkle, A. Smolic, K. Müller, and T. Wiegand. Multi-view Video Plus Depth Representation and Coding. *Proceedings of the International Conference on Image Processing*, pages 201–2004, 2007.

[46] I. E. G. Richardson. *H.264 and MPEG-4 Video Compression*. John Wiley & Sons, England, 2003.

[47] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, USA, 1991.

[48] Microsoft Corporation. HD Photo Device Porting Kit. *Available: http://www.microsoft.com/whdc/xps/wmphoto.mspx*, 2010.

[49] S. Srinivasan, C. Tu, S. L. Regunathan, and G. J. Sullivan. HD Photo: a new image coding technology for digital photography. *Proceedings of the SPIE*, 6696, 2007.

[50] ITU-T Rec. T.832 and ISO/IEC 29199-2. JPEG XR Image Coding System - Part 2: Image Coding Specification. 2009.

[51] L. Zhang, W. Gao, Q. Wang, and D. Zhao. Macroblock-Level Adaptive Scan Scheme for Discrete Cosine Transform Coefficients. *Proceedings of the IEEE International Symposium on Circuits ans Systems*, pages 537–540, 2007.

[52] Y. Ye and M. Karczewicz. Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning. *Proceedings of the IEEE International Conference on Image Processing*, pages 2116–2119, 2008.

[53] Y. Tao, Y. Peng, and Z. Liu. More scanning patterns for entropy coding for H.264. *Proceedings of the Symposium on Intelligent Signal Processing and Communication Systems*, pages 490–493, 2007.

[54] V. Testoni, M. H. M. Costa, D. Kirovski, and H. S. Malvar. On the Adaptive Coefficient Scanning of JPEG XR/HD Photo. *Proceedings of the Data Compression Conference*, 2010.

[55] T. Tran, L. Liu, and P. Topiwala. Performance comparison of leading image codecs: H.264/AVC intra, JPEG 2000, and Microsoft HD Photo. *Proceedings of the SPIE*, 6696, 2007.

[56] F. de Simone, L. Goldmann, V. Baroncini, and T. Ebrahimi. Subjective Evaluation of JPEG XR Image Compression. *Proceedings of the SPIE*, 7443, 2009.

[57] H. Sohn, W. De Neve, and Y. M. Ro. Region-of-interest scrambling for scalable surveillance video using JPEG XR. *Proceedings of the ACM Internacional Conference on Multimedia*, 2009.

[58] W. De Neve, S. Yang, D. Van Deursen, C. Kim, Y. M. Ro, and R. Van de Walle. Analysis of BSDL-based content adaptation for JPEG 2000 and HD Photo (JPEG XR). *Proceedings of the Internacional Conference on Visual Information Engineering*, pages 717–722, 2008.

[59] C. Tu, G. Sullivan, and S. Srinivasan. Optimizing JPEG XR Tile Structure for Fast Local Access. *Proceedings of the Internacional Conference Consumer Electronics*, pages 379–380, 2010.

[60] C. Tu, S. Srinivasan, G.J. Sullivan, S. Regunathan, and H.S. Malvar. Low-complexity hierarchical lapped transform for lossy-to-lossless image coding in JPEG XR/HD Photo. *Proceedings of the SPIE*, 7073, 2008.

[61] T. Richter. Visual Quality Improvement Techniques of HDPhoto/JPEG XR. *Proceedings of the IEEE International Conference on Image Processing*, pages 2888–2891, 2008.

[62] D. Schonberg, G.J. Sullivan, S. Sun, and Z. Zhou. Perceptual encoding optimization for JPEG XR image coding using spatially adaptive quantization step size control. *Proceedings of the SPIE*, 7443, 2009.

[63] H. Koichi, T. Hiroshi, O. Hiroyuki, and N. Yukihiro. An architecture of photo core transform in HD photo coding system for embedded systems of various bandwidths. *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems*, pages 1592–1595, 2008.

[64] A. Maalouf and M. C. Larabi. Enhancing the Intra-prediction in JPEG-XR by Using Edge Information. *Proceedings of the Fifth International Conference on Signal-Image Technology & Internet-Based Systems*, pages 138–143, 2009.

[65] T. Richter. Deadzone Based Rate Allocation for JPEG XR. *Proceedings of the Data Compression Conference*, 2011.

[66] S. He and D. Kirovski. A novel visual perceptual model with an application to high-fidelity image annotation. *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pages 92–97, 2006.

[67] S. G. Mallat. A compact multiresolution representation: the wavelet model. *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, pages 2–7, 1987.

[68] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaires. *IEEE Transactions on Signal Processing*, pages 3397–3415, 1993.

[69] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[70] J. J. Y. Huang and P. M. Schultheiss. Block quantization of correlated Gaussian random variables. *Proceedings of the IEEE Transactions on Communications*, 11:289–296, 1963.

[71] J. L. Starck, E. J. Candes, and D. L. Donoho. The curvelet transform for image denoising. *Proceedings of the IEEE Transactions on Image Processing*, 11(6):758–763, 2002.

[72] I. W. Selesnick, R. G. Baraniuk, and N. G. Kingsbury. The dual-tree complex wavelet transform. *IEEE Signal Processing Magazine*, 22:123–151, 2005.

[73] W. Ding, F. Wu, X. Wu, S. Li, and H. Li. Adaptive directional lifting-based wavelet transform for image coding. *Proceedings of the IEEE Transactions on Image Processing*, pages 416–427, 2007.

[74] B. Zeng and J. Fu. Directional discrete cosine transforms for image coding. *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 721–724, 2006.

[75] H. Xu, J. Xu, and F. Wu. Lifting-based directional DCT-like transform for image coding. *Proceedings of the IEEE Transactions on Circuits and Systems for Video Technology*, pages 1325–1335, 2007.

[76] J. Xu, F. Wu, J. Liang, and W. Zhang. Directional lapped transforms for image coding. *Proceedings of the Data Compression Conference*, pages 142–151, 2008.

[77] D. H. Ballard. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Elsevier Pattern Recognition*, 13(2):111–122, 1981.

[78] B. Tao and B. W. Dickinson. Texture recognition and image retrieval using gradient indexing. *Journal of Visual Communication and Image Representation*, 11(3):327–342, 2000.

[79] A. Çarkacioglu and F. Yarman-Vural. Sasi: a new texture descriptor for content based image retrieval. *Proceedings of the IEEE International Conference on Image Processing*, pages 137–Ű140, 2001.

[80] A. Çarkacioglu and F. Yarman-Vural. Sasi: a generic texture descriptor for image retrieval. *Elsevier Pattern Recognition*, 36(11):2615Ű–2633, 2003.

[81] S. D. Bayrakeri and R. M. Mersereau. A new method for directional image interpolation . *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2383–2386, 1995.

[82] H. Jiang and C. Moloney. A new direction adaptive scheme for image interpolation . *Proceedings of the IEEE International Conference on Image Processing*, pages 369–372, 2002.

[83] V. Velisavljevic and R. Coquoz. Image interpolation with directionlets. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 837–840, 2008.

[84] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[85] ISO/IEC MPEG & ITU-T VCEG. Quantization offset matrices for fidelity range extensions. *JVT-L032*, 2004.

[86] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.