INSTITUT DE MICROTECHNIQUE
UNIVERSITÉ DE NEUCHÂTEL

Algorithms, VLSI Architectures, and ASIC Design for Image Compression Systems

Javier Bracamonte

THÈSE PRESENTÉE À LA FACULTÉ DES SCIENCES
POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

Dissertation typeset with LaTeX 2_ε by the author.

IMPRIMATUR POUR LA THÈSE

Algorithms, VLSI Architectures, and ASIC Design for Image Compression Systems

de M. Javier Bracamonte

UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. F. Pellandini (directeur de thèse),
H. Hügli, C. Piguet, M. Ansorge et M. Kunt (EPF Lausanne)

autorise l'impression de la présente thèse.

Neuchâtel, le 20 février 1998

Le doyen:

F. Stoeckli

To my mother
and the memory of my father

Abstract

The content of this dissertation extends over the domains of digital image compression, VLSI architectures, and ASIC design. The presentation is divided into two parts.

The first part deals with the image compression issue from a hardware implementation point of view. The main contributions in this section are the conception of a design methodology for the VLSI implementation of image and video compression algorithms, the design of a bit-serial VLSI architecture for a motion estimation algorithm, and a power management strategy for wireless image communications systems.

The second part focuses on image compression mainly at an algorithmic level, keeping particular emphasis on the computational complexity issues, with a view to low-power hardware implementations. There are two main contributions in this section. First, the design of an efficient bit-rate control method for a still image coding algorithm, and second, the conception of a low-complexity, adaptive block-size transform-based image/video coding algorithm.

⊢ ◇◇◇ ⊣

Résumé

Le contenu de cette thèse s'inscrit dans les domaines de la compression numérique d'images, des architectures VLSI, et de la conception de circuits intégrés pour des applications spécifiques. La présentation est divisée en deux parties principales.

La première partie aborde le sujet de la compression numérique d'images d'un point de vue réalisation matérielle des algorithmes de codage. Dans ce domaine les contributions proposées sont les suivantes: le développement d'une méthodologie de conception pour la réalisation VLSI des algorithmes de compression d'images et des séquences vidéo; la conception d'une architecture VLSI bit-sérielle pour un algorithme d'estimation de mouvement; et finalement, le développement d'une stratégie de gestion de puissance applicable aux systèmes de communications audiovisuelles sans fil pour la transmission d'images.

Dans la seconde partie, la compression d'images est traitée d'un point de vue purement algorithmique, avec un regard particulier sur la complexité de calcul en vue d'implantations matérielles sur des systèmes à faible consommation de puissance. Deux contributions principales sont proposées dans ce domaine. Premièrement, la conception d'une méthode efficace pour contrôler le taux de compression d'un algorithme standardisé de codage d'images. Deuxièmement, la conception d'un algorithme de compression d'image/vidéo adaptatif, ainsi qu'une stratégie de réalisation pour diminuer fortement la complexité de calcul de cet algorithme.

⊢ ◇◇◇ ⊣

Acknowledgments

There are a number of people and institutions that deserve credit for the successful achievement of this dissertation.

First of all, I am deeply grateful to Prof. Fausto Pellandini for giving me the opportunity to work under his supervision and for his encouragement and guidance. Wben times were challenging, his helping hand was always present. 1 was privileged to have made this research work with a person of such an extraordinary human and scientific value, as him.

My effusive thanks go to Michael Ansorge, for his genuine interest in my work, for his priceless availability, and for sharing his large scientific knowledge profusively. I am very grateful to him for his constant support, and for the comments and suggestions, that with no doubt improved the quality of this thesis.

I express my gratitude to Prof. Murat Kunt, Prof. Heinz Hügli, and Dr. Christian Piguet, for their time and their effort in evaluating this dissertation as members of the jury. Their positive feedback during the private examination is also largely appreciated.

I extend special thanks to Dr. Ulf Sjöström and Dipl. Ing. Ivan Defilippis for introducing me into the VLSI world. I certainly benefited in a great deal from their experience. I also thank Ulf for initiating me into a lot of extracurricular activities.

Dr. Jean Pierre Amann provided assistance in many aspects, including frequent advise on software and hardware issues related with my computer-working environment. His support is here gladly acknowledged. I equally wish to thank Dr. Francois Corthay, who made himself always available for my questions and with whom I had a lot of pleasure

to work with during the testing of integrated circuits.

I am pleased to acknowledge the assistance received by our system manager Heinz Burri. Heinz always solved my computer and network problems willingly and promptly. My thanks also go to our secretary Catherine Lehnherr for the efficiency and amiability of her assistance.

As a research and teaching assistant I had the opportunity to supervise several diploma and semester projects for students from the University of Neuchâtel and the Swiss Federal Institute of Technology (EPFL). These interactions gave place to multiple interesting technical discussions. Let them *all* find in these lines my appreciation for those fruitful collaborations; with a particular thanks to Patrick Stadelmann.

I express my gratitude to all my colleagues and ex-colleagues for providing support, for making the Institute of Microtechnology (IMT) a very pleasant place to work at, and for the organization of those memorable weekends of ski and fondue dinners. Exchanging ideas in technical and non-technical topics with Sara Grassi, Alain Dufaux, Christophe Berthoud, and Vincent Moser was always very pleasant.

The work reported in this dissertation resulted from research supported by the Swiss National Science Foundation, under Grants FN 2000-40'627.94 and FN 2000-47'185.96, and by the Laboratory of Microtechnology. The latter is shared between EPFL and the IMT of the University of Neuchâtel. Their support is here thankfully acknowledged.

I am indebted to the Swiss Federal Scholarship Commission for Foreign Students, for the fellowship they provided me, which set the start of this enriching academic and cultural experience. In particular I gratefully acknowledge the support from Dr. F. Ehrler, Mr. Martial Renaud, and from all the University of Neuchâtel's delegates to the Commission.

Finally, but most important, I owe my warmest thanks to my mother, who since I can remember, was persistently encouraging me to always make one step further.

FRANCISCO JAVIER BRACAMONTE HENRIQUEZ

Neuchâtel, Switzerland
December 1997

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To produce an image in digital form, an enormous amount of data is required. Due to the explosive proliferation of images in today's digital world [1], the prodigality of digital image data has three direct economic and technical implications: a) the storage of digital images demands a significant qnantity of memory media, b) the transference of digital images requires long transmission times, c) image processing applications call for a large amount of computational power.

In their canonical form (just after they have been digitized), two images with the same spatial and gray-level resolution require exactly the same number of data (bits) regardless of their information content. For example, an image of a cloudless sky would require the same number of bits as an image of a colorful and varied flower garden, even though the latter contains much more information than the former. Technically speaking, it is said that the former image contains a high degree of redundancy, in the sense that in exchange for some processing, the same image could equally be described by using less data.

The type of redundancy pointed out in the previous paragraph is only one among others [2]. Image coding—or compression—is the art or science of reducing or eliminating all possible kinds of redundancy in a digital image, so as to represent it using the least possible amount of data, while retaining most of the original information. The same definition extends to video coding, where an additional exploitable kind of redundancy is present, given the high degree of similarity between two

successive images in a video sequence.

Many image compression techniques have been reported in the literature [3–5]. In general, the original image is processed with the selected coding algorithm, to produce a compressed version. Depending on the application, the compressed data is either stored, or transmitted via a communication channel. Then, when the original image is required, the decoding algorithm is applied to the compressed version to reproduce an exact copy of the original image (lossless compression [6, 7]), or a very good replica (lossy compression). Indeed, to increase the compression efficiency, most of the coding algorithms introduce some noise (errors) in the decompressed image due to the execution of some sort of quantization. The majority of the image/video coding schemes perform a lossy compression; some of these methods are referred to below.

1.1 Image and Video Compression Techniques

Current image compression methods can be classified into two groups. Traditional *waveform-based techniques*, and the so-called *second generation techniques*. The former rely on a statistical model of the image data, while the latter are rather based on a structural image model. Among the most popular waveform-based techniques are DPCM [2–4], transform coding [8, 9], vector quantization [10, 11], subband/wavelet coding [12–14], and block-based fractal coding [15, 16]. Second generation techniques mainly identify contour/texture decomposition algorithms [17].

Video coding methods can be classified into three groups. *Three-dimensional waveform techniques* which can be viewed as a natural extension to the video data volume of the still image waveform-based methods [18, 19]. *Motion-compensated waveform techniques* which typically feature a waveform-based method operating in combination with a motion-compensated prediction scheme [20, 24, 25]. Finally, *model-based techniques* which include object-based and knowledge-based schemes [21, 22].

Depending on the connotation of its *model* or on its implementation

Standard	Example of applications	bit-rate
JPEG	Still images	0.3–2 bpp
MPEG-1	CD-ROM imagery	1.5 Mbps
MPEG-2	Digital TV	2–10 Mbps
MPEG-4	Video on the Internet	< 64 Kbps
H.261	Videoconferencing	64–1200 Kbps
H.263	Video over PSTN	5–64 Kbps

Table 1.1: Standards for image and video coding.

a given technique can be associated with several of the aforementioned classification groups.

1.2 Standards for Image and Video Compression

Table 1.1 shows some of the most popular standards for image and video compression with some typical applications and output bit-rate characteristics. JPEG is a coding standard [23] for still images. In some applications JPEG is used to code a sequence of moving images; each image being JPEG-coded independently of the others. This procedure, which is not standardized, is called Motion (or Moving) JPEG, or M-JPEG for short. The MPEG-1 standard [24] was proposed for coding video for a wide range of applications, but specially for storage and retrieval of compressed imagery on CD-ROM. The quality produced by MPEG-1 is not acceptable for broadcast applications; thus, the MPEG-2 standard [25] was proposed for digital television applications at 2–10 Mega bits per second (Mbps) or even higher bit-rates for HDTV. The MHEG[1] standard [26, 27] approved in 1995 defines a system-independent encoding of the structure information used for storing, exchanging, and executing multimedia and hypermedia information objects.

ISO's future standards will include among others, MPEG-4 [28,29] for very low bit-rate applications such as interactive audiovisual multimedia over mobile and PSTN, and MPEG-7, which will allow the de-

[1] A list of the acronyms and abbreviations commonly used in this dissertation is given in Appendix C.

scription, identification, and access to multimedia information.

ITU's recommendations include, the H.261 standard [30] which is used for low bit-rate applications such as videophone or videoconference on ISDN, while the H.263 standard [31] is suitable for very low bit-rate applications such as transmission of video over PSTN. An emerging H.263+ standard incorporates numerous enhancements and new features over H.263 and promises better results for very low bit-rate applications.

1.3 Economic Significance

The study of image and video compression techniques has become very popular, particularly during the last decade. A rush of research and commercial activity has recently developed in this field. A multitude of literature is being published on a regular basis, and a substantial part of signal and image processing journals and conferences is being dedicated to addressing this issue. One of the reasons for this success, is the current market potential of data compression technology.

In a last year's issue of *The Economist* [32] a known Silicon Valley executive is reported as having calculated that the introduction of the personal computer (PC) has caused the largest creation of wealth in history. Later, in the same paragraph, the article cites a computer industry analyst as adding that the Internet has the potential of becoming even bigger. Though the Internet is certainly not just compression, an efficient management of the transmission of images and specially video has made compression technology a cornerstone element not only of the Internet but also of any other image/video transmission network.

An interesting analysis is given in [1] on how information is nowadays marketed in an *atom*-based support, that is, to obtain information people buy atoms (e.g., newspapers, books, videotapes, audio-CDs, DVDs). With the emergence of digital information distribution networks, the tendency will be towards getting these kinds of goods and services directly in a matterless *bit*-form. Thus, most types of information will be bought and in many cases even consumed directly in a digital form. Again, for these networks, a good data compression strategy will

be necessary. Otherwise, among other consequences, long transmission and distribution times could occur; and time as they say, is money.

One final example is the imminent development of portable image communications systems, which as experts are equally announcing, will have one of the largest markets in the telecommunications industry. In this application as well, image compression will be of paramount importance.

1.4 Scope of the Dissertation

Even if its economic value has been latent for a while, the catalyst that brought image coding from the lab into *the real world* has been the impressive development of VLSI (Very Large Scale Integration) technology. In effect, though algorithms for data compression had been proposed well before the current PC era, they did not develop its current industrial interest, until the advent of efficient VLSI circuits that could cope with the large computational complexity of image/video processing applications [33]. Thus, VLSI technology has been a sine-qua-non condition for the current success of image compression, and as more and more complex coding algorithms are developed, this dependence will be maintained in the future.

This dissertation stretches over these two research fields: VLSI and image compression algorithms. The first part is focused on the overlap of both domains, while a second part deals with image compression issues mainly at an algorithmic level. The chapter distribution is shown graphically in Figure 1.1.

1.5 Organization of the Dissertation

A different chapter has been dedicated to expose each of the major contributions of this dissertation. After this introductory chapter, a methodology for the design of VLSI circuits for image and video coding applications is reported in Chapter 2. The different tasks of the design procedure are discussed along with a description of the involved software

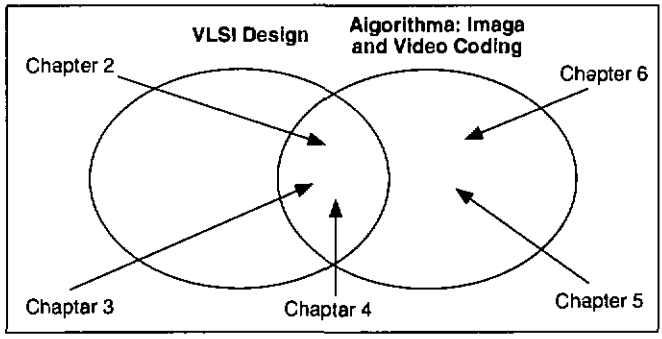


Figure 1.1: Scope and content distribution of this dissertation.

environments. The design of an ASIC for image compression is presented to illustrate the methodology. Chapter 3 presents an efficient VLSI architecture for a motion estimation algorithm. The selected bit-serial data processing associated with a particular implementation scheme produce excellent results in terms of input bandwidth reduction and silicon area. The result of an ASIC design based on this architecture is also reported. In Chapter 4 a strategy for saving power in portable devices for image communications is introduced. In this chapter the traditional trade-off between image quality and compression ratio is reformulated in terms of image quality vs. power consumption for particular applications.

The JPEG algorithm is a variable coding bit-rate method and in Chapter 5 a procedure for controlling the compression ratio produced by JPEG is introduced. This bit-rate control technique features an excellent accuracy, a low computational complexity, and its use does not degrade the quality of the decompressed images any further (i.e., there are no additional losses other than those inherent to JPEG, in opposition to previously reported schemes), to reach the target compression ratio.

Chapter 6 reports an adaptive block-size transform coding method and a scheme to significantly reduce the computational complexity of the algorithm. The results show the improvement on the compression efficiency with respect to the non-adaptive schemes. Finally, the conclusions and some possible extensions to the presented work, are given in

Chapter 7.

1.6 Major Contributions

The following is a list of the major contributions presented in this dissertation.

- A design methodology for the VLSI implementation of image and video compression algorithms.
- A bit-serial architecture for a motion estimation algorithm and an efficient implementation scheme for reducing the input bandwidth.
- A power management scheme for wireless image communication systems.
- A bit-rate control method for the JPEG algorithm.
- An adaptive algorithm for transform-based image and video coding and a strategy to reduce its computational complexity.

Chapter 2

Design Methodology for VLSI Implementation of Image and Video Coding Algorithms

2.1 Introduction

In order to produce a compressed version of an image or a video sequence, a significant amount of computational resources are required. For some applications, a general purpose personal computer would suffice to execute the coding/decoding algorithms. For others however—e.g., when the process must be performed in real time—the needed computational power might be so high that it can only be achieved by means of particular VLSI circuits. These circuits may correspond to special programmable video signal processors (VSP) of the last generation [34–37], or to application specific integrated circuits (ASICs) [38, 39]. The ASIC solution, towards which the methodology presented in this chapter is oriented, is essential for particularly demanding applications such as those featuring multimedia capabilities on miniaturized and/or portable equipment.

The process of materializing the mathematical description of an image compression algorithm into a chip implies the solution of a relatively complex problem. The evolution of the complete design process goes through several levels of abstraction [40, 41]; each level requiring the use of different methods and CAD technologies, whose intermedi-

ate results should facilitate the tasks of the next abstraction stage. An effective methodology is thus indispensable to orchestrate the multiple involved procedures in order to reduce time, effort and design errors, while constraining the final result for a particular feature such as high speed, small area or low power consumption.

In this chapter a methodology for the design of VLSI circuits for image and video coding applications is presented [42, 43]. The different phases of the design procedure are discussed, along with a description of the involved software environments. An example of an area efficient single-chip implementation of a JPEG coder is presented to illustrate the methodology.

2.1.1 Organization of the chapter

The remainder of this chapter is organized as follows. In Section 2.2 the methodology is introduced, that is, the different tasks, the order in which they are executed and the CAD tools are indicated. A brief description of the JPEG algorithm, which will be used as a case study, is presented in Section 2.3. High level modeling is described in Section 2.4, and a study of VLSI architectures is presented in Section 2.5. Bit-true level modeling is described in Section 2.6, and layout design is discussed in Section 2.7. Finally, the results are given in Section 2.8, just before a summary of the chapter in Section 2.9.

2.2 Methodology

Figure 2.1 shows the procedure stream and the supporting CAD tools of the methodology. Given an image or video coding algorithm, the different tasks involved in order to implement it in an ASIC form, start with a high level description of the algorithm using available software functions or routines from an ad-hoc developed toolbox. The algorithm is then decomposed into different modules, for each of which a VLSI architecture is designed. The functionality and definition of parameters of the VLSI architecture are studied by making models and simulations

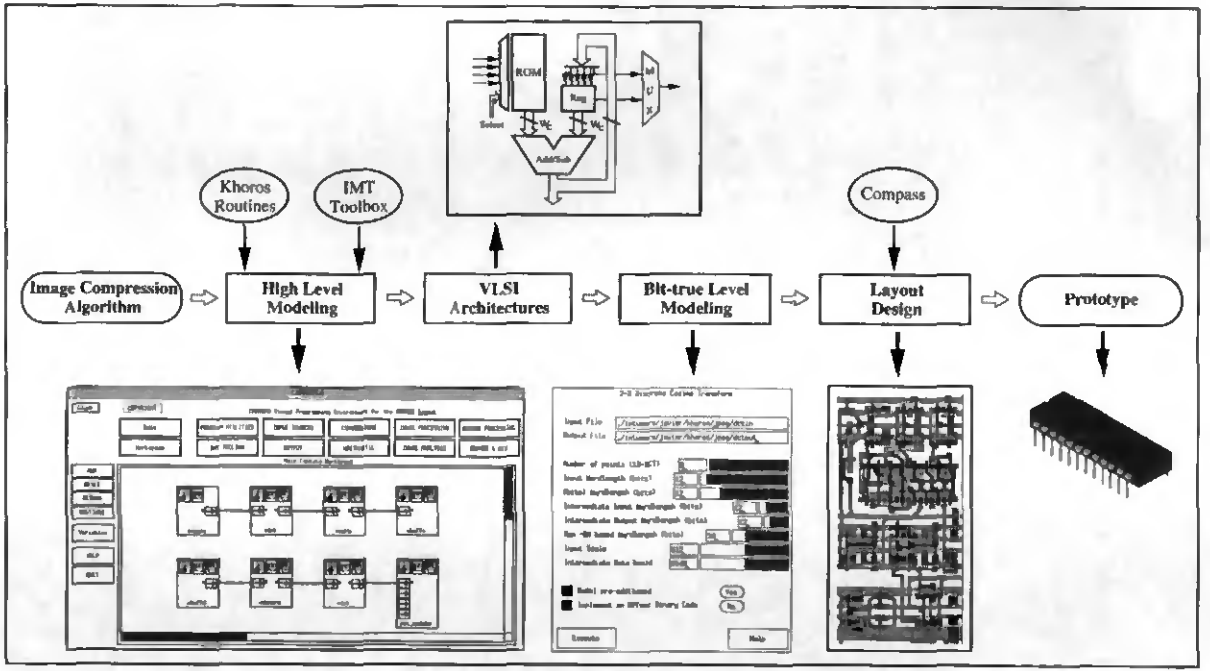


Figure 2.1: Methodology.

at the bit level. A layout for each of the bit-true level models is then produced, and finally, verification and testing of the resulting integrated circuit is performed.

The following sections describe each task of the methodology applied to develop an ASIC for the JPEG image compression algorithm.

2.3 JPEG

The Joint Photographic Expert Group (JPEG) algorithm defines an international standard for compression of continuous-tone still images. It specifies four modes of operation: sequential, progressive, lossless and hierarchical [44]. The sequential mode will be described in detail in the next section.

The progressive mode provides at successive scans, a decompressed version of an image, in such a way that each additional scan increases its reconstruction quality. In this manner, a first display allows a user to preview a rough version of the coded image; and if additional scans are requested, the amount of detail of the reconstructed image is increased progressively. Clearly, for storing/searching images in a database, this operation mode is the indicated one. Progressive JPEG files are also becoming increasingly popular on the Internet.

The lossless mode, as its name implies, allows the user to obtain a decompressed image in which each of its pixels has an identical value to the corresponding pixel in the original image. This is the only JPEG's mode of operation that is not based on a DCT. The underlying compression method is basically a classical 1-D or 2-D DPCM [2] followed by entropy coding.

The main feature of the hierarchical mode is its capability to provide reconstructed images at different spatial resolutions, i.e., decoded images at different heights and widths. This mode is appropriate when decompressing images in a multiresolution display environment.

Thus, rather than a single image compression method, JPEG defines a family of application-dependent coding schemes, from which the sequential mode is the most commonly implemented. The sequential

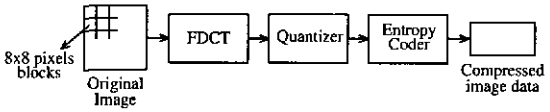


Figure 2.2: Baseline JPEG algorithm.

mode itself comprises two differents schemes: a *baseline system* and an *extended sequential system.*

The baseline (sequential) system is by far the most used among all the possible JPEG configurations, since it covers a wide range of applications. In fact, when writers refer to a coding system in the literature with the generic term JPEG, generally, it is the baseline implementation that is being implied (such implication is also made in this thesis). This is also the JPEG compression scheme that will be addressed in this chapter.

2.3.1 Baseline JPEG algorithm

The baseline sequential JPEG algorithm is depicted in Figure 2.2. The input image is subdivided into blocks of 8x8 pixels. Sequentially, each of these blocks is processed as follows,

- A forward Discrete Cosine Transform (DCT) operation transforms the 8x8 pixels from the original image into 8x8 coefficients in the frequency domain. The goal of this transformation is to decorrelate the original data and redistribute the signal energy among only a small set of transform (DCT) coefficients within the low frequency zone. Before applying the forward DCT, the value of the 64 pixels of the original block, is level shifted by -128 (i.e., 128 is subtracted from each pixel value).

- Based on a psychovisual analysis [45,46], a normalization array can be defined. Its purpose is to quantize those DCT coefficients that are visually significant with relative short quantization steps, while using large quantization steps for those coefficients which are less important. This large quantization-step, associated with the energy

packing effect of the transformation, results in general in the zeroing out of many DCT coefficients.

- Though the quantization of the DCT coefficients is the main mechanism of data compression (but also of information loss), additional data compression can be obtained by entropy coding the output of the quantizer. The entropy coding block in Figure 2.2 comprises three lossless operations.
  - *DC DPCM:* The value of the DC coefficient[1] is proportional to the average value (brightness) of the original block of pixels. Since the average brightness of two consecutive 8x8-pixel blocks is generally very similar, each DC coefficient is DPCM[2] encoded using the DC coefficient of the previous 8x8-pixel block as the prediction value.
  - *AC Run-length:* Many AC coefficients become zero after the quantization operation. Lining the quantized 8x8 elements of the 2-D DCT matrix up into a 64-element 1-D vector, by following a zigzag pattern [47], produces long sequences (or runs) of zero-valued coefficients. These long runs of zeros can be effectively abridged by using run-length coding [48].
  - *Huffman Coding:* Finally, the symbols produced at the output of both the DPCM and run-length operations are Huffman encoded [49]. Both DC and AC Huffman tables are defined respectively for the DC (DPCM-coded) and the AC (run-length coded) coefficients. While both Huffman and arithmetic coding are defined by the general JPEG standard for this last entropy coding stage, the baseline system requires Huffman coding only. This has two advantages: first, Huffman coding demands a less complex hardware implementation than arithmetic coding; and second, the arithmetic QM-coder suitable for JPEG is covered by

[1]The top-left element X(0,0), from the array **X** that represents the 2-D DCT coefficients, is referred to as the DC coefficient, the remaining 63 are referred to as the AC coefficients.

[2]Differential Pulse Code Modulation.

several patents; its commercial implementation, in consequence, requires the payment of license fees. It is reported in [50] that the use of arithmetic in place of Huffman coding improves JPEG's compression ratio by 5% to 10%.

To illustrate the algorithm, an array of 8x8 pixels, corresponding to a region of the eye in an image of a person's face, is given in Equation (2.1).

$$\mathbf{A} = \begin{bmatrix} 77 & 69 & 69 & 70 & 71 & 67 & 67 & 67 \\ 74 & 70 & 72 & 67 & 64 & 66 & 66 & 65 \\ 67 & 72 & 68 & 68 & 64 & 66 & 68 & 66 \\ 65 & 72 & 71 & 71 & 62 & 65 & 69 & 90 \\ 68 & 76 & 79 & 65 & 61 & 56 & 58 & 99 \\ 78 & 93 & 95 & 69 & 59 & 56 & 56 & 71 \\ 73 & 108 & 106 & 92 & 68 & 60 & 60 & 83 \\ 70 & 105 & 110 & 105 & 90 & 84 & 83 & 102 \end{bmatrix} \tag{2.1}$$

The result after the 2-D DCT has been applied to the array $\mathbf{A}$ (whose elements are previously offset by -128) is given in Equation (2.2). It can be seen that most of the energy is concentrated in only a few low frequency—i.e., the top left corner region—coefficients.

$$\mathbf{X} = \begin{bmatrix} -431 & 25 & 11 & -48 & -3 & -16 & 4 & -6 \\ -52 & -16 & 6 & 40 & 8 & 15 & 0 & 5 \\ 35 & 9 & -20 & 4 & -5 & 4 & -6 & 0 \\ -18 & 19 & 11 & -2 & 2 & -3 & 6 & 6 \\ 15 & -26 & 2 & -3 & 11 & -3 & 2 & -1 \\ -1 & 6 & -3 & -2 & -1 & 1 & -5 & -4 \\ -1 & 1 & -4 & 9 & -4 & 2 & -1 & 3 \\ -2 & -1 & 0 & -4 & 4 & -4 & -1 & -1 \end{bmatrix} \tag{2.2}$$

The quantization of the 2D-DCT coefficients is carried out by dividing one-to-one, each element of the matrix $\mathbf{X}$ by the corresponding element of a normalization array, which in this example is given by the matrix $\mathbf{N}$ in Equation (2.3).

$$\mathbf{N} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{2.3}$$

The quotients of these 64 divisions are rounded to their nearest integers, giving as a result the matrix **Q** below.

$$\mathbf{Q} = \begin{bmatrix} -27 & 2 & 1 & -3 & 0 & 0 & 0 & 0 \\ -4 & -1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.4}$$

Reordering the elements of the matrix **Q** by following the zigzag pattern $(0, 1, 2, \ldots, 63)$ depicted in Figure 2.3 yields the 64-element vector z,

$$\mathbf{z} = [-27 \;\; 2 \;\; -4 \;\; 2 \;\; -1 \;\; 1 \;\; -3 \;\; 0 \;\; 1 \;\; -1 \;\; 1 \;\; 1 \;\; -1 \;\; 2 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; -1 \;\; \underbrace{0 \ldots 0}_{44}] \tag{2.5}$$

The value of the DC coefficient of the previous transformed and quantized 8x8-pixel block was found to be -25. The DPCM operation over the DC coefficients, simply substitutes the first element of the vector z by -2 [= -27 - (-25)], becoming

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure 2.3: Zigzag pattern reordering.

$$\mathbf{d} = [-2 \;\; 2 \;\; -4 \;\; 2 \;\; -1 \;\; 1 \;\; -3 \;\; 0 \;\; 1 \;\; -1 \;\; 1 \;\; 1 \;\; -1 \;\; 2 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; -1 \;\; \underbrace{0 \ldots 0}_{44}] \tag{2.6}$$

A zero run-length coding of the AC coefficients in vector **d** results in a sequence of pairs of values $(run, coeff)$ given by the expression in Equation (2.7). The first value, *run*, indicates the number of zeros that precedes the second value, *coeff*, in the sequence represented by elements of vector **d**. The DC coefficient is not concerned by this operation, a $run = 0$ has been associated to its value in Equation (2.7) just to allow a generalization of the explanations that follow below.

$$\mathbf{r} = [(0,-2) \;\; (0,2) \;\; (0,-4) \;\; (0,2) \;\; (0,-1) \;\; (0,1) \;\; (0,-3) \;\; (1,1) \;\; (0,-1) \;\; (0,1) \;\; (0,1) \;\; (0,-1) \;\; (0,2) \;\; (5,-1) \;\; \text{EOB}] \tag{2.7}$$

The symbol EOB stands for End-Of-Block. Since the compressed data of all the 8x8-pixel blocks which make up the complete input image are concatenated into a single bit stream; an EOB symbol allows the decoder to detect the boundaries of each 8x8-pixel block. In this case EOB is also implicitly indicating that the coefficients following the fourth -1 in Equation (2.6) are all zeros.

Though a statistical analysis could generate an (optimal) Huffman code for each possible combination of values of $(run, coeff)$, that would

produce a Huffman table with a too large number of entries. By representing *coeff* in a different format, the number of Huffman codes can be reduced. Thus, the value of *coeff* is broken down into two components: category (*cat*) and complement (*cmp*). Table A.1 in Appendix A indicates the different ranges for the values of *coeff* defined by JPEG, both for the (DPCM-ed) DC and AC coefficients. Each range is identified with a category number *cat*. For the AC coefficients, *coeff* can never be zero, therefore the appearance of the Non-Available (N/A) symbol in Table A.1.

The value of *cat* only specifies a range of values. In order to completely specify *coeff* a complementary number *cmp* is required. The binary representation of *cmp* is given by the least significant *cat*-bits of *coeff*, that is, the number of bits of *cmp* is given by the value of *cat*. Furthermore when $coeff < 0$, the value of *cmp* is decremented by one. This last operation is needed to represent without ambiguity both positive and negative number, in the same range of magnitude, with a same number of bits.[3]

Following these manipulations on *coeff*, the pairs (*run,coeff*) on Equation (2.7) can be rewritten as (*run/cat,cmp*), in which by expressing *cmp* as a binary number yields,

$$\begin{aligned} c = [&(0/2,01)\ (0/2,10)\ (0/3,011)\ (0/2,10)\ (0/1,0) \\ &(0/1,1)\ (0/2,00)\ (1/1,1)\ (0/1,0)\ (0/1,1) \\ &(0/1,1)\ (0/1,0)\ (0/2,10)\ (5/1,0)\ \mathrm{EOB}] \end{aligned} \tag{2.8}$$

The last part of the JPEG's entropy coding operation consists in looking a binary code up in a Huffman table for each of the *run/cat* combinations in Equation (2.8). For this example the JPEG Huffman tables reported in Annex K of the JPEG standard document [23] has been used; part of these tables are reproduced on Tables A.2 and A.3 on Appendix A. This operation generates the following bit stream,

[3] In two's complement representation, with n bits, it is possible to represent integers in the asymmetric-to-zero range $[-2^{n-1}, 2^{n-1} - 1]$.

01101/0110/100011/0110/000/001/0100/11001/000/001/001/
000/0110/11110100/1010.

where for readability of the example a slash has been inserted between the binary codes associated with each of the elements of the vector c. The last code, namely"1010", is the binary code that corresponds to the EOB symbol.

The final result indicates that the original 8x8-pixel block requires 62 bits to be coded. The representation of each pixel of matrix **A** in Equation (2.1) requires 8 bits, which makes a total of 512 bits to represent the complete block. JPEG is thus coding this block with a compression ratio[4] of $512/62 = 8.26$.

2.4 High Level Modeling

A modular high level implementation of the JPEG algorithm is shown in Figure 2.4. The top and bottom flowgraphs represent the encoder and decoder respectively. The DPCM and run-length coding/decoding operations described in the previous section are rather simple and were merged into a single module with the Huffman coder/decoder (i.e., *vhuffc/vhuffd*). From an image processing point of view, this high-level model is an application program on its own. For good-quality reconstructed images, typical compression ratios are around 10; this figure varies depending on the image activity (i.e., the complexity of the scene) and on its spatial resolution.

In the high level model, each process of the algorithm was implemented with the C programming language, using floating and integer point precision for the arithmetic operations. No concerns were raised at this point on issues as to how the operations will be carried out by the hardware (e.g., hardware multiplexing, parallel processing, bit-serial/parallel architectures, etc.). With respect to our methodology, developing a high level model of an algorithm allows us to have both a reference for the application and a programming framework to support

[4]The compression ratio is defined as the quotient of the number of bits of the uncompressed image divided by the number of bits of the compressed image.

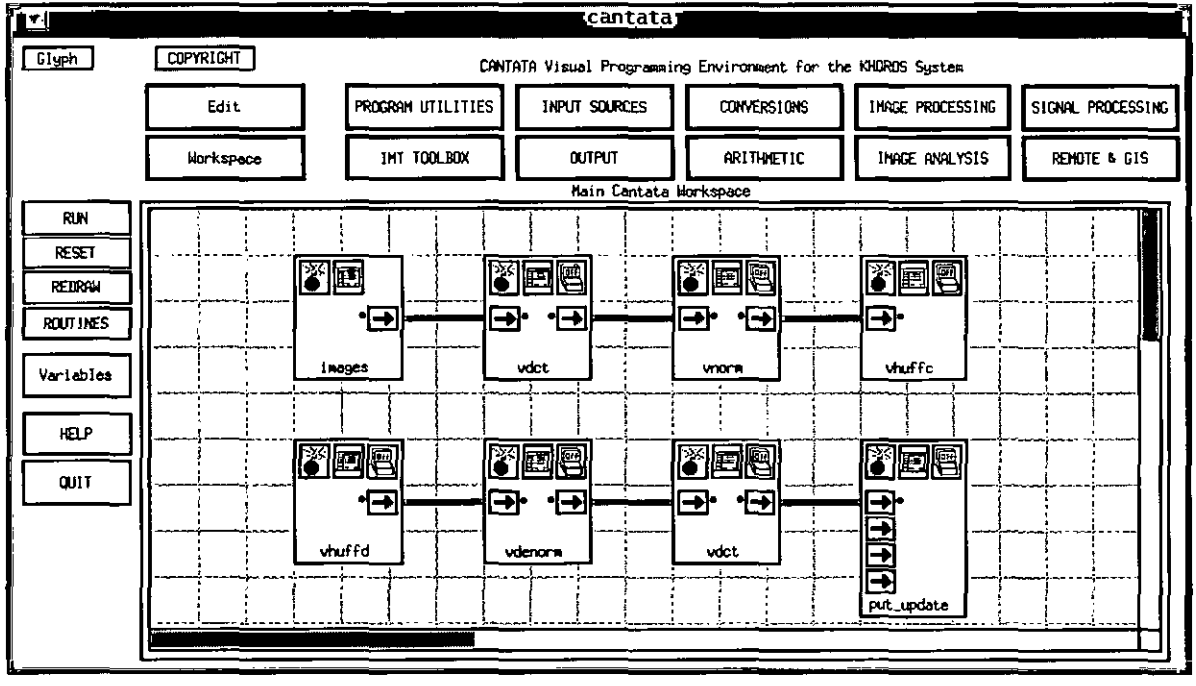


Figure 2.4: High level implementation of JPEG in Khoros.

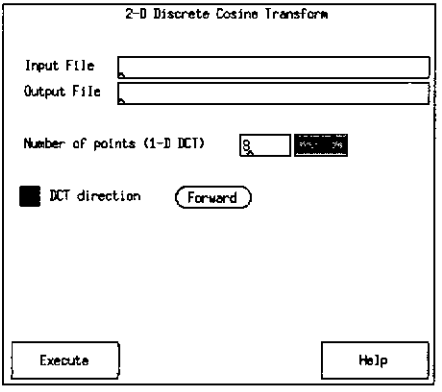


Figure 2.5: GUI of the 2-D DCT.

the bit-true level modeling (Section 2.6).

The Khoros software [51, 52] has been chosen as the environment for developing the high and bit-true level modeling. Due to its user-friendliness and visual programming capability, it permits a rapid evaluation of the results of a given configuration. Each glyph (or icon) in Figure 2.4 corresponds to a C program that has been converted into a Khoros routine. The glyph named *images* supplies the file of the image to be coded. The glyph *put_update* is used to display the decompressed image. These two routines are part of the original Khoros software.

The graphical user interface (GUI) of the routine *vdct* is shown in Figure 2.5. Though JPEG specifies a fixed block size of 8x8 pixels, the number of points of the DCT has been left as a parameter for experimental purposes. The same GUI is used to switch between the forward and inverse 2-D DCT.

The GUI of the routine *vnorm* is shown in Figure 2.6. It performs the quantization of the 2-D DCT coefficients with a normalization array that is user selectable. By default the normalization array given in Equation (2.3) has been programmed; it corresponds to the array proposed in Table K.1 in Annex K of reference [23]. The *Scale Factor* parameter allows the user to control the compression ratio. The higher the setting

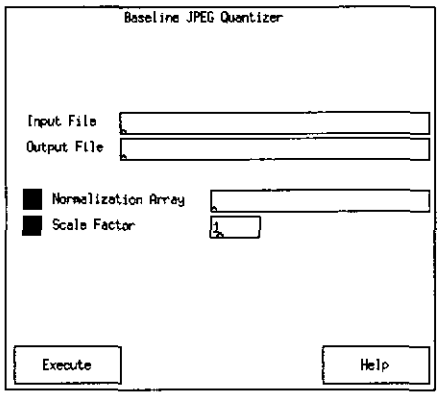


Figure 2.6: GUI of the quantizer.

of this scale factor the lower the amount of the resulting compressed image data. This kind of improvement of the compression ratio (CR) is made to the detriment of the quality of the reconstructed image. By default, *Scale Factor* is set to 1, which offers a good compromise: image quality/CR.

The *vhuff* routine implements the DC-DPCM, the AC run-length and the Huffman coding; its GUI is shown in Figure 2.7, as part of a toolbox baseline JPEG operations menu. The user is given the possibility of selecting a Huffman table customized to the application. By default, the Huffman codes proposed in Annex K of reference [23] have been programmed, an extract of these tables are reproduced on Tables A.2 and A.3 in Appendix A.

As the number of new developed Khoros routines grows, it is convenient to integrate them into a toolbox. This provides a software structure that is easier to reuse, maintain, document, and distribute [53]. For this purpose the IMT Toolbox, referred to in Figure 2.1 and whose integration into the Khoros main interface is shown on Figure 2.4, was developed in the frame of this thesis. It contains the JPEG related routines shown in Figure 2.4, and several routines for other applications on digital image processing.

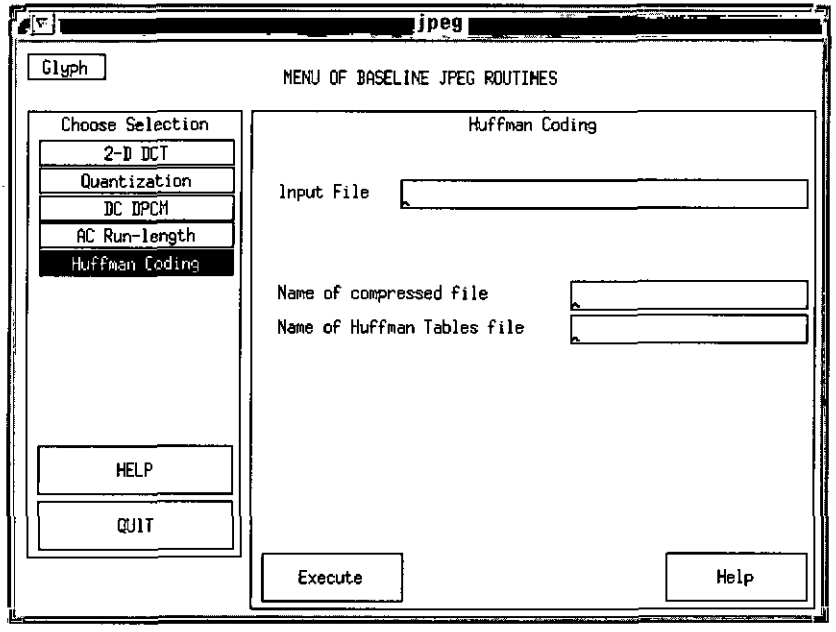


Figure 2.7: GUI of the entropy coder.

2.5 VLSI Architectures

To introduce this section, it is worth making a point regarding the design flowgraph in Figure 2.1; although it is not explicitly shown in this flowgraph, there is a bidirectional interaction among the different processes. When a proposed scheme turns out not to be a good solution, the designer should go back and forth in the design process while exploring other approaches. The previous remark is particularly pertinent to the design of the VLSI architectures. Thus, the reader should be aware that the solutions presented in this section are the result of several evaluations and trade-offs.

Taking into consideration the constraints of the target application, a VLSI architecture for each operator in Figure 2.2 is to be developed. As has been pointed out before, given their associated high data through-

put, image and video coding applications require particular hardware implementations that operate at high clocking frequencies. If the compression circuits are intended to be used in portable equipment, then power consumption is also of paramount importance.

Even though the relentless advance in the development of new semiconductor technologies makes feasible the production of devices that run faster or consume less power, a good architecture is still essential to achieve the specifications of a given application. Low power VLSI for image processing applications is currently a very active research topic [54–56], and though it was an issue that was considered during the design of the architectures below, the main objective remained the realization of high speed and small silicon area circuits.

Bit-serial architectures present several advantages [57–59], a) the size of bit-serial structures is considerably smaller than their bit-parallel counterparts; b) wiring, which in a VLSI circuit requires a large part of the silicon area, is dramatically reduced, since a single wire is required for each input word; c) communication between bit-serial modules is easier to implement, and d) bit-serial architectures lead to tightly pipelined structures at the bit-level, which implies that a maximum clock-rate can be achieved. Thus, whenever it was possible, we have chosen a fully pipelined bit-serial approach.

In the following paragraphs we describe the VLSI architecture for each of the main modules of the encoder in Figure 2.2.

2.5.1 FDCT

The forward 2-D DCT appropriate for JPEG is defined in [23] as:

$$X_{uv} = \frac{1}{4} C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} x_{ij} \cos[(2i+1)u\pi/16] \cos[(2j+1)v\pi/16]; \quad (2.9)$$

for $u, v = 0, 1, \ldots, 7$, where $C_u, C_v = 1/2$ for $u, v = 0$; $C_u, C_v = 1$ otherwise. X_{uv} represents the value of the 2-D DCT coefficient at the point (u, v) in the frequency domain, and x_{ij} represents the value of the image pixel at the point (i, j). Given that the 2-D DCT is separable, it can

be reformulated as two successive 1-D DCTs: the first one being applied to the rows of the block of 8x8 pixels, whereas the second is applied to the columns of the resulting matrix (or to the rows of its transpose) [9]. This decomposition leads to a simpler hardware implementation.

Using matrix notation, the N-point 1-D DCT $\mathbf{X}$, of an input vector $\mathbf{x}$, can be expressed as:

$$\mathbf{X} = \mathbf{A}\,\mathbf{x}, \tag{2.10}$$

where $\mathbf{A}$ represents the cosine transformation matrix. For $N = 8$ (as required by the JPEG standard) each element of $\mathbf{X}$ is obtained by means of an 8-point inner product of the following form:

$$X_k = \sum_{i=0}^{7} A_{ki}\, x_i, \quad \text{for } k = 0, 1, \ldots, 7. \tag{2.11}$$

Distributed arithmetic [60] is appropriate for the calculation of inner products. It has the interesting property of circumventing all the multiplications, which are elegantly replaced by the less complex addition and shift operations. Its principle is described in the following paragraphs.

Let us consider the evaluation of the following inner product:

$$y = \mathbf{a}\,\mathbf{x} = \sum_{i=0}^{N-1} a_i\, x_i, \tag{2.12}$$

where $\mathbf{x}$ and $\mathbf{a}$ denote an input signal vector and a fixed coefficient vector, respectively.

If x_i is represented as a W_d bits binary number in 2's complement form, then it can be expressed as:

$$x_i = \sum_{j=1}^{W_d-1} x_{ij} 2^{-j} - x_{i0}, \tag{2.13}$$

where x_{ij} represents the jth bit of x_i. Substituting Equation (2.13) into (2.12) and interchanging the order of the summations, we obtain:

$$y = \sum_{j=1}^{W_d-1} \left[\sum_{i=0}^{N-1} a_i\, x_{ij} \right] 2^{-j} - \sum_{i=0}^{N-1} a_i\, x_{i0}. \tag{2.14}$$

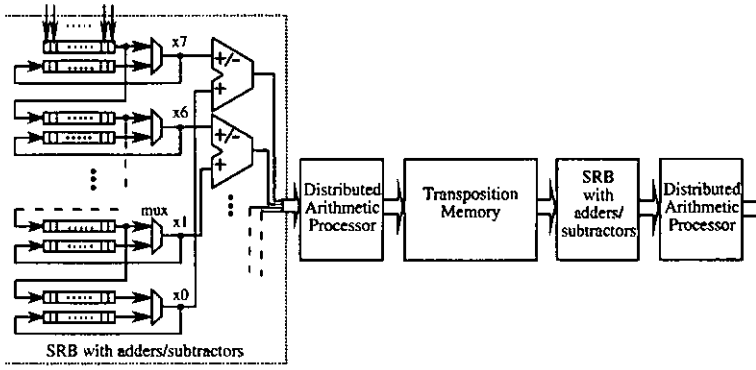


Figure 2.8: Architecture of the 2-D DCT.

Let us define F_j as:

$$F_j = \sum_{i=0}^{N-1} a_i x_{ij}, \tag{2.15}$$

since the coefficients a_i are fixed, and x_{ij} is a binary number, then any evaluation of F_j must be a number among only 2^N possible values. In a hardware implementation, these 2^N values can simply be stored in permanent memory.

By substituting Equation (2.15) into (2.14), the original expression of the scalar product of Equation (2.12) can be rewritten as:

$$y = \sum_{j=1}^{W_d-1} F_j 2^{-j} - F_0, \tag{2.16}$$

which can be implemented by a ROM for storing the values of F_j, an accumulator to implement the sum, and a shifter for executing the multiplications by 2^{-j}.

The architecture of the 2-D DCT is shown in Figure 2.8. Each 1-D DCT is executed by a single, multiplexed, distributed arithmetic processor (DAP) [61], whose architecture is shown in Figure 2.9. For the computation of each of the eight coefficients X_k, in Equation (2.11), the input vector $\mathbf{x}$, is applied eight consecutive times to the DAP. Each coefficient is computed by using a different lookup table. Thus, the

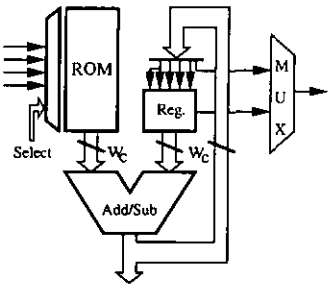


Figure 2.9: Distributed arithmetic processor.

ROM depicted in Figure 2.9, can be viewed as composed of eight different smaller ROMs which are chosen with the signal *Select*. The shift register bank (SRB) in Figure 2.8 provides the storage and sequential dataflow required for the repeated application of the input vector.

It happens that the cosine transformation matrix in Equation (2.10) presents a symmetry on its even rows and an antisymmetry on its odd rows. This implies that Equation (2.11) can be rewritten as:

$$X_k = \sum_{l=0}^{3} A_{kl} \left(x_l \pm x_{8-l} \right), \quad \text{for } k = 0, 1, \ldots, 7 \tag{2.17}$$

where the plus sign is used for the even values of k, and consequently, the minus sign for the odd values. The importance of Equation (2.17) is evident: the number of multiplications has been halved. In terms of the DAP's hardware, and in accordance with Equation (2.15), this means that the size of the DAP's ROMs in Figure 2.9, can be reduced from 2^N to $2^{N/2}$ words. For $N = 8$, eight small ROMs of 16 (instead of 256) words each, are required. Due to this significant memory saving, Equation (2.17) was retained for implementation; in conformity with this equation, the adders/subtracters in Figure 2.8 perform the inter-terms additions and/or subtractions.

2.5.2 Quantizer

The second operation of the JPEG coder is the quantization of the 2-D

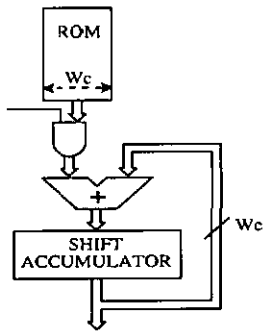


Figure 2.10: Architecture of the quantizer.

DCT coefficients and it is defined as: $Cq_{ij} = round(C_{ij}/N_{ij})$ for $i, j = 0, 1, ..., 7$, where C_{ij} denotes the ij-th element of an 8x8 DCT coefficients matrix, whereas N_{ij} denotes the ij-th element of an 8x8 normalization matrix. Reformulating the previous equation as $Cq_{ij} = round(C_{ij} * (1/N_{ij}))$ allows the replacement of the circuitry of a divisor with that of a simpler multiplier. The VLSI architecture of the quantizer is shown in Figure 2.10. Since the output of the 2-D DCT processor is bit-serial, a serial-parallel multiplier was implemented. The parallel input to this multiplier being the output of a ROM that contains the inverse of the normalization coefficients.

2.5.3 Entropy coder

The last operation of the baseline JPEG algorithm is entropy coding. Its goal is to increase the compression performance of the encoder by taking advantage of the statistics from the data at the output of the quantizer. The architecture of the entropy coder is shown in Figure 2.11. The zigzag reordering, required prior to the run-length coding, is executed by writing and reading a RAM with two different address sequences. The DPCM and run-length operations are implemented by means of a simple subtracter and a zero detection counter, respectively; plus additional control logic and registers. A ROM contains the Huffman codes; since they are inherently of variable length (minimum 2 bits and maximum 16

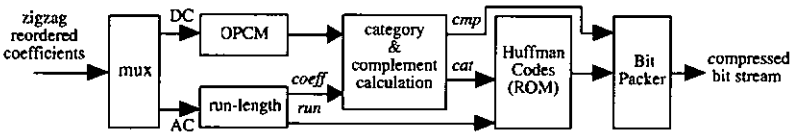


Figure 2.11: Architecture of the entropy coder.

bits, for the chosen Huffman table), a bit packer [62] has been added at the output of the circuit. The function of the bit packer is to concatenate consecutive codewords in order to output, at irregular intervals, fixed-length words.

2.6 Bit-true Level Modeling

The motivation for implementing a bit-true level model, is to assure the functionality of the architectures, the scheduling, the resource allocation and to find the optimum wordlength of the signals and coefficients of the algorithm.

Based on the programming framework of the high level implementation, a bit-true level model of the baseline JPEG algorithm was built. Each process of the algorithm was again implemented with C code and converted into a Khoros routine. However, unlike during the high-level modeling, now the arithmetic operations are all carried out in binary arithmetic, modeling accurately the same processing and dataflow as would be executed by the corresponding processors and architectures described in Section 2.5.

The GUI corresponding to the bit-true level model of the 2-D DCT is shown in Figure 2.12. Both rounding and truncation can be modeled. *Input Wordlength* refers to the number of bits per pixel required at the input of the 2-D DCT circuit. The original number of bits of an input image (e.g., 8 bits for an image of 256 gray levels) might be increased by padding trailing or leading zeros, to comply with the timing of the control signals in bit-serial structures.

Output Wordlength refers to the number of bits required to represent the 2-D DCT coefficients. As pointed out in Section 2.5.1, the

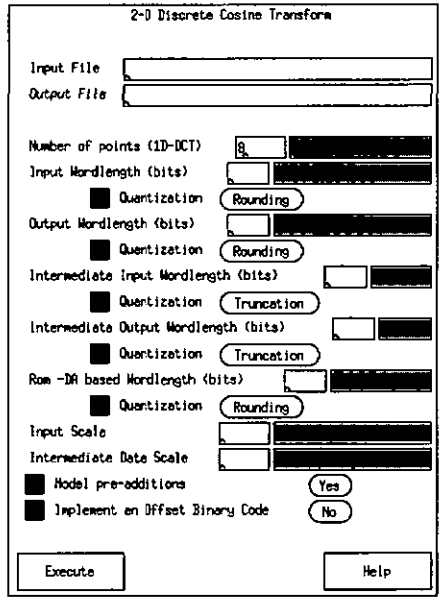


Figure 2.12: GUI of the 2-D DCT bit-true level model.

2-D DCT is computed by means of two successive 1-D DCTs. The *Intermediate* parameters in Figure 2.12 refer to the data in between the two 1-D DCTs. The wordlength of the DAP's ROM refers to the number of bits required to represent the lookup table coefficients given by Equation (2.15). In Figure 2.9 this wordlength is denoted by W_c.

The *Scale* parameters are required to fix the binary point to the same bit-position for all the data registers. In general the scale factors are chosen to be a positive integer power of 2, in order to facilitate the implementation by means of simple shifts.

The *Model pre-additions* parameter lets the user decide whether the implementation of Equation (2.11) or (2.17) should be modeled. As corroborated by the appearance of the adder/subtracters in Figure 2.8, this option was chosen to be set to *Yes*, in the presented design.

The Offset Binary Coding (OBC) technique allows a binary table to be reduced from 2^N to 2^{N-1} elements, as explained in [58]. Though

this option was not selected for our circuit (the reduction of a DAP's ROM size from 16 to 8 words was not very significant compared to the required logic and control overhead), its model was implemented solely for experimental purposes.

For the 2-D DCT circuit, the optimum DAP's data- and ROM-wordlength found was 12 and 10 bits respectively. The intermediate input and output wordlength was evaluated to 12 bits. Simulations showed no meaningful difference in the results between using rounding or truncation. Truncation mode was thus used due to its simpler implementation.

The simulations of the bit-true level model of the quantizer circuit gave as a result a ROM wordlength value of 9 bits and an input/output data wordlength of 12 bits. No particular bit-true level model was developed for the entropy coder. In effect, the Huffman coder involves mainly logical operations that had already been implemented (and modeled) in the high-level Huffman model. Furthermore, the result of the operations related to the DPCM and run-length coding can be accurately determined without any further low-level simulation.

2.7 Layout Design

In the context of this chapter, by layout design we refer to the translation of the architectures into a CMOS circuit which can be characterized by its area, speed, power consumption, and so on. Several software environments exist on the market today, to design circuits at any abstraction level (i.e., processor, register, gate or transistor level) [63–65]. In our case, we have used the Compass environment [63], which contains a set of tools that support the designer during the different stages of the circuit development.

When a module of an architecture is highly regular [66], the layout editor tool was used for developing full custom modules. The layout editor allows the design of circuits at the transistor level. For highly regular structures, this does not penalize the time for development since only a few cells have to be designed. On the other hand, minimum silicon

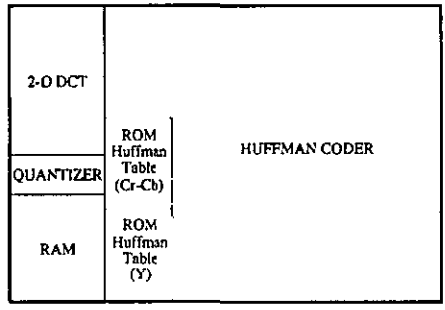


Figure 2.13: Floorplan of the JPEG circuit.

area, minimum power consumption and high clock frequency circuits are achieved.

For less regular structures, the standard cell approach was applied. A cell library containing a collection of logic and arithmetic circuits (e.g., inverters, ANDs, NORs, latches, flip-flops, adders) are available to the user. By selection, arrangement and interconnection, the circuit designer is able to build circuits of higher complexity. The Logic Assistant tool of Compass is well suited to this task.

Cell compilers were also used during the datapath development of the JPEG circuit, mainly to generate the RAM for matrix transposition in Figure 2.8, and the ROM containing the Huffman tables [67] of the entropy coder in Figure 2.2.

At each hierarchical level of the design, extensive simulations were carried out with both the Mixed-Mode and SPICE simulators. This allows verification of the functional correctness of the circuit, and the extraction of characteristics such as the maximum attainable working frequency.

2.8 Results

The floorplan and the layout of the JPEG coder circuit are shown in Figures 2.13 and 2.14 respectively. The area of the chip is 4.6 x 3.1 $mm^2 \approx 14.5\ mm^2$. It was implemented in the 1.2μm 5V CMOS CMN12

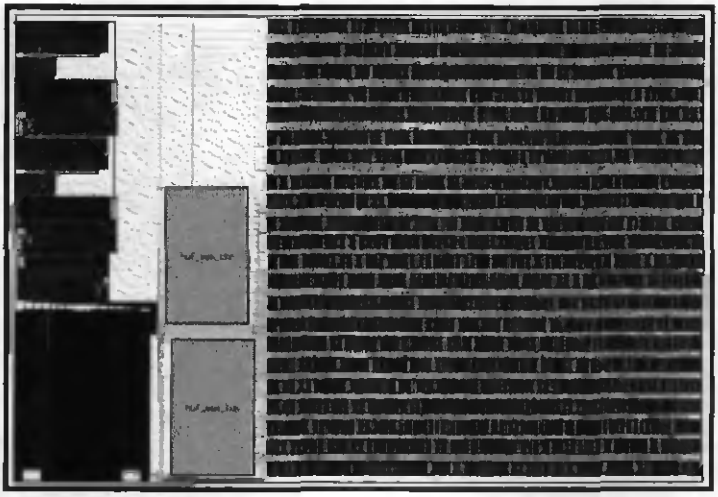


Figure 2.14: JPEG coder circuit.

process from VLSI Technology Inc.

On the top-left part of the layout are located the elements corresponding to the architecture shown in Figure 2.8, namely the shift register bank (SRB), the adders and subtracters, and the DAP. This 1-D DCT structure is repeated once in order to complete a 2-D DCT, as described in Section 2.5. The circuit of the quantizer is located just below the 2-D DCT module. All the previous circuits were built with full-custom layouts, a fact that is reflected by their very small area. For their construction, the reutilization of the basic layout cells was efficiently maximized.

The RAM for executing the matrix transposition and the zigzag reordering is located at the bottom-left corner of the circuit. The Huffman coder was realized by using the standard cell approach. It corresponds to the regular structure composed of rows of cells in the right part of the layout. The two blocks in the middle of the layout correspond to the ROMs containing the Huffman codes tables (DC and AC tables both for the luminance Y, and for the chrominance Cr-Cb).

2.8.1 Applications for video coding

At a clock frequency of 36 MHz, this JPEG circuit is capable of processing 25 CIF (Common Intermediate Format: 352 x 288 pixels) images per second. Thus, it is very suitable for performing Motion JPEG (M-JPEG) or for the non-recursive path of the H.261 [30] low-bit rate video coder.

M-JPEG, though formally is not a coding standard, represents one practical solution for DCT-based video coding. That is, if an available JPEG circuit operates fast enough, then one could use it to compress a video sequence by coding each video frame as an independent still image.

A more effective video coding system can be seen as a still coding scheme with an additional operator that reduces the frame-to-frame redundancies. Accordingly, motion compensation is a very well-suited technique for reducing the interframe redundancies particularly related to the movements of objects across consecutive frames in a video sequence. The current frame plus the motion information, is used to predict the immediately following frame(s); then, the prediction error frame—the difference between the actual frame and the prediction frame—is encoded as in a still image compression scheme.

The estimation of the motion vectors is the most computationally intensive operation in a video coder (e.g., more than 80% of the operations in an MPEG-2 encoder are estimated to be dedicated uniquely to the motion estimation). Thus, this operator deserves particular attention, specially for hardware implementation. In accordance with the architectures discussed in Section 2.5, a bit-serial architecture [68] and an ASIC design for a motion estimation algorithm will be described in detail in Chapter 3.

2.9 Summary

A general methodology for the VLSI implementation of image compression systems was reported. The different processes and software environments were described along with an example of implementation of

an image coder circuit. Starting with the mathematical description of each of the different modules of JPEG, this chapter illustrated a particular sequence of tasks and processes that transforms an image coding algorithm into an integrated circuit.

It is also worth noting that some intermediate results of this methodology can also be used to develop solutions for other kinds of technologies (e.g., DSP, FPGA).

Chapter 3

Bit-Serial Architecture for a Motion Estimation Algorithm

3.1 Introduction

An examination of two consecutive frames of a given television scene would immediately reveal their striking similarity. A consequence of the two frames being captured by the TV camera at very close sampling times. This high degree of temporal redundancy is exploited in video compression, by attempting to convey the information that is common to two (or more) frames only in its first instance. Subsequent occurrences of that information being described by a reference to previous appearances. Interframe coding, is the term that has been given to this kind of frame-to-frame redundancy reduction, while intraframe coding is used to refer to the spatial redundancy reduction within a single image/frame.

A simple interframe coding method consists in processing the frames obtained from subtracting the consecutive images of a video sequence. That is, the encoding algorithm is applied on the difference-image resulting from the subtraction of adjacent frames. This interframe DPCM operation reduces a significant part of the temporal redundancies, particularly in low time-varying regions. However, for fast moving objects or areas in a scene, the simple frame differenting scheme is not very efficient, and indeed could lead to a decrease of the coding efficiency of a system.

Another popular approach for reducing the temporal redundancy in a video sequence is by using motion prediction techniques. Most of current standards for video coding (e.g., H.261, MPEG-1, MPEG-2) integrate motion prediction by executing some kind of motion estimation/compensation algorithms.

In this context, the process of predicting the *movement* of pixels associated to objects or regions across two (or more) frames, is known as motion estimation. Its implementation, along with the complementary motion compensation stage, contributes to a significant improvement of the coding efficiency with respect to the simple frame differencing scheme. Thus, it has become an essential component of current video compression systems.

The inconvenience of using motion estimation is its huge computational complexity. In current industry video coding standards, for example, the complexity of the algorithms is always driven by the motion estimator. For the low-bit rate coding standard H.261 more than 60% of the operations in the encoder are dedicated to the calculation of the motion vectors. This percentage increases to more than 80% for the high bit-rate MPEG-2 encoder.

Regarding the hardware implementation of a motion estimator, one of the main issues is input bandwidth, owing to the large amount of data that must be supplied to the system, for processing, in a relatively very short span of time. Complexity and cost are, as well, of paramount importance. A bit-serial VLSI architecture for a motion estimation algorithm is proposed in this chapter. The architecture deals directly with the bandwidth and complexity issues. Cost is implicitly addressed.

3.1.1 Organization of the chapter

The remainder of this chapter is organized as follows. The block matching motion estimation algorithm is introduced in Section 3.2 and an efficient implementation of the algorithm for bandwidth reduction is presented in Section 3.3. The proposed bit-serial architecture and each of its constituent parts are exposed in Section 3.4. Then, an ASIC of a motion

estimator, based on the proposed architecture is reported in Section 3.5. Finally, the chapter is closed with a summary in Section 3.6.

3.2 Algorithms for Motion Estimation

There exists several techniques that have been developed to estimate the motion vectors in a video sequence. These methods can be grouped into three categories. a) Phase correlation techniques [69, 70], b) Element recursive techniques [71–73], and c) Block matching techniques [74–76].

The techniques in categories b) and c), have received particular attention for interframe predictive coding, and have been the object of numerous optimization schemes. But, by far, the block matching techniques are preferred in current video compression systems, due to their better performance at minimizing the energy of the prediction error frames [77][1].

3.2.1 Block matching techniques

Figure 3.1 shows the principle of block matching motion estimation. The current frame is divided into non-overlapping blocks of $N \times N$ pixels (indeed in general, the frame can be divided into blocks of rectangular form), which are called the reference blocks (RBs).

For each RB a corresponding search area (SA) of $(2p+N) \times (2p+N)$ pixels is defined on the (decoded version of the) previous frame. The SA contains $(2p+1) \times (2p+1)$ different blocks of $N \times N$ pixels, which are called the candidate blocks (CBs). The goal of the block matching algorithm is to find, based on a given similarity criterion, where is the best match for each RB, among the different $(2p+1)^2$ CBs in its corresponding SA.

[1] In a motion-compensated prediction scheme, once the motion vectors have been *estimated*, they are used to form a prediction image for the incoming frame (an operation that is referred to as motion *compensation*). The prediction error frame is the difference between the real frame and its prediction. The term Displaced Frame Difference (DFD) is often also used to identify the prediction error frame.

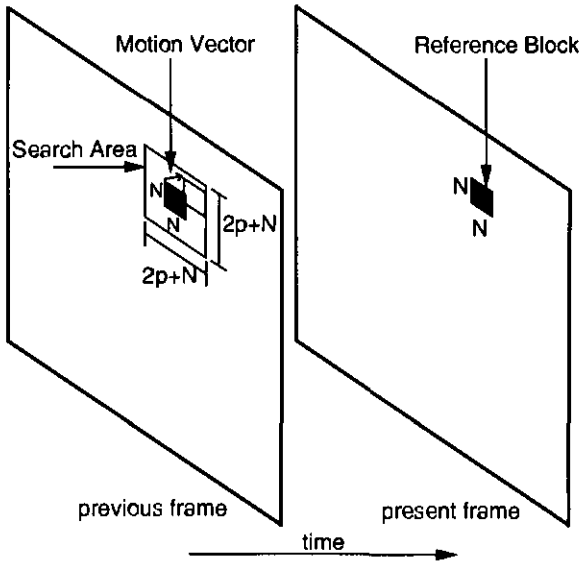


Figure 3.1: Block matching motion estimation.

When the RB is compared with all the CBs in the SA, the process is called full-search block matching. Obviously a complete exploration of the SA assures that the best match will be found for each RB. Nonetheless the procedure is computationally expensive. Some methods have been proposed in order to reduce the complexity, by reducing the number of matches in a strategic manner [78–80]. The main drawback of the latter methods is that the finding of the best matching CB is no longer guaranteed.

Further improvement of motion estimation methods has been made by considering motion vectors with fractional accuracy [81,82]. Since the *real motion* of an object from one frame to the next, is independent of the artificially pixel grid, an amelioration on the precision of the motion vectors is expected by computing the motion estimation with a subpixel accuracy. Obviously, the improvement is obtained in exchange for an increased complexity.

3.2.2 The matching criterion

Three cost functions have usually been used in order to find the best match of a reference block among the candidates blocks on the search area. The Cross-Correlation Function (CCF), the Mean Square Error (MSE), and the Mean Absolute Difference (MAD) [83]. The definition of the MAD, which is also known as Mean Absolute Error (MAE), is given by Equation (3.1),

$$\mathrm{MAD}(i,j) = \frac{1}{N \times N} \sum_{x=1}^{N} \sum_{y=1}^{N} |A(x,y) - B(x+i, y+j)| \qquad (3.1)$$

for $-p \leq i, j \leq p$. $A(x,y)$ represents the intensity of a pixel with coordinates (x,y) on an RB, and $B(x+i, y+j)$ corresponds to the intensity of a pixel with coordinates $(x+i, y+j)$ on the SA. The values of i and j, associated with the minimum MAD value that results after having explored the SA, indicate the components of the motion vector corresponding to the current RB.

3.3 Implementation of the Algorithm

A simple example of the block matching algorithm with parameters $N = 3$ and $p = 3$ is shown in Figure 3.2. It will be used to illustrate the implementation of the algorithm and the operation of the bit-serial architecture. The SA of 9x9 pixels is shown in Figure 3.2(a), corresponding to the RB of 3x3 pixels shown in Figure 3.2(c). There are a total of 7x7=49 CBs in the SA to be compared with the RB.

3.3.1 Reducing the input bandwidth

As previously noted, input bandwidth is one of the main issues in full search motion estimation algorithms. In this sense, an important optimization can be made, if advantage is taken of the multiple pixels that are common to different CBs. That is, if a pixel is used in the matching process of more than one CB (which is the case for all the pixels on the

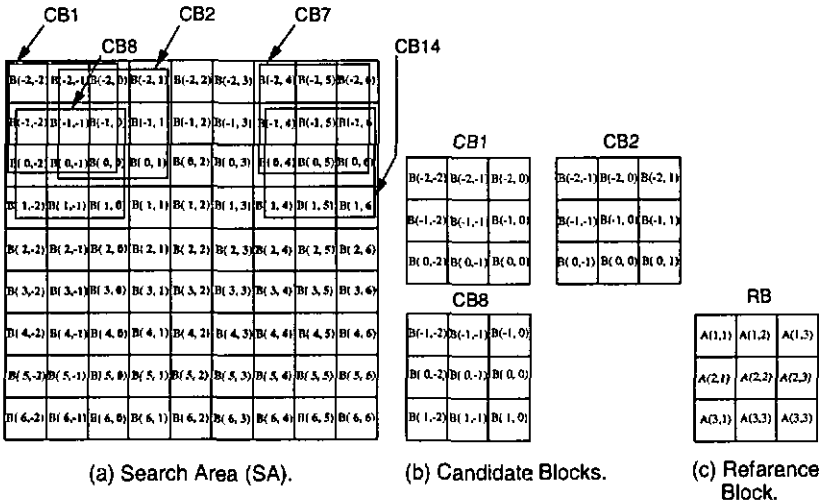


Figure 3.2: Example of a block matching algorithm with N = 3 and p = 3.

SA, except for the four pixels on the corners), then this pixel can be read from the previous frame memory only once.

Depending on the position of this pixel in its first CB, once it has been read, it can be either directly pipelined into its new position on the immediately following CB, or stored temporarily in local memory until the next CB that uses this pixel is evaluated. An example of the former case is illustrated in Figure 3.2, where it can be noticed that the pixels in the 2nd and 3rd columns of CB1 correspond to the 1st and 2nd columns of CB2 respectively. Thus, after CB1 has been loaded and matched, in order to form CB2, only 3 additional pixels should be read (i.e., the 3rd column of CB2: B(-2,-1), B(-1,-1) and B(0,1)). The six remaining pixels being obtained by means of two column shifts from the previous CB1.

In general, no additional memory is required when a CB uses data from the neighbor horizontally adjacent to its left. This is so because when a pixel is broadcast to the module that implements the sum of absolute differences (ADs), the pixel's bits are simultaneously being pipelined into the new columns that this pixel is going to be part of in the next CB to be processed.

In the preceding description, the input bandwidth was reduced by taking advantage of common data between two horizontal adjacent CBs. A further reduction can be obtained by also taking advantage of data common to two vertically adjacent CBs. For example, to form CB8 (and in general all the leftmost CBs, i.e., CB15, CB22,...,CB43) one only needs to read three additional pixels: B(1,-2), B(1,-1) and B(1,0), to complete the 3rd row.

Finally, with a combination of both pixel-data circulation paths, to form CB9 (CB16, CB23, etc) and all their CBs horizontally to their right, only one new pixel has to be loaded, i.e., B(1,1). This will be the most common case; meaning that to form a new CB, the majority of the time it will be needed to read only one additional pixel, which represents a dramatic reduction of the initial bandwidth requirement.

3.4 Architecture

The proposed architecture is shown in Figure 3.3. The frame memories (RAMs) output the pixel data in parallel form; since the reported architecture is bit-serial, a parallel-serial converter is required. For this purpose two of the Conversion Register Banks (CRB) proposed in [84] were used.

3.4.1 Candidate block memory

The Candidate Block Memory module (CBM) is shown in Figure 3.4. It is formed of a series of k-bit registers, each having the logic structure shown in Figure 3.5. The C registers contain the $N \times N$ pixels of the current CB being matched. The wires identified with an arrow pointing to the left margin of the page, perform the pipelined column-pixel shifting. For example, for $l \neq 1$, a pixel value stored in a register C(k, l) when CB2 is evaluated, will move to the register C$(k, l-1)$ one matching cycle later, when CB3 takes the place of the current candidate block.

The R registers provide both the row-pixel shifting (for the 1st and 2nd columns) and the delay of data common to two leftmost CBs, e.g.,

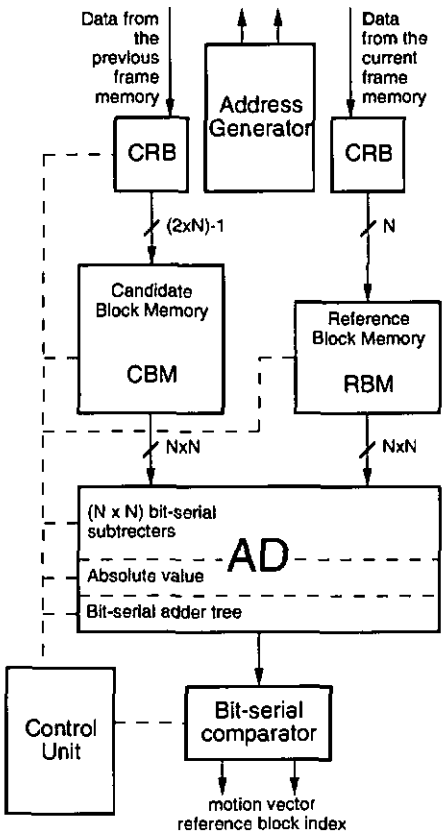


Figure 3.3: Architecture.

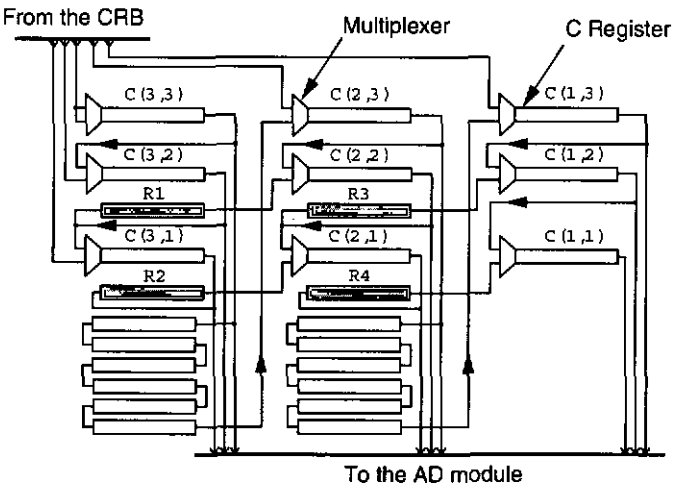
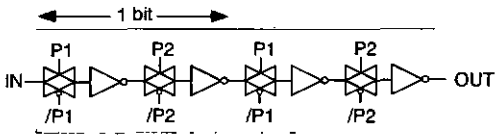
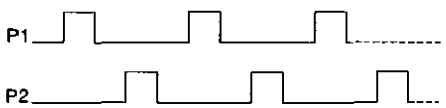


Figure 3.4: Candidate block memory (CBM).



(a) Logic structure



(b) Control signals

Figure 3.5: A two-bit serial register.

CB1 and CB8; CB8 and CB15, and so on. For example, when CB1 is being evaluated, register C(3,1) contains the value of pixel B(0,-2). At the end of this evaluation, the content of C(3,1) has been completely pipelined into the register R2. The clocking of the R registers is stopped during 5 matching cycles, being resumed at the start of the evaluation of CB7. At the end of this evaluation, the original content of register C(3,1), i.e., B(0,-2), is now in the register C(2,1), and is ready to be used for the evaluation of CB8. The fact of stopping the clocking of the R registers (which as the rest of the registers are dynamic) does not cause any loss of data. Experimental tests of chips, carried out by the author, have demonstrated that the data is still preserved in this kind of dynamic registers, even at relatively low frequencies. Since these registers are expected to operate in the MHz range, the temporary inhibition of their clock, during a few cycles, is a safe operation. By using this scheme, a significant number of buffer registers is saved.

The two blocks of six shift registers at the bottom of the figure, perform the row shifting of the 3rd column and contribute to the respective delay of 7 matching cycles, between data used for example by the 3rd columns of CB1 and CB8, CB2 and CB9, CB3 and CB10 and so on. Besides, these blocks of registers also provide the resources to store the rest of the pixels, not covered by the R registers.

3.4.2 Reference block memory

The block diagram of the Reference Block Memory (RBM) is shown in Figure 3.6. Since the same RB is compared with all the $(2p+1)^2$ CBs in the SA, each pixel of a RB must be loaded into the AD module $(2p+1)^2$ times. This explains the self-looping structure of the $A(k,l)$ registers. The wires providing the register data re-circulation, are also used during the loading of a new RB, to transfer data from one register to its neighbor one floor lower (e.g., from A(3,3) to A(3,2), from A(3,2) to A(3,1)). This permits a reduction of the number of wires required to charge the RBM module with a new RB, in exchange for a few clock cycles of latency. The same principle is used to fill the CBM module in

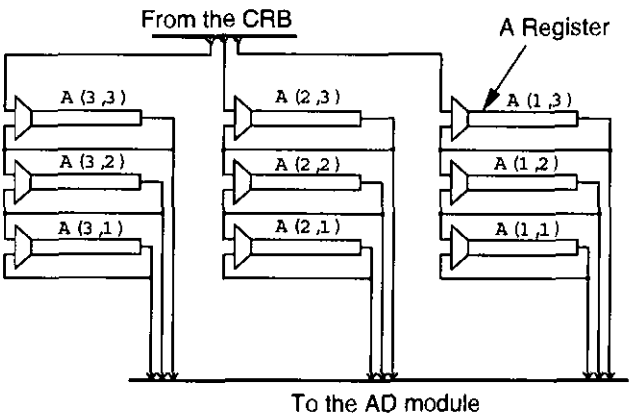


Figure 3.6: Reference block memory (RBM).

Figure 3.4.

3.4.3 Absolute difference module

The Absolute Difference (AD) module is shown in Figure 3.7. It corresponds to a set of $N \times N$ serial subtracters, each followed by a logic circuit that performs the absolute value. An adder tree sums all the $N \times N$ absolute values, producing the value of the MAD for the CB that is being compared. The wordlength of the binary number at the output of the adder tree (considering a gray-level resolution of 8 bits/pixel) is given by: $8 + \lceil \log_2(N \times N) \rceil$ bits, where $\lceil x \rceil$ denotes the largest integer greater than or equal to x. For an application in which $N = 8$, the output of the adder tree will be represented with 14 bits, and accordingly, all the data registers downstream should have a fourteen-bits wordlength.

3.4.4 Comparator

The bit-serial comparator selects the minimum value among the $(2p + 1)^2$ MAD computations, and the position of the best-matching CB is generated in order to indicate the motion vector of the current RB. Figure 3.8 shows a simplified logic diagram of the comparator. Basically,

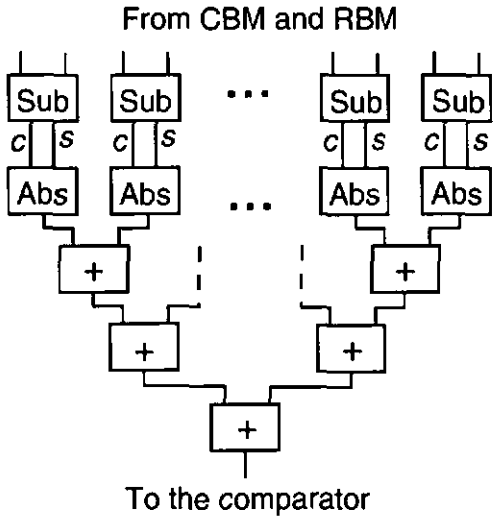


Figure 3.7: Absolute difference module.

it is composed of three registers, one subtracter, and one multiplexer. In order to indicate the position of tbe CB that is being matched, one counter is used. A set of latches (L_i) stores the best matching position.

The register R1 is used to keep the (temporary) minimum MAD. When m=1, the register R1 is refilled with its same previous value (a data loop). When m=0, the content of the register R1 is replaced by the content of the register R2. The latest computed MAD value is loaded into the register R3 from the output of the adder tree. The content of R3 (which is overwritten each matching cycle with a new MAD value) is always pipelined into the register R2. Thus, the registers R3 and R2 always contain the last and the second to last MAD values.

When the value of m is 1, R1 is compared with R3 by means of a subtraction (R1 minus R3). The value of the carry c is sampled by making S=1 at the moment when the two most significant bits of R1 and R3 have just been subtracted. If c=1, it means that the value of R1 is still the minimum, and the value of m is held at 1. In case c=0, that means that R3 < R1, and m becomes 0, so that the next

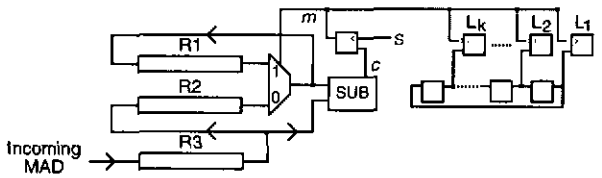


Figure 3.8: Comparator.

comparison will be made between the content on the registers R2 (that temporary contains the minimum MAD value) and the latest calculated MAD contained in R3.

The counter is incremented each matching cycle. When m becomes zero, it means that a new minimum has been found, and the state of the counter is transferred into the latches L_i. Thus, the content of the latches always indicates the position of the latest best matching CB, i.e, the current best motion vector.

3.4.5 Appraisal of the architecture

The study of architectures for motion estimation has been and continues to be the object of intensive research [85–87]. For the architecture presented in this chapter, the main characteristics that are worth pointing out are the following.

The combination of bit-serial schemes with the strategy of implementation described in Section 3.3 produces an architecture with a minimum complexity of structure. This, as will be shown in the next section, efficiently leads to a low complexity high and low level hardware design, and to a very small silicon area circuit.

As noticed from the previous presentations, the basic building blocks of the architecture are bit serial units (registers, adders, etc.), in which the data processing is maximally pipelined. In terms of hardware this has two advantages. First, the capacitive load downstream of each node is minimal and thus the different modules can run at high frequencies [57] (hundreds of MHz). Second, the full pipeline characteristic of the blocks performing both the arithmetical operations and

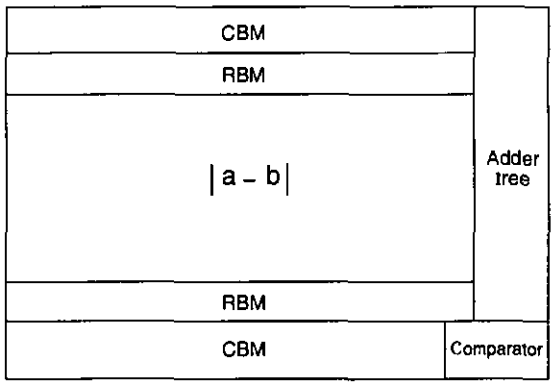


Figure 3.9: Floorplan of the motion estimation circuit.

movement of data, allows an efficient use of hardware at nearly 100% during a frame cycle.

3.5 An ASIC for Motion Estimation

The previously reported architecture was used for the realization of an ASIC for a low bit-rate video application. The specifications of the application were the following. Frame size, common intermediate format (CIF): 288 lines and 352 pixels/line; frame frequency, 10Hz; reference block (RB) size, 8x8 pixels; and maximum displacement p, 8 pixels.

The number of RBs per frame is (288x352)/(8x8)=1584. The MAD of each RB, is obtained after having made $(2p+1)^2 = 289$ matches. As has been pointed out before, for $N = 8$, a new match is made every 14 clock cycles. To obtain the motion vector of each RB one needs: 289x14=4,046 clock cycles; plus 8x14=112 clock cycles to fill the pipeline of the CMB (only once per RB), which gives a total of 4,158 clock pulses. The current frame is thus calculated in: 4,158x1584=6,586,272 clock pulses; plus 196 cycles required to fill the pipeline of the CRB (only once per frame). At the specified 10 frames/s, a clock rate of approximately 65 MHz is required; this frequency was largely surpassed during simulation of the circuit.

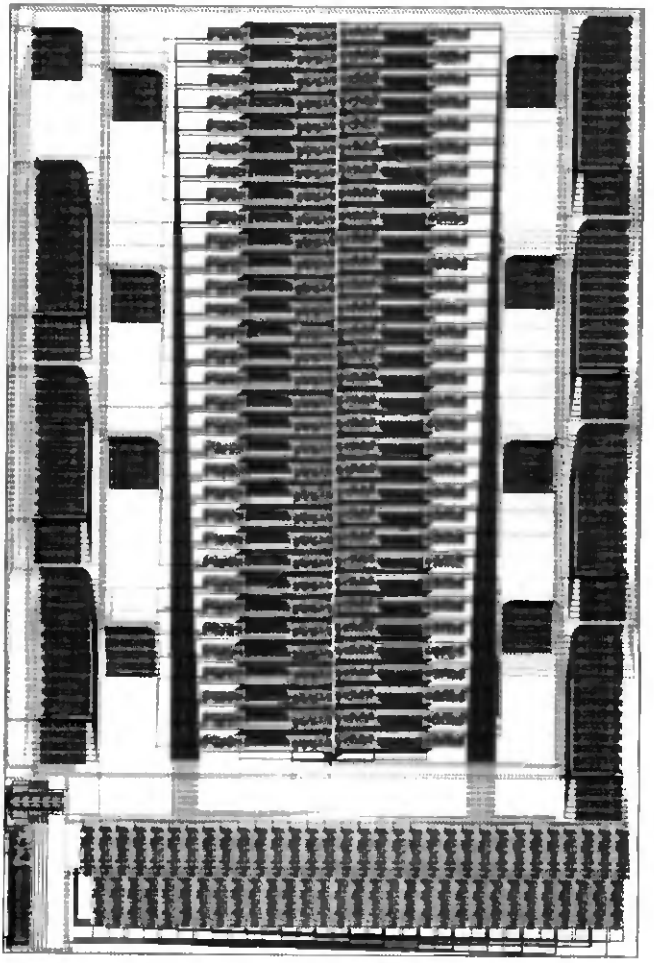


Figure 3.10: Layout of the motion estimation circuit.

Two clocking approaches were evaluated for the blocks, pseudo 2-phase [66] and true single phase [88] clocking strategies. The floorplan and the layout of the motion estimation circuit are shown in Figures 3.9 and 3.10 respectively. The central part of the circuit contains the 64 subtracters each one followed by an absolute value unit. The adder tree is located at the bottom of the circuit in Figure 3.10, while the bit-serial comparator occupies a small area at the bottom-left corner of the chip. In order to reduce the surface dedicated to run the wiring between the candidate and reference block memory registers and the AD unit, the CRB and the CMB have been split into 8 parts each. The corresponding parts of the CRB and CRM have been arranged in pairs and distributed in the vertical periphery of the Absolute Differences unit. The area of the chip is 4.26 x 2.89 $mm^2 \approx 12.3$ mm^2 and it contains approximately 41,500 transistors. The circuit was implemented in the 1.2μm 5V CMOS CMN12 process from VLSI Technology Inc.

3.6 Summary

A bit-serial architecture for a block matching motion estimation algorithm has been reported. It is well suited for the estimation of the motion vectors in low bit-rate applications as teleconferencing and videophone systems. The architecture features a minimum complexity, and achieves a significant reduction of the input bandwidth. Based on the presented architecture, an ASIC of a motion estimator has been realized for a particular application. The resulting chip is very competitive in terms of silicon area in comparison with other reported designs.

Chapter 4

Power Management Strategy for Wireless Image Communications

4.1 Introduction

Low power consumption is a requirement for battery powered portable equipment. When designing ASICs for image and video compression, emphasis has been placed mainly on realizing circuits that are fast enough to satisfy the high data throughput associated with image and video processing. The imminent development of portable systems featuring full multimedia capabilities, has added the low-power constraint to the design of VLSI circuits for this kind of application.

Several techniques such as lowering the supply voltage [89], architectural parallelization [90], and pipelining [90], to mention a few, have been proposed to achieve low-power consumption in digital circuits. In this chapter a power management strategy that allows to trade image quality for power consumption will be presented.

4.1.1 Organization of the chapter

The remainder of this chapter is organized as follows. In Section 4.2 an alternative, implementation supported, trade-off in the image compression domain is presented, along with some examples of applications. Section 4.3 introduces a power management scheme for transform-based image coding. The advantages of the simple scheme are examined in

Section 4.4 in comparison with approaches that make it possible to control the quality of the encoded images, though without any positive effect on the power consumption. The power-down scheme is discussed in Section 4.5. Some results are presented in Section 4.6 and finally, a summary of the chapter is given in Section 4.7.

4.2 An Alternative Trade-off

For data compression purposes, lowering the high frequency content of an image has proved to be a successful method for achieving low coding bit rates. In zonal coding for example, a unitary transform, such as the Discrete Cosine Transform DCT, is applied on 2-D blocks of pixels of the original image. By selecting only a few of the coefficients on a predefined zone on the frequency domain, one is able to obtain a more compact representation of the image in exchange for a loss of quality in the reconstructed image. A similar principle is used on the baseline JPEG algorithm, where the 2-D transform operation is followed by a quantization and an entropy coding of the coefficients.

In the above-mentioned methods, and in general in a classical image compression scheme, the issue is to obtain a minimum distortion on the reconstructed image, while using a minimum number of bits to represent the compressed image. However, since image distortion and coding bit rate are mutually exclusive for absolute optimization [92], the operation of image compression systems responds to a trade-off between decoded image quality and coding bit rate.

In certain applications, the best image quality might not be required at particular periods of time. For example, while image browsing a database, during "hold-on" periods during one- or two-way communications, and during "no activity" periods in surveillance, to mention a few cases. Many of these applications are expected to enter the market in the near future, integrated in a portable device. For these systems, the trade-off alluded to on the previous paragraph can be reformulated in terms of image resolution and power consumption [91]. That is, in these portable video systems, whenever the highest image quality is not

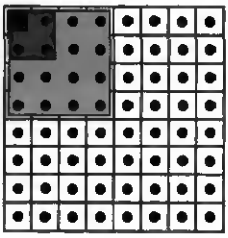


Figure 4.1: PM modes.

required, the user and/or the system can trade image resolution (degree of discernible detail) for power consumption.

4.3 Power Management Strategy

Power management (PM) techniques play an important role in the reduction of power consumption in portable equipment [93]. They are being successfully applied in the design of current microprocessors [94] and notebook computer systems [95].

A PM scheme allows the reconfiguration of a system in order to save power, while still operating in a consistent manner with the application. The basic principle is to power down a whole system (or parts of it) or stop the clock to a module, when it is idle for a significant period of time.

An architecture featuring a power management scheme is here proposed for transform based coding algorithms. It has been implemented in the circuit that performs the sequential baseline JPEG coding algorithm reported at the end of Chapter 2. Functioning under this PM scheme, the circuit has four different modes of operation, which are represented in Figure 4.1: a) Mode DC, b) Mode 4, c) Mode 16 and d) full JPEG mode. Each dot in Figure 4.1 represents one of the 64 coefficients that results after the application of a 2-D DCT on a block of 8x8 pixels. The top-left point represents the DC coefficient.

In Mode DC, only the DC coefficient is computed and encoded. In this mode a rough blocky version of the image is obtained. Mode

4, commands the encoder to compute and encode only the four lower frequency coefficients. In this mode, the main content of the image is already perceptible and a particular known image could be easily recognized. However the image still presents a blocking aspect, especially at the edges of objects. In Mode 16, the encoder selects and codes only the 16 lower frequency coefficients. In this mode the images are of a relatively good quality. The blocking structure on the edges of objects is strongly attenuated, with respect to Mode 4. Finally, the full JPEG mode, allows the circuit to produce a complete (JPEG-quantized) spectral resolution.

When a high image quality is not required for transmission (or reception), the user can select the mode of operation at the start of the encoding (or decoding) process. As soon as the number of coefficients corresponding to a mode of operation has been computed, the control unit powers down the DCT and the quantization units. A multiplexer allows the setting of the values of all the non-computed coefficients to zero, in order to generate a fully JPEG-compliant bit stream.

This scheme saves power by avoiding unnecessary computation. It can be interpreted as a hardware adaptive zonal coding scheme embedded in the baseline JPEG algorithm. Since in JPEG, the DCT is the most expensive operation in terms of computational power, the gain in terms of energy-savings can be very significant. In full JPEG mode the circuit works as a regular JPEG coder system.

4.4 Further Considerations

It is worth noting that in a regular JPEG encoder, when the best image quality is not required, one could scale-up the JPEG normalization matrix in order to achieve higher compression ratios. The reduction of the coding bit rate is achieved due to an increase of the number of 2-D DCT coefficients that become zero after quantization. Nevertheless, in a JPEG coder circuit without a PM strategy, that implies a waste of power, since the DCT unit must always compute the 64 2-D DCT coefficients, regardless of the potentially large number of zero-valued coeffi-

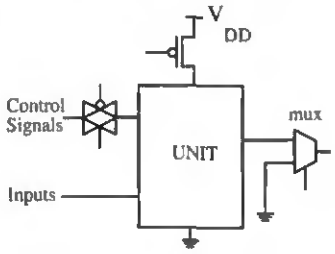


Figure 4.2: Power-down scheme.

Figure 4.3: Original image.

4 word cycles). The quantization unit is equally powered off during 60 word cycles. For the reported JPEG circuit, a word cycle corresponds to 12 clock cycles.

Due to its storage function and its inherent power-dependent condition, the RAM that executes the matrix transposition of the 2-D DCT (Figure 2.8) must remain powered on, regardless of the selected PM mode. The same is true for the Huffman coder that must release a complete decodable bit stream.

cients, that would result after a coarse quantization. A similar statement can be made for the quantization unit.

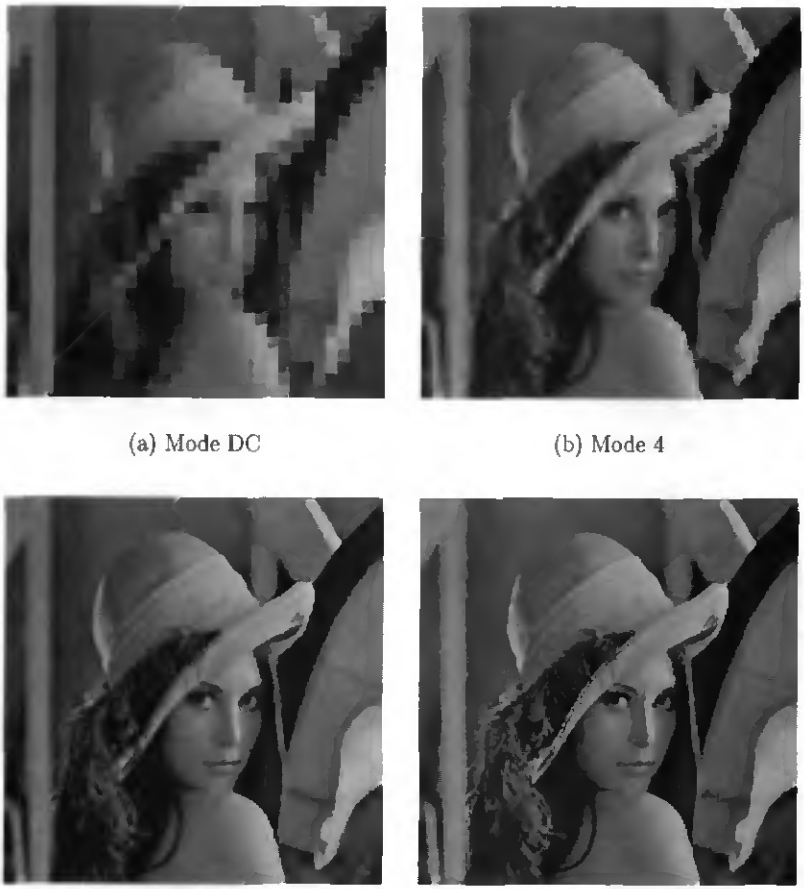
The JPEG standard also defines other modes of operation (i.e., progressive and hierarchical modes) that can be properly used in cases when the best image quality is not required. However, a hardware implementation of these modes does not necessarily lead to power-saving architectures. This is because, these more elaborated JPEG modes take place at the expense of additional power-consuming hardware owing to the significant increase of the system complexity with respect to the baseline sequential algorithm.

4.5 Power Controlling Scheme

The PM strategy described in the preceding section can be applied to trade image quality for power consumption at both the encoder and the decoder. Both realizations are independent, and no information overhead is required to indicate the mode in which the images are coded.

The VLSI architecture of the 2-D DCT was shown in Figure 2.8, and that of the quantizer in Figure 2.10. The basic approach of the PM scheme is shown in Figure 4.2, which for the JPEG coder, the UNIT block represents botb the 2-D DCT and the quantization modules. The multiplexer at the outpnt is required to keep the bit stream uninterrupted when the unit is powered off. It sets the value of all the non-computed coefficients to zero. One or several large pMOS transistors are used to power the unit on and off.

The PM strategy is dynamic [96], meaning that only certain modules of the encoder (or decoder) are to be powered off, namely the DCT and the quantizer. In Mode 4, for example, sixty of the 2-D DCT coefficients are not required and thus, should not be calculated. This allows the control unit to power off, the two DAPs (Distributed Arithmetic Processors) of the 2-D DCT circuit and the whole quantization unit. Thus, in Mode 4, the first DCT's DAP is powered off during 48 word cycles (i.e., it is active only during 16 word cycles), whereas the second DCT's DAP is powered off during 60 word cycles (i.e., it is active only during



(a) Mode DC

(b) Mode 4

(c) Mode 16

(d) Full JPEG mode

Figure 4.4: Image quality for the different power management modes.

4.6 Power-down State Periods and Image Quality

The test image Lena (256x256 pixels, 8 bits/pixel), is shown in Figure 4.3. The image quality obtained with the different modes of operation, after they have been JPEG-decoded, is shown in Figure 4.4.

When the JPEG coder circuit is operating on full JPEG mode, it executes 128 scalar products per block of 8x8 pixels (64 scalar product each DAP of the 2-D DCT) and 64 multiplications that correspond to the quantization of the 64 DCT coefficients. For an image of 256x256 pixels, that means a total of 131,072 scalar products and 65,536 multiplications.

When the coder circuit is set in Mode DC, for the previously given image resolution, the number of operations is reduced to 9,216 scalar products and 1,024 multiplication. That means that the 2-D DCT unit is powered off about 93% of the time. Figure 4.4(a) shows the quality of an image that has been coded in Mode DC.

In Mode 4, the most computational intensive part of the JPEG circuit is powered off 84% of the time. The images present a blocky aspect on the edges of objects, but their content is still intelligible. In this mode the peak signal-to-noise ratio (PSNR)[1] of the JPEG-decoded image Lena, shown in Figure 4.4(b), is 25.91 dB.

In Mode 16 the most computational intensive part of the JPEG circuit is powered on only 38% of the time. The decoded images are of a relatively good quality. In this mode, the PSNR of the JPEG-decoded image Lena, shown in Figure 4.4(c), is 30.30 dB.

In full JPEG mode, no power-saving is made and from an algorithmic point of view, the circuit works as a regular JPEG system. For comparison, the PSNR of Lena in full JPEG mode is 32.70 dB, the decoded image is shown in Figure 4.4(d).

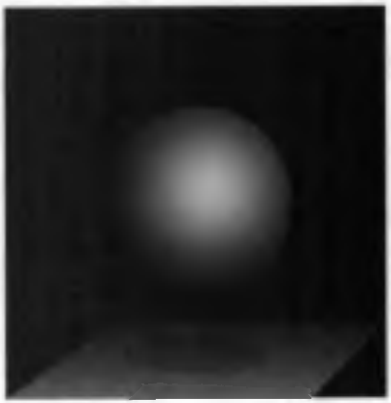
[1] If $f(i,j)$ and $\hat{f}(i,j)$ represent the pixels of the original and the coded-decoded NxN pixels image respectively, then the $PSNR$ is given by: $PSNR = 20\ \log_{10}(\frac{255}{RMSE})$, where $RMSE = \sqrt{\frac{1}{N \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{N} [f(i,j) - \hat{f}(i,j)]^2}$.

4.7 Summary

Power management is a successful approach to reduce power consumption in portable equipment. The latest microprocessors, desktop and laptop microcomputers are using this kind of strategy, which consists in shutting down the power to certain units when they have been idle for a determined period of time.

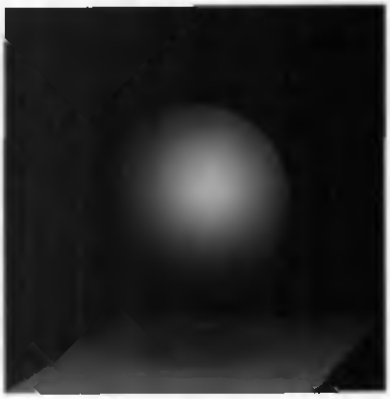
In this chapter a similar mechanism has been proposed to save power in portable image communications systems. The basic idea is that whenever the best image quality is not required for transmission or reception, a significant saving of power can be made by avoiding the computation of the components that convey the detailed structure of the images. Thus, at particular times, some modules of the compression system can be powered-off and the value of the non-computed data is replaced with zero or any other appropriate fixed value.

(a) Baboon: Original



(b) Ball: Original

(c) Reconstructed: CR = 5.73



(d) Reconstructed: CR = 44.36

Figure 5.1: JPEG compression: different CRs for different images.

Chapter 5

Bit-Rate Control for the JPEG Algorithm

5.1 Introduction

One of the characteristics of the JPEG algorithm is that of producing a compression bit rate that varies depending on the level of information content of the image that is being coded. As a result, two images with the same spatial and gray-level resolutions—i.e., same width, height, and number of bits per pixel—but with different degrees of image detail, will result into two JPEG-compressed bit streams that, in general, do not have the same number of bits. These two images are thus coded with different bit rates.[1]

Figures 5.1(a) and (b) show two images of 512x512 pixels that have largely-different degrees of information content. The compression with JPEG of the image in Figure 5.1(a), which is an image that contains a significant amount of detail, results in a bit stream whose number of bits corresponds to a compression bit rate of 1.40 bit/pixel (bpp). The reconstructed image for this bit rate is shown in Figure 5.1(c). The compression bit rate for the image in Figure 5.1(b), which is an image that contains substantially less detail than the image in Figure 5.1(a), is

[1] In the image compression literature, the compression (bit) rate is used, alike the compression ratio (CR), to characterize the performance of a compression algorithm. When the input images are of 256 gray levels—i.e., 8 bits per pixel—the compression bit rate (CBR) is related to the compression ratio (CR) by: CBR=8/CR.

0.18 bpp; its reconstructed version is shown in Figure 5.1(d). In terms of compression ratio (CR), JPEG is coding the former image with a compression ratio of 5.73, while is coding the latter, using the very same set of JPEG parameters, with a CR of 44.36: a large difference!. It is worth observing that in Figures 5.1(c) and (d), the higher CR for the image in Figure 5.1(b) is not obtained to the detriment of the quality of the decompressed image. Actually, in terms of PSNR, the quality of the reconstructed image in Figure 5.1(d) is 15.22 dB *better* than that of the reconstructed image in Figure 5.1(c). This fact is reminded since it is always possible to increase the compression ratio in exchange of a loss in the quality of the reconstructed images; nevertheless, in this example this is not the case nor the point that it is wanted to be emphasized.

This image-depended, variable compression bit-rate characteristic of JPEG represents a main drawback for some applications. In digital still cameras for example, it is important to be able to predict the size of the compressed pictures in order to control the allocation of the solid-state memory of the camera. Being able to control the resulting compression bit rates is also required in networking applications that involve the transmission of image/video signals. In these networks, the time required to transmit a compressed image is generally fixed or not allowed to exceed a certain value; precluding in consequence, the use of non-controllable bit-rate schemes. An example of the latter application is given in [97] where a bit-rate controlled Motion-JPEG (M-JPEG) is needed to fulfill particular systems requirements.

In this chapter, two bit-rate control methods for the JPEG algorithm are presented. These techniques permit to produce, with an excellent accuracy, a user- or system-requested compression ratio (or its equivalent compression bit rate) of JPEG-coded images. The presented methods have several advantages [98] with respect to other bit-rate schemes that have been previously reported [99–102]. Some of these advantages are:

- A full compliance with the JPEG algorithm. This feature gives the system all the advantages inherent to *standard* algorithms; as, for

example, a widespread use of the compressed images.

- A very low computational complexity. This is an important characteristic in order to produce high-speed, cost-effective hardware implementations, and also to keep power consumption as low as possible.

  The bit-rate control system proposed in [99], for example, requires too many iterations to converge. While the systems in [100–102] must execute many computations for each 8x8 block of pixels in order to check if the allocated bits per block have not been exceeded.

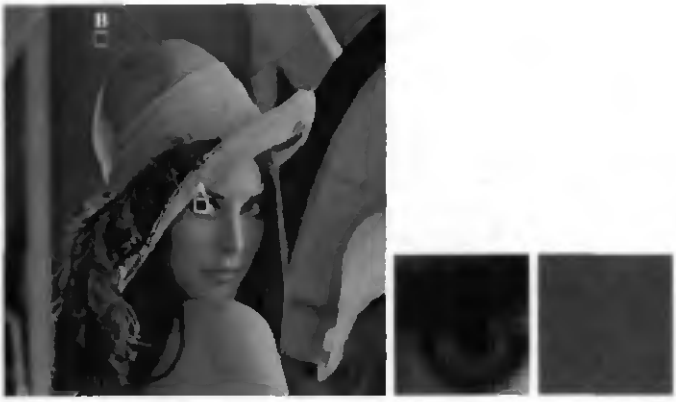
- By their own principle, the application of the bit-rate control mechanisms proposed in this chapter, do not cause any further degradation to the compressed images. The quality of the decompressed images is exactly the same that would be given by JPEG (with the corresponding parameters) in the absence of the bit-rate control system.

  The lack of this feature is a major drawback of the systems reported in [100–102], which must suppress high frequency information in order to make the bit rate converge to the target value.

- Finally, the methods proposed in this chapter, provide an excellent accuracy for setting the compression ratio, and the largest application operational range.

5.1.1 Organization of the chapter

The remainder of this chapter is organized as follows. Section 5.2 illustrates JPEG's variable compression rate nature. Section 5.3 illustrates how it is possible to modulate the output compression ratio given by JPEG. In Section 5.4 a study of the characteristic of the compression ratio vs. a modulating parameter is studied. The general scheme of the bit-rate control method is given in Section 5.5, and a particular implementation of this scheme is described in Section 5.6. Then, an optimized implementation is reported in Section 5.7, and its results are presented in Section 5.8. The computational complexity analysis is discussed in Section 5.9, just before a summary of the chapter in Section 5.10.



(a) (b) (c)

Figure 5.3: a) Two different image-activity blocks. b) Block A. c) Block B.

to Section 2.3.1 in Chapter 2, in order to perceive the difference between the encoding of the blocks A and B, at the various levels of the algorithm.

The values of the 64 pixels of block B are given by the following matrix:

$$\mathbf{B} = \begin{bmatrix} 134 & 133 & 139 & 141 & 134 & 134 & 135 & 137 \\ 133 & 134 & 136 & 136 & 136 & 136 & 135 & 133 \\ 138 & 135 & 137 & 140 & 137 & 139 & 138 & 135 \\ 132 & 138 & 137 & 141 & 139 & 137 & 137 & 136 \\ 135 & 138 & 140 & 142 & 136 & 139 & 138 & 137 \\ 136 & 135 & 134 & 135 & 140 & 138 & 139 & 140 \\ 136 & 136 & 138 & 138 & 137 & 136 & 135 & 141 \\ 137 & 136 & 140 & 140 & 138 & 139 & 138 & 140 \end{bmatrix} \tag{5.1}$$

The result of the application of a 2D-DCT on $\mathbf{B}$ (after subtracting 128 from each of its elements, as required by JPEG) is the matrix $\mathbf{X}$:

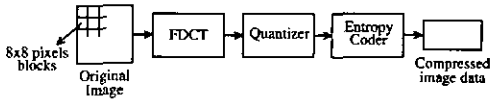


Figure 5.2: Baseline JPEG algorithm.

5.2 JPEG: A Variable Compression Bit-rate Algorithm

5.2.1 The effect

This section presents an example to illustrate the image-dependent, variable-rate mechanism inherent to JPEG. To support the presentation, the operations of the baseline JPEG algorithm are shown again in Figure 5.2. As depicted in the first block of the figure, these operations are not carried out at once over the whole image, but sequentially on subimage blocks of 8x8 pixels.

In Figure 5.3 two of these 8x8-pixel blocks, A and B, are specified within the test image Lena (only for the purpose of highlighting the difference in the content of image detail, the two blocks in Figure 5.3 are indeed 16x16-pixels blocks, the center region of which are the real 8x8-pixel blocks). Each block will be coded separately, following the JPEG scheme of Figure 5.2. The outcome of each of the operations will be shown and at the end of the algorithm, the two resulting bit streams will be compared. Block A has been selected as to contain more image detail than block B and should (due to the JPEG mechanism we are discussing) require more bits to be coded.

Block A is the same 8x8-pixel subimage that was used to illustrate the baseline JPEG algorithm in Chapter 2, Section 2.3.1. The values of the 64 pixels of block A are given by the matrix **A** on Equation 2.1 on page 15. The resulting compressed bit stream for this particular block was found to have 62 bits, which corresponded to a compression ratio of 8.26, for that particular block of the input image.

Block B will be coded by following the same steps as for coding block A. At this point the reader might find it useful to refer alternately

$$\mathbf{X} = \begin{bmatrix} 72 & -4 & -6 & -4 & 1 & 2 & 1 & -2 \\ -6 & 3 & -3 & 0 & -1 & 3 & -1 & -1 \\ -2 & 0 & 2 & -2 & 2 & 1 & 3 & 1 \\ -2 & -2 & 2 & -2 & 1 & -1 & 0 & 1 \\ 3 & 1 & -3 & -5 & -1 & 2 & 0 & -2 \\ 0 & 0 & 0 & -2 & 2 & -1 & -1 & 0 \\ 3 & -2 & 2 & 3 & 1 & 2 & 0 & 1 \\ 3 & 3 & 1 & -2 & 0 & 3 & 2 & -2 \end{bmatrix} \tag{5.2}$$

The smaller amount of image activity of block B in comparison with block A, is reflected in the transform domain, by the fact that the AC coefficients in Equation (5.2) have smaller magnitudes than the corresponding AC coefficients in Equation (2.2). This difference is even more emphasized after the quantization.

The quantization of the 2D-DCT coefficients in $\mathbf{X}$ with the normalization array $\mathbf{N}$ given in Equation (2.3), produces the following matrix,

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.3}$$

Notice the higher number of zero-valued coefficients in this matrix with respect to the matrix in Equation (2.4). As previously stated, this is a clear indication of the different "image-complexity" of the two original block of pixels.

The zigzag reordering of the elements of matrix $\mathbf{Q}$ produces the following vector,

$$\mathbf{z} = [5 \;\; 0 \;\; -1 \;\; 0 \;\; 0 \;\; -1 \;\; \underbrace{0 \ldots 0}_{58}] \tag{5.4}$$

The value of the DC coefficient of the previous transformed and quantized 8x8-pixel block was found to be 4. The DPCM operation over the DC coefficients gives the vector **d** below,

$$\mathbf{d} = [1 \;\; 0 \;\; -1 \;\; 0 \;\; 0 \;\; -1 \;\; \underbrace{0 \ldots 0}_{58}] \tag{5.5}$$

A zero run-length coding of the sequence of numbers represented by vector **d** results in the following vector,

$$\mathbf{r} = [(1) \;\; (1,-1) \;\; (2,-1) \;\; \text{EOB}] \tag{5.6}$$

The decomposition of the elements of vector **r** into the representation (*run/cat,cmp*) ((*cat,cmp*) for the DC coefficient) as explained in Section 2.3.1, yields,

$$\mathbf{c} = [(0,1) \;\; (1/1,0) \;\; (2/1,0) \;\; \text{EOB}] \tag{5.7}$$

By searching for a binary code in the Huffman tables A.2 and A.3 given in Appendix A, for each of the elements of vector **c** the following bit stream is obtained:

0101/11000/111000/1010

again for illustration purposes, a slash has been inserted between the binary codes associated with the different symbols of the vector **c**. The resulting compression ratio is 26.95 for the block B, which only requires 19 bits in compressed form: approximately one third of the number of bits needed to code the block A.

This example shows how an image that is abound in details, and thus contains a high proportion of high-activity A-like blocks, will require more bits to be coded than a relatively less *elaborated* image.

5.2.2 The cause

It is interesting to point out that the Discrete Cosine Transform operation is not the cause of the variable-rate feature of JPEG (as it is sometimes implied in the literature). There do exist image compression schemes, based on the DCT that yield a non-varying compression ratio.

Zonal coding [103] for example, is based on the same compression principle as JPEG, that is, on the application of a block-based linear transform such as the DCT. Yet zonal coding can produce the same compression bit rate, regardless of the content of the input image (all the same with a coding efficiency that is less than JPEG's).

The cause of the variable compression rate in JPEG is due to the contributions of two factors (albeit any of the two factors, on its own, is enough to render varying, the output bit rate of the system). The first one is the use of a run-length coding to represent the non zero-valued, normalized, 2D-DCT coefficients. The second, is the use of variable-length Huffman codes to represent the DPCM and run-length encoded coefficients. In Figure 5.2 these two elements are both part of the block denominated entropy coder.

In general, any image or video compression system that includes an entropy coder (e.g., Huffman or arithmetic coding) is expected to deliver a variable compression bit rate. For example, video compression standard MPEG-1, MPEG-2, H.261 and H.263, all includes entropy coders at the output stage, and thus, they all deliver variable compression bit rates. A bit-rate smoothing buffer [104, 105] is commonly used in order to regulate the output rate of these compression systems. The main inconvenience of this buffer is its contributed delay to the latency of the compression system.

5.3 Modifying the Compression Ratio

Let us suppose that one would have preferred to code the block B in the previous example with a higher compression ratio—i.e., with a lower bit rate. This would have been possible by means of scaling the normalization array by a constant factor, before the one-to-one division operation.

For example, quantizing the 2D-DCT coefficients of the block B given in Equation (5.2) with the normalization array $\mathbf{N}$ in Equation (2.3) previously multiplied by a factor of two would result in the following matrix,

$$Q_1 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.8}$$

The effect of having doubled the magnitude of the coefficients of the normalization array **N** is clear: more 2D-DCT coefficients have been zeroed out in matrix $\mathbf{Q}_1$ than in matrix **Q** in Equation (5.3). The effect at the output of the DPCM (the previous quantized DC coefficient was found to be 2) and run-length coders is reflected in the vector $\mathbf{d}_1$, which displays a smaller number of non-zero elements than the corresponding vector **d** in Equation (5.5). This change induces a large reduction in the number of bits required to represent the elements of vector $\mathbf{d}_1$.

$$\mathbf{d}_1 = [0 \underbrace{0 \ldots 0}_{63}] \tag{5.9}$$

After the category and complement representation, followed by Huffman coding, the output bit stream for the vector $\mathbf{d}_1$ contains 6 bits which corresponds to a compression ratio CR of 85.33, in comparison with a CR of 26.95 for the non-scaled normalization array case. In this instance, the increase of the compression ratio is obtained in exchange for a degradation of the quality of the decompressed block. This because the higher the number of coefficients that are *forced* to zero during the quantization stage, the lower the quality of the decompressed images.

5.4 Modeling the CR-SF Characteristic

In Section (5.3) it was shown how by scaling the normalization array one can modify the compression bit rate of the coded images. The question that remains at this point is how to find the constant factor by which

Image	Slope m	LSE Approximation	Size (pixels)	Error R^2
Airplane	4.730	$y = 4.730x + 8.359$	512x512	0.9920
Baboon	3.194	$y = 3.194x + 2.736$	512x512	0.9996
Barbara	3.710	$y = 4.710x + 5.591$	512x512	0.9986
Boats512	4.630	$y = 4.630x + 6.856$	512x512	0.9971
Lena	5.604	$y = 5.604x + 9.924$	512x512	0.9922
Peppers	5.223	$y = 5.223x + 9.033$	512x512	0.9881
Bridge	3.226	$y = 3.226x + 3.056$	256x256	0.9997
Camera	4.278	$y = 4.278x + 6.409$	256x256	0.9931
Lena256	4.367	$y = 4.347x + 6.473$	256x256	0.9959

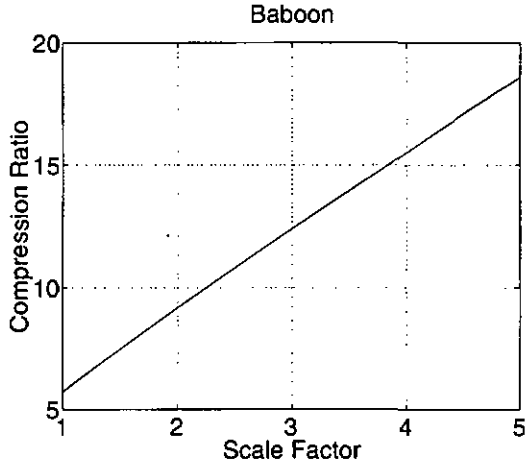
Table 5.1: Least squares error approximation of the CR-SF characteristic.

the normalization array should be scaled, so that one obtains a given target compression ratio. This is a major issue, since without any effective strategy the determination of the correct scale factor could take numerous iterations. A solution to this issue is discussed in this and the next sections.

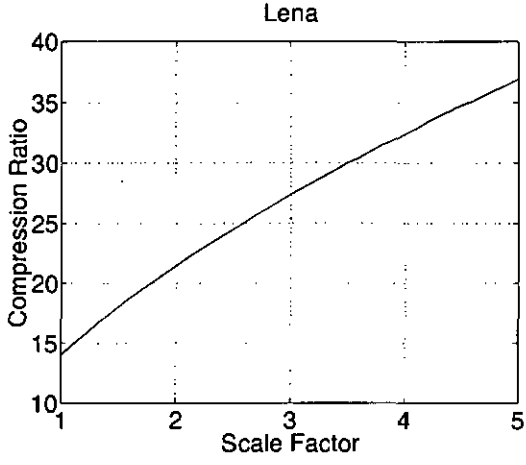
The study began by carrying out intensive JPEG-compression computations on a set of the most popular test images used by the coding community (i.e., Airplane, Baboon, Barbara, Boats, Bridge, Cameraman, Lena, Lena256, Peppers). This set contains images of different spatial resolutions and different scene-complexity content. The procedure was to let the scale factor (SF) change over a large range of values and measure the corresponding obtained compression ratios (CR). The objective being to analyze the characteristic of the curves Compression Ratio–Scale Factor for each of the images on the set.

The obtained CR-SF characteristic is shown in Figures 5.4 and 5.5 for two different image sizes. It is interesting to note the strong linear behavior of these four curves. Even more interesting and significative for the study under question, was the fact of finding out that the same linear behavior is featured by the CR-SF curves of all the images in the study set.

A linear approximation seems thus a plausible choice to model the

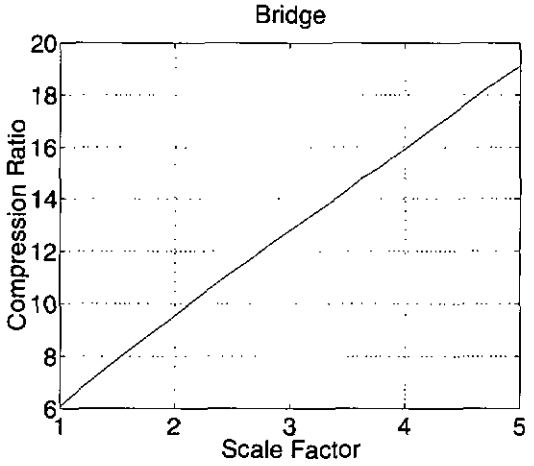


(a)

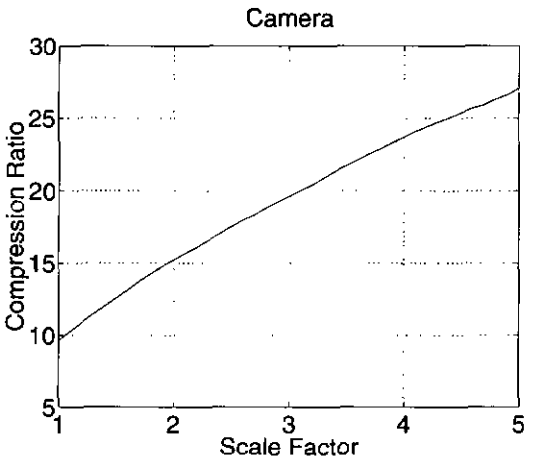


(b)

Figure 5.4: Compression ratio – Scale factor characteristic: 512x512 pixels images.



(a)



(b)

Figure 5.5: Compression ratio – Scale factor characteristic: 256x256 pixels images.

CR-SF characteristic. Table 5.1 shows the results of a linear model built with the classical least squares error LSE method. The R-square value metric[2] is also reported, in order to show the excellent correlation between the real curve and the model (for a perfect fit, $R^2 = 1$). Figure 5.6 shows both the real curve and its linear approximation for the two worst cases of the images in the study set.

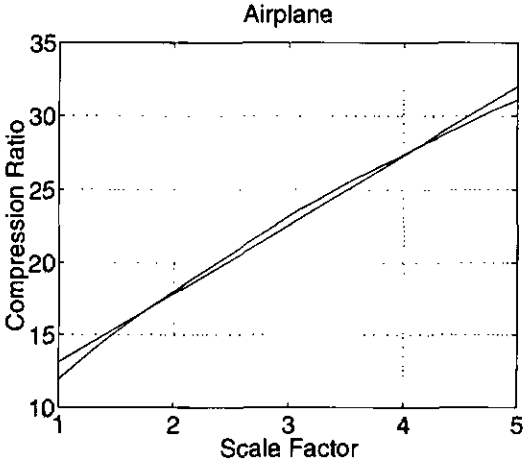
5.5 General Bit-rate Control Scheme

The question raised in the last section was how, given a target compression ratio CR_T, one can predict the scale factor SF_T by which the JPEG normalization matrix must be scaled in order to obtain the requested compression ratio. In this section the general scheme of a method that figures out this problem is presented.

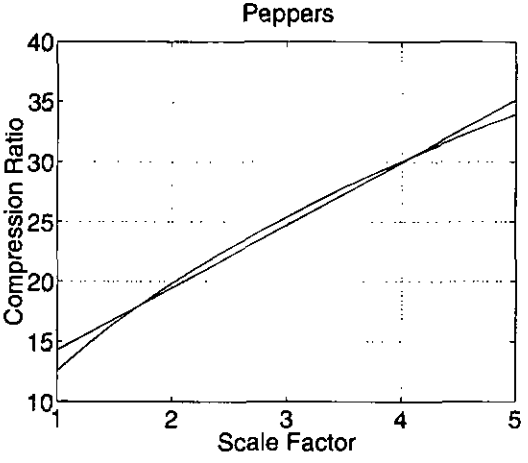
The principle of the proposed bit-rate control method is shown in Figure 5.7. The different blocks of the diagram are described in the following paragraphs.

- Parameter Estimation. To model the CR-SF characteristic with a linear function $y = mx + b$, two parameters are required, namely, the slope m and the intercept b. As can be noticed from Table 5.1 these values are proper for each of the images. It is reasonable to expect that the value of these parameters is related in some way to the information content of the images, and accordingly, one can consider executing some kind of pre-analysis over the image in order to estimate the most suitable values of m and b for a given image.
- Computation of SF_T. The previously determined linear model $cr = m \cdot sf + b$, is used to compute the scale factor target SF_T. That is, given CR_T, its associated scale factor can be found straightforwardly by two simple operations, $\mathrm{SF}_T = (\mathrm{CR}_T - b)/m$.
- JPEG. The value SF_T is used to scale the normalization array; after which the JPEG compression properly said is executed on the input

[2] $R^2 = 1 - \mathrm{SSE}/\mathrm{SST}$, with $\mathrm{SSE} = \sum(x_i - \hat{x}_i)^2$, and $\mathrm{SST} = (\sum x_i^2) - (\sum x_i)^2/n$; n= number of data, x=real data, $\hat{x}$=model data.



(a)



(b)

Figure 5.6: Worst cases LSE approximation of the CR-SF characteristic.

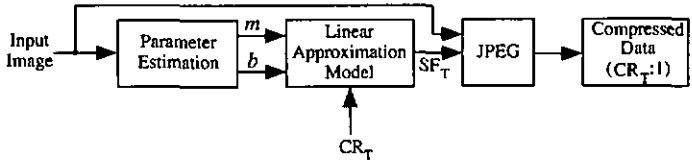


Figure 5.7: General scheme of the bit-rate control method.

image.

Depending on the technique used to estimate the parameters m and b, different bit-rate control implementations are viable for the general scheme in Figure 5.7. Sections 5.6 and 5.7 describe two different methods that were developed.

In both methods, the constraint on the accuracy was defined as to obtaining a relative error, between the target compression ratio and the actual compression ratio given by the system, whose absolute value should be less than five percent.

5.5.1 Range of the scale factor

Looking at the Figures 5.4 and 5.5 on pages 74–75, one observes that the defined range of operation for the scale factor is the interval $1 \leq sf \leq 5$. This range was chosen for the following reasons. For the lower limit $sf = 1$ (the default value of the algorithm), the quality of the reconstructed images is fairly good (as witness the pictures in Figures 5.1(c) and (d)), and so very much appropriated for the most common applications. The upper limit $sf = 5$, was selected by experimentation, concluding that when JPEG-compressing images with a scale factor greater than 5, the quality of the reconstructed images starts to be seriously compromised.

It is important to remark that a small change in the scale factor can have a dramatic effect on the quality of the reconstructed images, and thus, in spite of the relatively short scale factor variation range, the associated range of quality of the reconstructed images is extremely large.

5.6 Bit-rate Control by Average-slope Linear Approximation

In this method the parameter estimation block in Figure 5.7 is replaced by one or two JPEG coding operations. That is, the image is initially compressed with JPEG using a scale factor of 1, for the normalization matrix. Then, the resulting compression ratio is used to (implicitly) estimate the parameter b of the linear model.

By looking over the slope value m for the different linear approximations in Table 5.1, one observes that its range of variation, among the different images, is relatively narrow. This simple observation suggested the construction of an average-slope linear model, in which the value of m is given by the average value of the slopes reported in Table 5.1.

In some instances, the average-slope linear model might not be accurate enough, and thus, a slope-adjustment is necessary, as described in the following section.

5.6.1 Description of the algorithm

The mechanism of operation of this method is shown in Figure 5.8. The different steps of the algorithm are

1. Enter the target compression ratio CR_T.

2. Compress the input image using a scale factor of 1 (the default value) for the JPEG normalization matrix. Let us call CR1 the resulting compression ratio (Figure 5.8(a)).

   - If the absolute value of the relative error between CR_T and CR1 is less than 0.05 then the process stops here. Otherwise it will continue in step 3.

3. With the point (1,CR1) and a given slope m_0, which is determined as the average of the slopes values given in Table 5.1, the value of b_0 is implicitly defined and so is the linear model $cr = m_0 \cdot sf + b_0$.

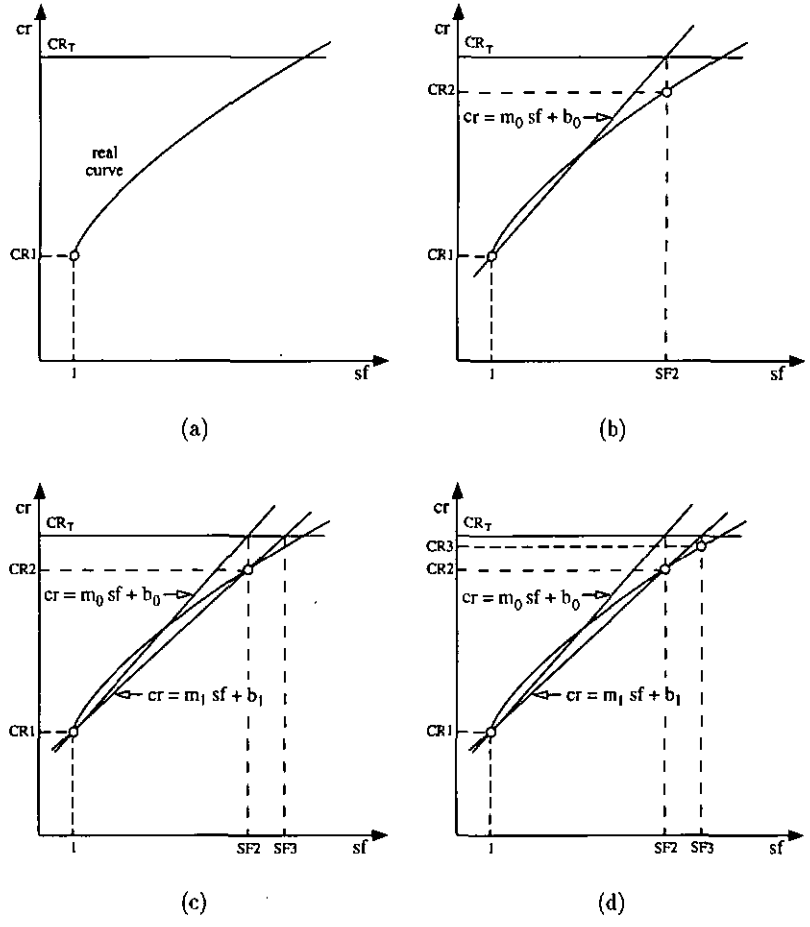


Figure 5.8: Linear approximation method.

Use this model to find the scale factor that corresponds to a compression ratio $cr = \mathrm{CR}_T$. Let us call SF2 this computed scale factor (Figure 5.8(b)).

4. Compress the input image using a normalization array scaled with the value SF2. Let us call CR2 the resulting compression ratio (Figure 5.8(b)).

   - If the absolute value of the relative error between CR_T and CR2 is less than 0.05 then the process stops here. Otherwise it will continue in step 5.

5. The fact that CR2 failed to approximate CR_T, at the end of the step number 4, indicates that the average-slope model is not fitting accurately the CR-SF characteristic of the input image. In this case a slope (and an intercept) adjustment are necessary in order to satisfy the maximum relative error constraint of plus/minus five percent.

   Both (1,CR1) and (SF2,CR2) are points on the real CR-SF curve. As such, they are appropriated to develop a more accurate approximation that is given by $cr = m_1 \cdot sf + b_1$. In this updated model, find the scale factor that corresponds to a compression ratio $\mathrm{cr} = \mathrm{CR}_T$. Let us call SF3 this computed scale factor (Figure 5.8(c)).

6. Finally, using SF3 to scale the JPEG normalization array, compress the input image. Let us call CR3 the resulting compression ratio (Figure 5.8(d)).

5.6.2 Validation of the average-slope linear model

Exhaustive testing with pictures from and out of the initial set of images, showed that the value of CR3 is an excellent approximation of CR_T (absolute value of the relative error < 0.05). The results are given in Figures 5.9–5.11 for several images. The graphs in Figures 5.11(c) and (d) belong to a picture that was not part of the initial set of test images (which are listed in Table 5.1).

For each of the graphs, the first value on the x-axis corresponds to the target compression ratio whose associated scale factor is $sf = 1$. The last value corresponds to a scale factor of 5.

5.6.3 Improving the average-slope linear model method

Though the bit-rate control method described in the last section gives very good results, it is nevertheless susceptible of improvement. The following are two important ameliorations that are worthy of consideration.

1. *Reduce the number of iterations.* When the initial average-slope model does not fit accurately the CR-SF curve, an extra JPEG iteration is required in order to compute CR3 with an updated model $cr = m_1 \cdot sf + b_1$ (Figure 5.8(c) and (d)). It would be of great advantage, from a computational complexity point of view, to make the bit-rate control system converge to the value of CR_T—with the same predefined accuracy—within a maximum of two JPEG operations. That is to say that CR_T should be accurately approximated, with no further computations beyond the calculation of CR2. This goal is achieved with the *two-pass method*, whose results are reported in Section 5.8.1.

2. *Extended range of operation for the scale factor.* Though the range of operation previously defined for the scale factor, is fairly enough for a large number of applications, it would be interesting to have a system that is able to control the compression bit rate of JPEG for the largest "practical" range of applications. This goal is achieved with the *three-pass method*, whose results are reported in Section 5.8.2. In this method, the range $1 < sf \leq 5$ is extended in both directions.

   EXTENDED RANGE FOR THE SCALE FACTOR

   Experimentally, the extended scale factor range, $0.5 < sf \leq 15$, was considered for the following reasons.

   - *Lower limit.* The quality of the reconstructed images given by default ($sf = 1$) by JPEG is in general satisfactory; still, if nec-

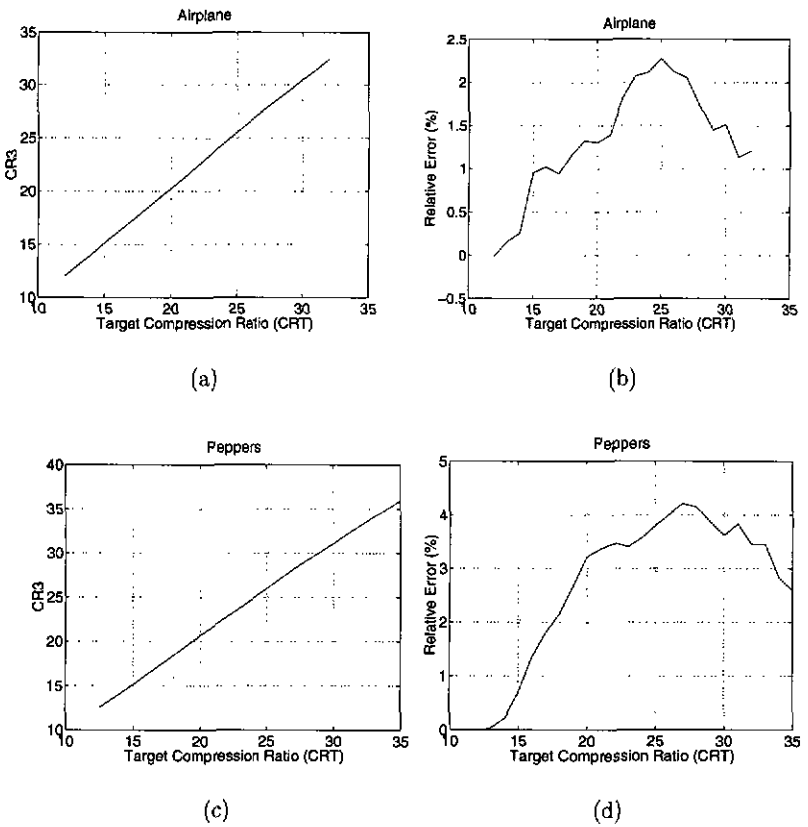


Figure 5.9: Results for the images Airplane and Peppers.

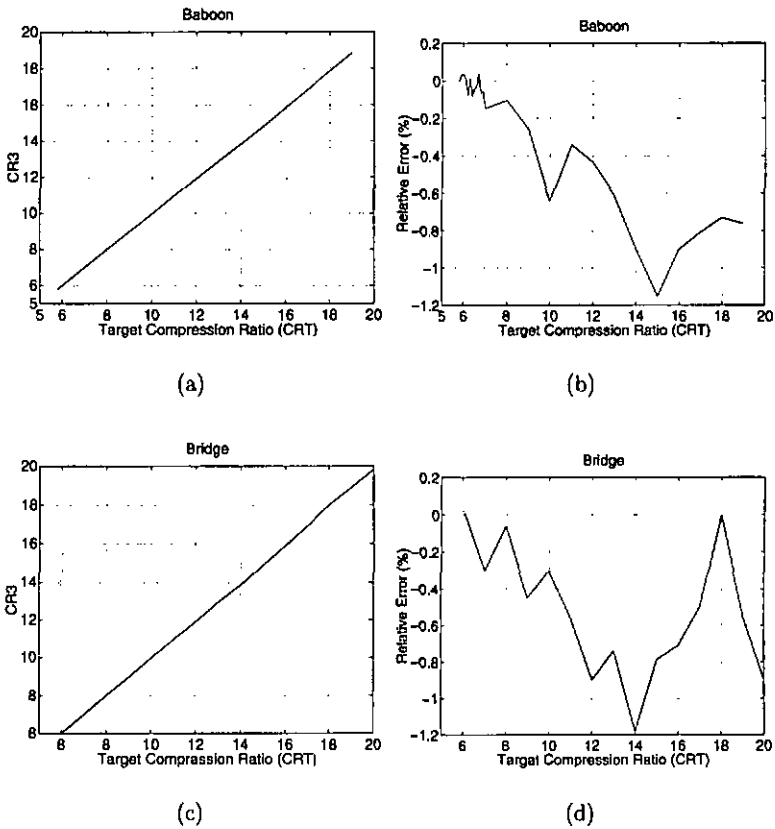


Figure 5.10: Results for the images Baboon and Bridge.

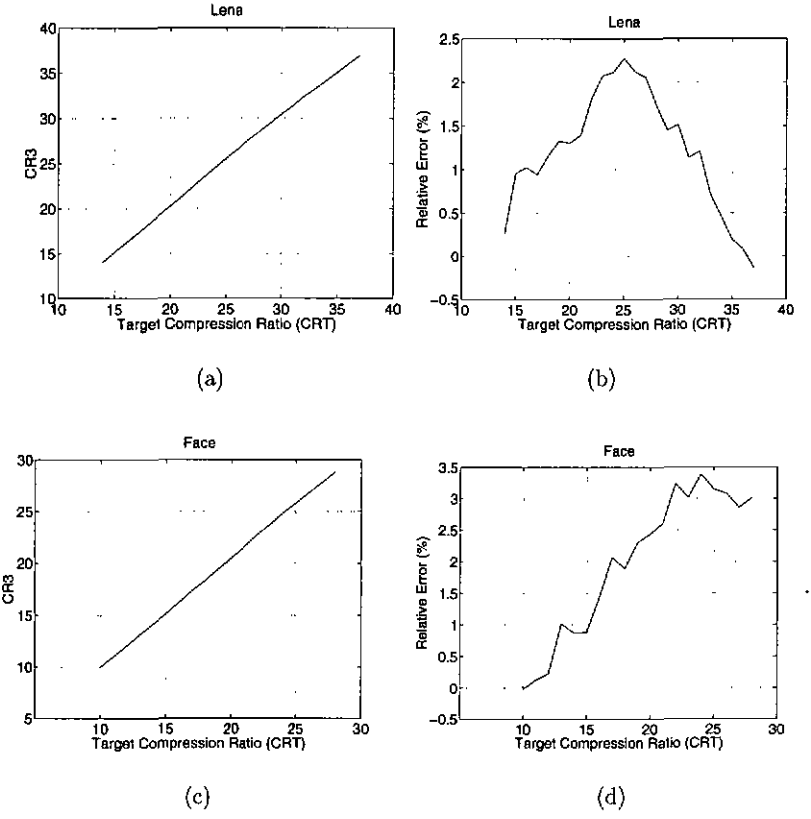


Figure 5.11: Results for the image Lena and Face.

essary, this quality can be upgraded by using a scale factor that is less than the unity. In effect, when the normalization array is scaled by a factor that is less than one, the effect on the resulting compression bit rate would be the opposite than in the example given in Section 5.3, that is, a decrease of the compression ratio. This loss of compression efficiency is however obtained in favor of an improvement of the quality of the reconstructed images. The reason for choosing $sf = 0.5$ as the lower limit, is because at this value the original and the reconstructed images are normally indistinguishable ([44], Chapter 4). Making thus worthless any attempt to improve the quality of the decoded images below a scale factor of 1/2.

- *Upper limit.* Experiments on a large set of images proved that the quality of the reconstructed images beyond a scale factor of 15, is strongly objectionable. Decoded images with $sf > 15$, are generally useless for most applications (artwork excluded).

5.7 Bit-rate Control by Piecewise-linear Approximation

5.7.1 Piecewise-linear model

The CR-SF characteristic, on the extended scale factor range, for all the test images was analized. It was interesting to note that the behavior among the curves was again very similar; nonetheless, a single average-slope model for the entire new range was no longer valid.

A typical CR-SF characteristic is given in Figure 5.12 for the range $0.5 \leq sf \leq 15$. The second bit-rate control method consists in approximating this curve not with a single, but with multiple straight lines whose slopes change in function of their region in the scale factor space. Furthermore, the slope associated with each region, is not constant but linearly depends on the value of the compression ratio obtained with a scale factor $sf = 2$.

Six different regions of the CR-SF curves were found to be suitable

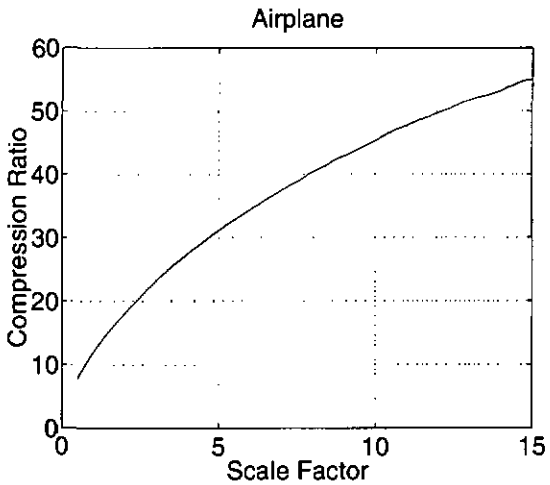


Figure 5.12: Compression ratio – Scale factor characteristic.

for separate modeling. The intervals of the scale factor that correspond to these regions are: $[0.5, 1)$, $[1, 1.5)$, $[1.5, 2)$, $(2, 5]$, $(5.10]$ and $(10, 15]$. The interval $0.5 \leq sf \leq 2.0$ is the interval in which the first derivate of the CR-SF curve is changing at the highest rate, thus, this region was divided into shorter sub-intervals than the less derivate-changing region $2 < sf \leq 15$. The point at $sf = 2$ is, as will be explained below, the starting step of the algorithm, for this reason it is not included in any of the regions above.

It can be noticed that the modeling of the CR-SF characteristic has become more complex with respect to the *single* straight-line model $y = mx + b$. That is, now the system must be able to predict the slope m_i and the intercept b_i for the different six regions that have been defined.

The problem of the slope estimation for each of the different regions of the piecewise-linear model, was solved by studying the relationship between the value of the compression ratio (at different scale factors) and the value of the slope for each of the regions. It turned out that a strong linear relationship exist between the slope and the compression ratio CR1 obtained with a scale factor $sf = 2$. Hence, the estimation

Region No.	Scale Factor Range	m:*CR1* Relation
1	$0.5 \leq sf < 1.0$	$m_1 = 0.4939 \cdot CR1 - 0.7064$
2	$1.0 \leq sf < 1.5$	$m_2 = 0.3947 \cdot CR1 - 0.3122$
3	$1.5 \leq sf < 2.0$	$m_3 = 0.2894 \cdot CR1 + 0.6224$
4	$2.0 < sf \leq 5.0$	$m_4 = 0.1565 \cdot CR1 + 1.6517$
5	$5.0 < sf \leq 10$	$m_5 = 0 \cdot CR1 + 3.0175$
6	$10.0 < sf \leq 15.0$	$m_6 = -0.1098 \cdot CR1 + 3.8832$

Table 5.2: Slope determination for the different regions.

parameters block in Figure 5.7 is replaced by a JPEG operation in which the normalization array is scaled by a factor of 2. The relation m : CR1 for the different regions is given in Table 5.2.

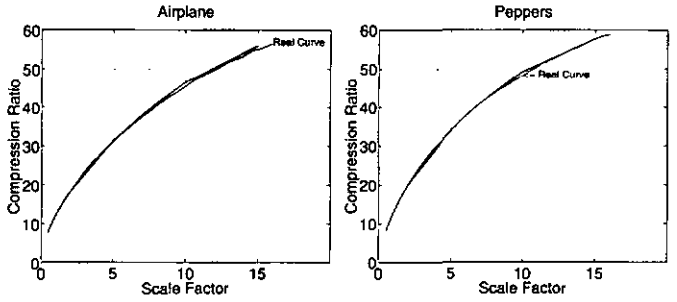
The good fitting of the piecewise-linear model to the real curve is shown in Figure 5.13 for the two worst cases of all the images in the original set.

5.7.2 Description of the algorithm

The operation mechanism of the piecewise-linear approximation method is shown in Figure 5.14. Though very simple, the complete description of the algorithm is rather repetitive and lengthy due to the multiple conditional branchings. In this section the method is reported for the particular case when the obtained compression ratio lies in the region $5 < sf \leq 10$. This case involves all the main operations of the procedure and perfectly illustrates the mechanism of the algorithm; which is given in full description in Appendix B.

The different steps of the algorithm are:

1. Enter the target compression ratio CR_T.
2. Compress the input image using a scale factor of 2, for the JPEG normalization array. Let us call CR1 the resulting compression ratio (Figure 5.14(a)).
   - If the absolute value of the relative error between CR1 aud CR_T



(a) (b)

Figure 5.13: Piecewise-linear LSE approximation.

is less than 0.05 then the process stops here. Otherwise it will continue in step 3.

3. If (CR_T > CR1) then go to step 4 otherwise go to step 11.

4. Insert the value of CR1 in the equation m : CR1 of region No. 4 in Table 5.2, in order to find the slope of the linear model for the region $2 < sf \leq 5$. Having the value m_4 and the point (2,CR1), the intercept b_4 is implicitly defined and so is the linear model $cr = m_4 \cdot sf + b_4$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = CR_T$. Let us call SF2 this computed scale factor (Figure 5.14(a)).

   - If (SF2 > 5) then go to step 5 otherwise go to 10.

5. Use the linear model $cr = m_4 \cdot sf + b_4$, in order to find the compression ratio CR_V that corresponds to a scale factor $sf = 5$. This operation defines the point (5,CR_V) (Figure 5.14(b)).

   Insert the value of CR1 in the equation m : CR1 of region No. 5 in Table 5.2 in order to find the slope m_5 of the linear model for the region $5 < sf \leq 10$. Having the value m_5 and the point (5,CR_V), the intercept b_5 is implicitly defined and so is the linear

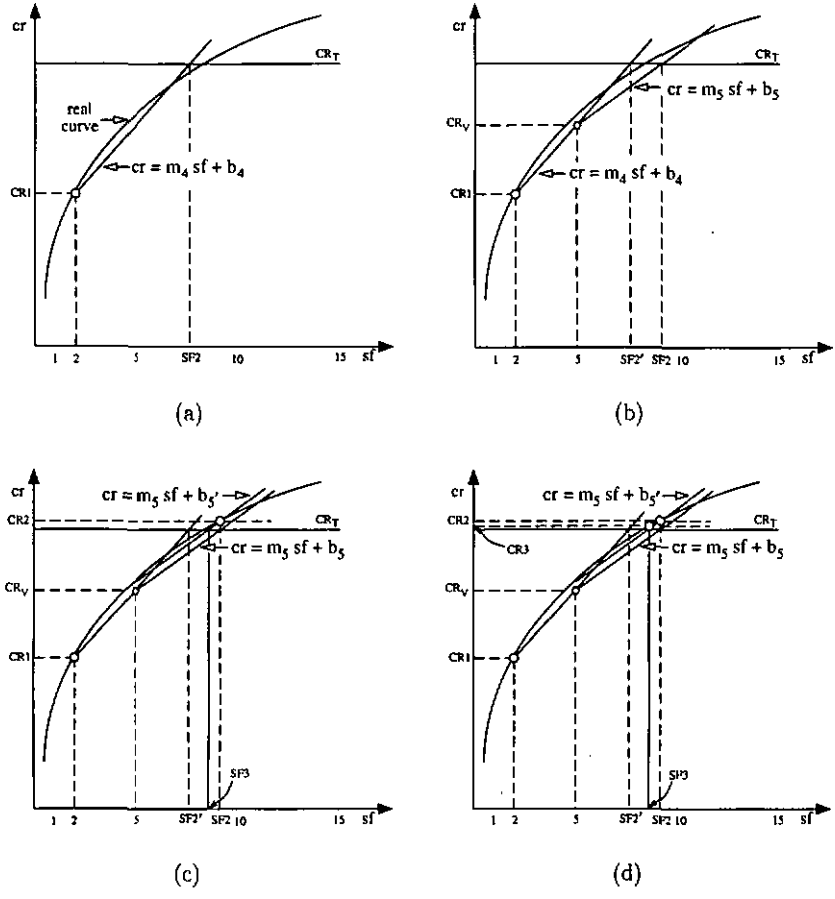


Figure 5.14: Piecewise-linear approximation method.

model $cr = m_5 \cdot sf + b_5$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = \mathrm{CR}_T$. Let us call SF2 this computed scale factor (Figure 5.14(b)). The previous value of SF2 is overwritten, becoming SF2'.

- If (SF2 > 10) then go to step 9 otherwise go to step 6.

6. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio (Figure 5.14(c)).
   - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise it will continue in step 7.
7. In order to appropriately approximate CR_T an intercept adjustment is necessary. The slope m_5 and the point (SF2,CR2) implicitly define a more accurate model $cr = m_5 \cdot sf + b_{5'}$. In this updated model, find the scale factor that corresponds to a compression ratio $cr = \mathrm{CR}_T$. Let us call SF3 this computed scale factor (Figure 5.14(c)).
8. Finally, using SF3 to scale the JPEG normalization array, compress the input image. Let us call CR3 the resulting compression ratio (Figure 5.14(d)).
9. This step is explained in the complete description of the algorithm in Appendix B.
10. This step is explained in the complete description of the algorithm in Appendix B.
11. This step is explained in the complete description of the algorithm in Appendix B.

5.8 Validation of the Piecewise-linear Model

The piecewise-linear approximation method can be implemented in two forms. A simple and yet suitable for a large number of applications, two-pass method, ends after the calculation of CR2 in Figure 5.14(c)). The

more general three-pass method implementation, operates on a larger scale factor range, as described in the previous section, and requires an extra quantization and an extra entropy coding operation with respect to the two-pass method (since the 2-D DCT coefficients are always the same regardless of the number of iterations, the 2-D DCT operation is executed only once).

5.8.1 Two-pass method

Exhaustive testing with pictures from and out of the initial set of images, showed that when the piecewise-linear model algorithm operates in the range $1 \leq sf \leq 5$ the value of CR2 is an excellent approximation of CR_T, within the defined relative error constraint. The results are given in Figures 5.15 to 5.17 for several images. The graphs in Figures 5.17(c) and (d) belong to a picture that was not part of the initial set of test images.

5.8.2 Three-pass method

Exhaustive testing with pictures from and out of the initial set of images, showed that the value of CR3 is an excellent approximation of CR_T (absolute value of the relative error < 0.05), all along the range $0.5 \leq sf \leq 15$. The results are given in Figures 5.18 to 5.20 for several images.

5.9 Computational Complexity

Due to its better performance with respect to the average-slope linear model, the following discussion will be limited to the computational complexity of the piecewise-linear model.

The computational complexity analysis will be separated into two parts. Part one, particular requirements, concerns the computational resources that are required to find the target scale factor and that are proper to the method proposed in this chapter. Part two, general re-

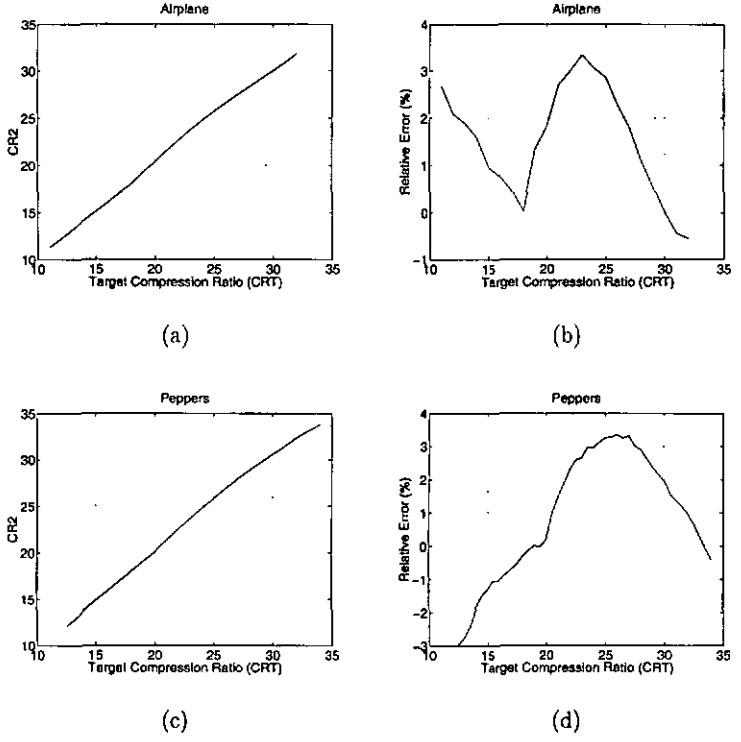


Figure 5.15: Results for the images Airplane and Peppers.

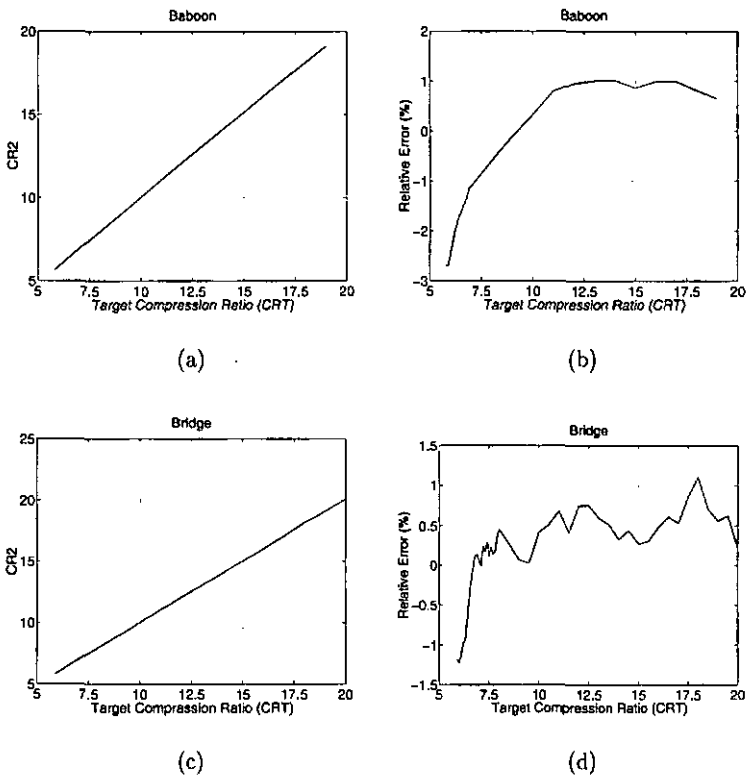


Figure 5.16: Results for the images Baboon and Bridge.

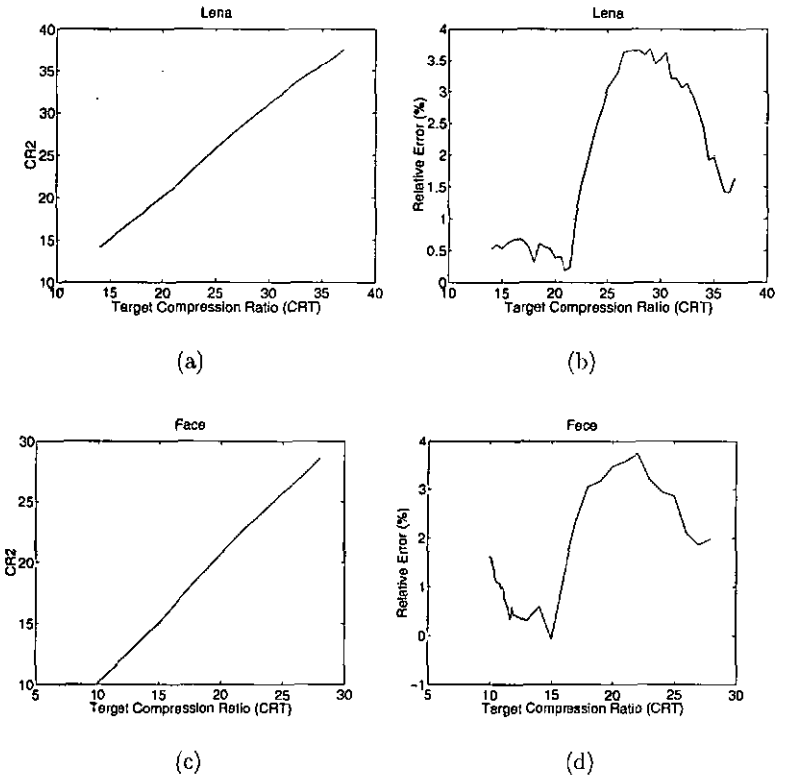


Figure 5.17: Results for the images Lena and Face.

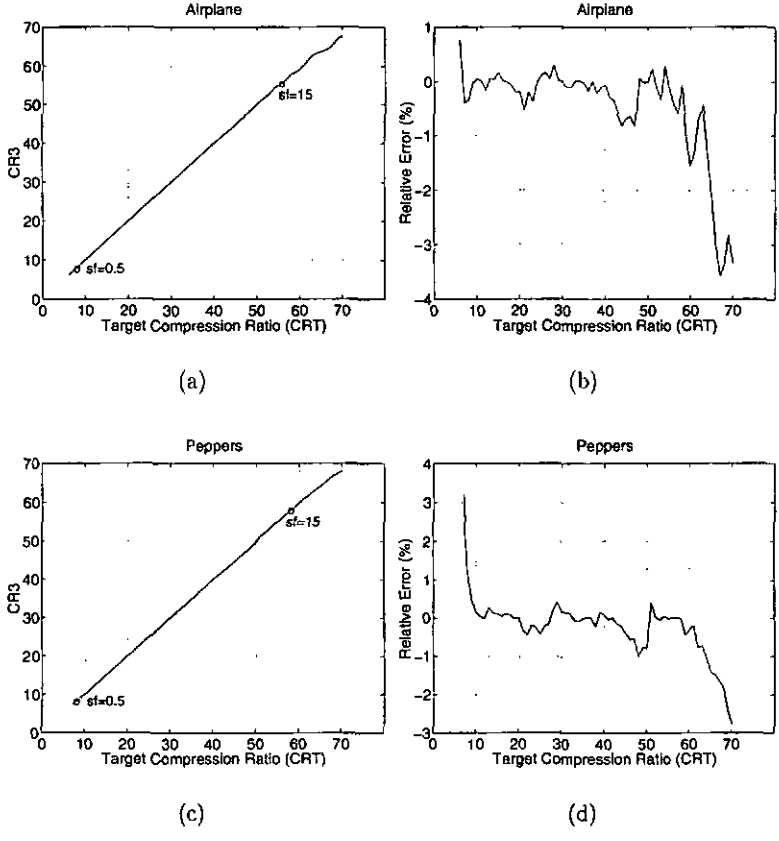


Figure 5.18: Results for the images Airplane and Peppers.

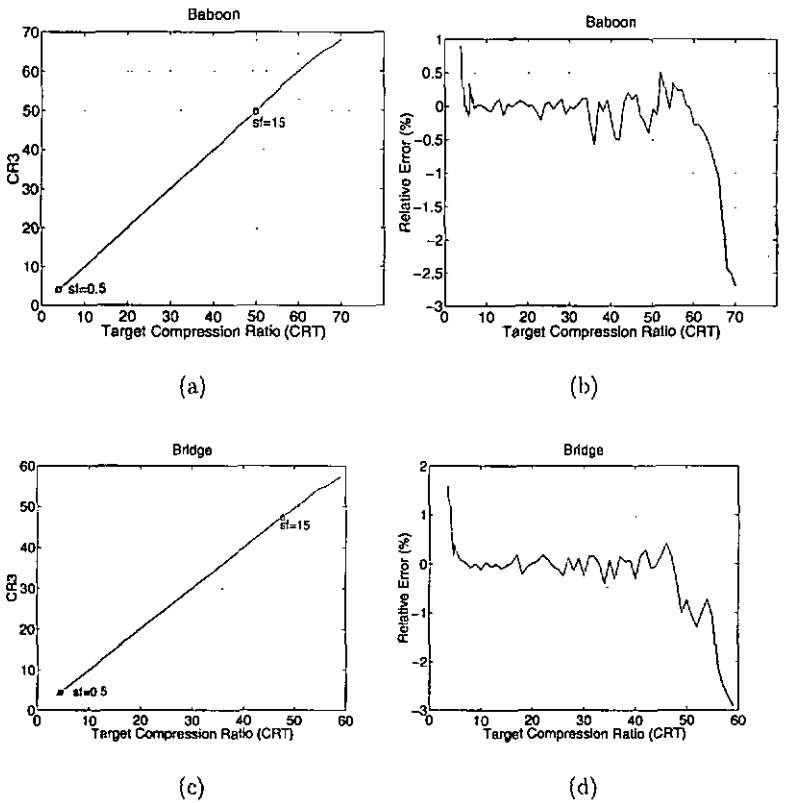


Figure 5.19: Results for the images Baboon and Bridge.

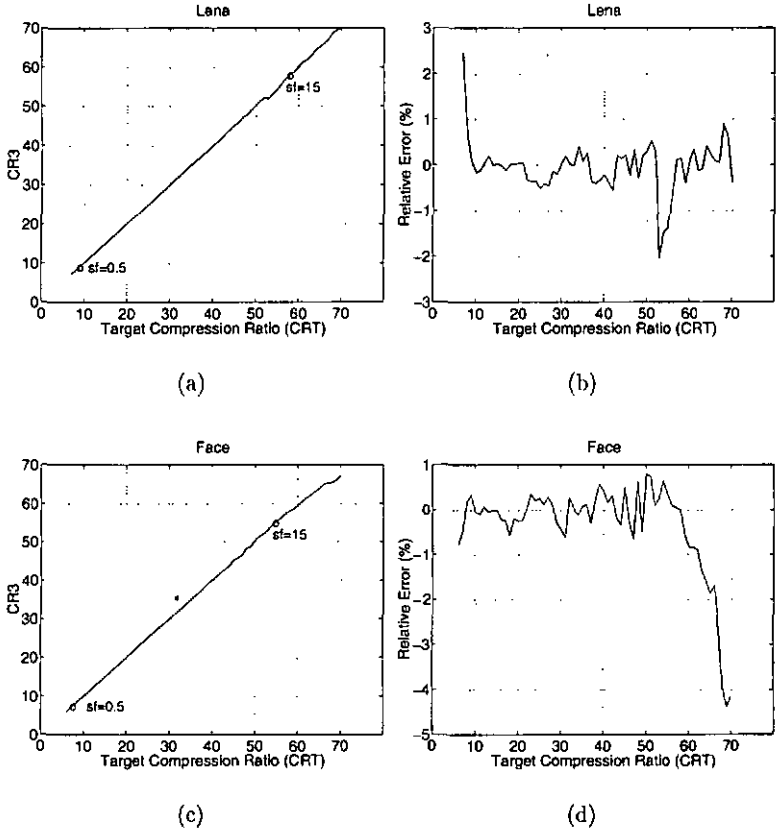


Figure 5.20: Results for the images Lena and Face.

quirements, concerns the computational resources that are required due to the repetitive computations executed during the iterations.

To simplify, in the following discussion the terms DCT, quantization, and entropy coding, will be used to indicate the respective global operations applied over the entire image (and not over a pixel or block of pixels). For example, for an image of $(M \times N)$ pixels, *one* quantization will stand for the execution of the corresponding $(M \times N)$ rounded divisions (or multiplications).

5.9.1 Two-pass method

- *Particular requirements*: To find the slope m_i that best fits the CR-SF characteristic in the regions 2, 3 and 4 defined in Table 5.2, a function $m_i = m_{i0} \cdot CR1 + b_{i0}$ must be evaluated. In order to store the different m_{i0}'s and b_{i0}'s, six units of permanent memory are required. Working memory, is required to temporarily store the values of CRT, CR1 and SF2.

  The number of operations will be computed with respect to the worst-case (from a computational complexity point of view), which happens when the final compression ratio lies in the region $1.0 < sf \leq 1.5$. One comparison indicates that the target region is either region No. 3 or 2. Given the value of CR1, the slope m_3 is calculated with the appropriate equation given in Table 5.2, requiring 1 addition and 1 multiplication.

  The first SF2 (later SF2') is computed as $\mathrm{SF2} = 2+(\mathrm{CR}_T - \mathrm{CR1})/m_3$. One more comparison indicates that the target region is region No. 2. The computation of CR_{II} demands 1 division, 1 addition and 1 subtraction. In order to find the slope m_2, 1 addition and 1 multiplication are required.

  Finally, to find the second (and target) SF2, 1 division, 1 addition and 1 subtraction are necessary.

- *General requirements*: One JPEG operation (DCT + Quantization + Entropy coding) is required in order to compute CR1. Once the

Operation	*Number*
Quantization	1
Entropy Cod.	1
Division	3
Multiplication	66
Additions	5
Subtractions	3
comparisons	2
Memory	*Requirement*
Permanent	6 coeffs.
Working	3 coeffs.

Table 5.3: Computational complexity of two-pass method.

target scale factor SF2 has been determined, the normalization matrix is scaled, requiring 64 multiplications. Then one re-quantization of the 2-D DCT coefficients with the *new* normalization array is executed. Finally, an additional entropy coding operation is required to produce the compressed bit stream.

It is important to note that the 2-D DCT operation, which is the most computationally intensive operation of JPEG, could, if memory is not an issue, be executed only once. Accordingly, working memory (RAM or bank of registers) would be required in order to store the 2D-DCT coefficients for the time necessary to execute the second iteration. The size of this image buffer obviously depends on the spatial resolution of the input image.

Table 5.3 summarizes the total number of operations and memory requirements for the piecewise-linear approximation two-pass method. The number of operations given is per image, which for the quantization and entropy coding depends on the space resolution of the input image. The operations due to the default JPEG compression are not shown. If the implementation option described in the previous paragraph is chosen, then the corresponding working memory should be added to this table, to store the 2-D DCT coefficients.

5.9.2 Three-pass method

- *Particular requirements*: To find the slope m_i that best fits the CR-SF characteristic in all the regions defined in Table 5.2, a function $m_i = m_{i0} \cdot CR1 + b_{i0}$ must be evaluated. In order to store the different m_{i0}'s and b_{i0}'s, twelve units of permanent memory are required. Working memory, is required to temporarily store the values of CRT, CR1, CR2, SF2 and SF3.

  The number of operations will be computed with respect to the worst-case (from a computational complexity point of view), which happens when the final compression ratio lies in the region $0.5 < sf \leq 1.0$ (the region $10 < sf \leq 15$ is equally a worst case region).

  One comparison indicates that the target region is either region No. 3, 2 or 1. Given the value of CR1, the slope m_3 is calculated with the appropriate equation given in Table 5.2, requiring 1 addition and 1 multiplication.

  The first SF2 (later SF2′) is computed as $SF2 = 2+(CR_T-CR1)/m_3$. One more comparison indicates that the target region is region No. 2. The computation of CR_{II} demands 1 division, 1 addition and 1 subtraction. In order to find the slope m_2, 1 addition and 1 multiplication are required. The computation of the second SF2 (later SF2″) requires again 1 division, 1 addition and 1 subtraction. One additional comparison indicates that the target region is region No. 1. The computation of the point CR_I involves 1 division, 1 addition and I subtraction.

  The final SF2 is calculated and used to compress the input image, which produces the value of CR2. A comparison indicates that CR_T is not approximated with the predefined precision and that the computation of CR3 is necessary. In order to do so, an intercept adjustment ($b_1 \rightarrow b_{1'}$) is necessary, whose new value is implicitly known by having the point (SF2,CR2). Finally, on the updated model the target scale factor SF3 is calculated by means of 1 division, 1 addition and 1 subtraction.

Operation	*Number*
Quantization	2
Entropy Cod.	2
Division	6
Multiplication	130
Additions	8
Subtractions	6
comparisons	4
Memory	*Requirement*
Permanent	12 coeffs.
Working	5 coeffs.

Table 5.4: Computational complexity of three-pass method.

- *General requirements*: One JPEG operation (DCT + Quantization + Entropy coding) is required in order to compute CR1. The normalization matrix is scaled twice, once with the scale factor SF2, and once with the scale factor SF3, which demands 2x64=128 multiplications. After each of these scalings, a quantization of the 2-D DCT coefficients is carried out. Each quantization is always followed by an entropy encoding. The first iteration produces the value of CR2, while the second produces the final bitstream representing the compressed image at an excellent approximation CR3, of the requested compression ratio.

Table 5.4 summarizes the total number of operations and memory requirements for the piecewise-linear approximation three-pass method. The number of operations given is per image, which for the quantization and entropy coding depend on the space resolution of the input image. The operations that correspond to the default JPEG compression are not included. The same comment that was made at the very end of Section 5.9.1, regarding the possible requirement of additional of working memory, holds true for Table 5.4.

5.10 Summary

In this chapter two methods for controlling the compression ratio given by JPEG have been presented. A study of the characteristic of the compression ratio (CR) versus the scale factor (SF) has been reported. The results of this study were used to build a linear and a piecewise-linear model of the CR-SF characteristic for any given image. The first reported method consisted in defining a single average-slope model; this first approach is then improved by using a more elaborated piecewise-linear model that requires one less iteration to give the same results. Finally, a more general bit-rate control method was defined on an extended range for the scale factor. The validations of all these method have beens equally reported, showing the effectiveness of these bit-rate control techniques.

Chapter 6

Adaptive Block-Size Transform Coding for Image and Video Compression

6.1 Introduction

Transform-based image coding algorithms have been the object of intense research during the last twenty years. Eventually they have been selected as the main mechanism of data compression of current digital image and video coding standards. For example JPEG, MPEG-1, MPEG-2, H.261 and H.263 standards all rely on an algorithm based on a linear transformation of the input image/video data.

As already mentioned in previous chapters, a transform-based image coding method involves subdividing the original image into smaller $N \times N$ blocks of pixels and applying a unitary transform, such as the Discrete Cosine Transform (DCT), on each of these blocks. A plethora of methods have been proposed regarding different kinds of processing executed after the transform operation. Normally, once the value of N has been selected for a particular algorithm, it remains fixed. In JPEG for instance, the value of N is 8, and accordingly, the input image is divided into blocks of 8x8 pixels *exclusively*.

Several *adaptive* transform-based methods are described in [8, 9, 106]. Most of these methods are *fixed* block-size schemes in which an adaptive quantization of the transform coefficients is made. A variable

block-size technique has been reported in [107], in which it is also claimed that it was one of the first departures from traditional *fixed* to *variable* block-size schemes. In their approach the transform coefficients are processed by vector quantization.

In this chapter an adaptive block-size transform coding scheme based on the JPEG and H.261 algorithms is presented and a mechanism to significantly reduce its computational complexity is also proposed [108]. The reported results show the improvement of the compression efficiency with respect to the non-adaptive schemes.

6.1.1 Organization of the chapter

The remainder of this chapter is organized as follows. A review of adaptive transform coding techniques is presented in Section 6.2. Then, a discussion on the issue of choosing the size of the basic block ($N \times N$) in transform coding is made in Section 6.3. The adaptive scheme proposed in this chapter is described in Section 6.4, followed by a discussion on the selection of its classification parameters on Section 6.5. The results and performance of the adaptive scheme are reported in Section 6.6. The computational complexity of the system, and strategies to reduce the number of operations are analyzed in Section 6.7. Finally, a summary of the chapter is given in Section 6.8.

6.2 Adaptive Transform Coding Methods

Research on transform-based adaptive techniques for image coding has been very prolific, even since the early studies on image compression. Reference [109] presents one of the first reviews on the topic, albeit not all the reported methods therein are transform-based. Later, the same author proposed a fixed block-size adaptive transform coding scheme [110], in which the transform coefficients are DPCM encoded.

A commonly referenced work on adaptive transform-based techniques is reported in [111]. In this method the input image is divided into blocks of 16x16 pixels, each block is transformed by means of a

16-point DCT, and the resulting 16x16 DCT coefficients are then quantized adaptively by using one of four different bit allocation matrices. The bit allocation matrix to be used is selected in function of the image activity of the original block. This method requires two passes to code the image (the first pass is required among other things to set up the bit assignments matrices), which in real time implementations produces a coding delay of one frame. A similar 16x16 scheme is presented in [112], in which the one frame delay is suppressed.

6.2.1 Adaptive TC/VQ

Reports on adaptive transform coding (TC) with vector quantization (VQ) have been eqnally abundant. In [113] the input image is divided into blocks of 8x8 pixels, after an 8-point DCT operation, several subvectors are formed out of the 64 DCT coefficients. These vectors are adaptively coded by using different codebooks, the right codebook is chosen depending on the energy content of the elements of the vector.

Two similar adaptive DCT/VQ methods are reported in [114, 115]. In both schemes after an 8-point DCT operation, the original 8x8-pixel block is categorized according to the region in which are located the DCT coefficients with the maximum magnitudes. Depending on the category and on the energy of the AC coefficients, subvectors are formed adaptively from different zones of the 2-D matrix of DCT coefficients. Other adaptive DCT/VQ schemes have been reported in [116, 117].

6.2.2 Variable block-size schemes

The methods referenced in the previous sections are all fixed block-size schemes in which the 2-D DCT coefficients are quantized or encoded by following an adaptive scheme. In [107] is presented an approach in which the original image is divided into blocks of different sizes. The smallest block size, is coded directly by using classified vector quantization [118], while the rest of the block sizes are coded by using a combination of DCT followed by VQ. It is claimed in the paper, that their first report on the

topic [119] was one of the first approaches in the *variable* block-size sense, to which followed some adaptations of the original scheme [120].

The results presented in all these adaptive block size schemes were exclusively oriented to the still image coding application.

6.3 Selection of the Block Size in Transform Coding

6.3.1 Interpixel correlation

The similarity among the values of the pixels that compose the different regions of an image makes it possible and/or easier to reduce the original amount of data required to represent the image. In general, this interpixel correlation is inversely proportional to the distance separating the pixels: the closer the pixels are, the higher the resemblance is among their gray level values.

A linear transformation of a given image is efficient for data compression for as long as the $N \times N$ pixels on which the transform is applied are highly correlated. This fact suggests that choosing a block size $N \times N$ that is too large is not advantageous, since the stationarity of the image in such block sizes is not very likely.

6.3.2 Energy packing

Though choosing a small block size $N \times N$ has a pragmatic appeal in accordance with the account in the previous section, selecting too small a block size is not the most appropriate either. Coined in [122], energy packing is the term associated with the property of a linear transform of distributing most of the input signal energy into the smallest number of transform coefficients; and it is well known from several reports [123] that as the value of N decreases so does the energy packing efficiency of the N-point transform. In this sense, since the level of data compression of a system is strongly dependent on this packing efficiency, the larger the chosen block size, the better.

It was reported in [121] that a significant data correlation is obtained only within a region of 20 adjacent pixels, which hints a practical

Block Size	Pixel Correlation	Memory Requirements	Computational Complexity	Energy Packing
4x4	√√√	√√	√√	X
8x8	√√	√	√	√
16x16	√	X	X	√√
32x32	X	XX	XX	√√√

Table 6.1: Comparing different block sizes for transform coding.

upper limit for the block size. The values of N that have proved to be of practical interest for data compression applications are: 4, 8 and 16. These values are normally chosen to be an integer power of two, because that allows an efficient fast implementation of the transform [124–126].

6.3.3 Computational complexity

Among these candidate values for N, and for general purpose image coding systems, the issue of selecting the block size is simplified when the computational complexity associated with the implementation of the N-point linear transform is taken into account. Indeed, the substantial growth of the computational complexity and memory requirements associated with increasing values of N, in some instances overshadows its associated outperformance in terms of compression efficiency.

For example, this was the case for the question of selecting a block size for the JPEG algorithm, where both N=8 and N=16 were considered. It turned out that the significant increase of the computational complexity and memory requirements of a 16-point DCT seemed not to justify its slightly higher compression ratios. This analysis resolved in favor of the now familiar JPEG's block size of 8x8 pixels.

Table 6.1 presents a summary of the advantages (√) and disadvantages (X) of using four $N \times N$ different block sizes for transform coding applications. The computational complexity and memory requirements entries are related to the complexity of implementation of the associated N-point linear transform

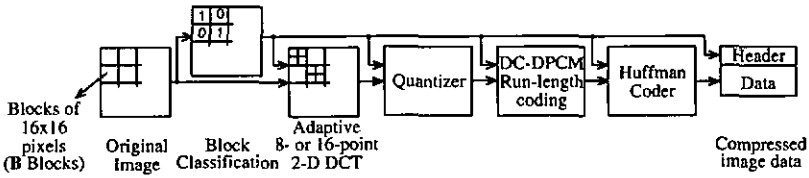


Figure 6.1: Adaptive JPEG-based algorithm.

6.4 Adaptive Block-size Scheme

A block diagram of the proposed adaptive block-size scheme is shown in Figure 6.1. It is based on the sequential JPEG algorithm, and it is described in the following paragraphs.

The input image is divided into blocks of 16x16 pixels, which will be referred to as **B** blocks in the context of this chapter. On each of these blocks a metric is evaluated in order to determine its degree of image activity (i.e., the rate of variation of the luminance levels of the pixels in a block). If the resulting measure is below a predefined threshold, then the block is classified as a **0** block, identifying it as a smooth-textured low-varying luminance block. Otherwise, it is classified as a **1** block.

A DCT module receives the **B** blocks along with their classification bit. The DCT module executes a 16-point 2-D DCT on those **B** blocks that have been classified as **0** blocks. The **B** blocks classified as 1 blocks are further divided into 4 subblocks of 8x8 pixels each, and then an 8-point 2-D DCT is applied on each of the four subblocks.

The 8- or 16-point 2-D DCT coefficients are quantized, according to their classification bit, by an 8x8 or a 16x16 normalization array. After quantization, the DCT coefficients are zigzag reordered in order to increase the sequences of zero-valued coefficients as described in Chapter 2.

The DC coefficients are DPCM coded, by using the prediction value path shown in Figure 6.2, where each dot represents the DC coefficient corresponding to each block. The other possible paths for this purpose were equally evaluated, all giving nearly the same results. The AC coefficients are run-length coded and finally, the resulting DC and AC symbols

Figure 6.2: Prediction path for the DPCM.

are Huffman coded. Obviously, the block classification bitmap comes to form part of the header information.

The rationale underlying the adaptive scheme is to provide a means to take advantage of large uniform regions within the image to be coded. Thus, instead of coding those regions with four blocks of 8x8 pixels, as it would be done by a fixed block scheme, the system will code them as a single block of I6xI6 pixels. By doing so, the better energy packing efficiency of a 16-point DCT is exploited in a region in which a high interpixel correlation is assured. This results in a significant improvement of the compression efficiency of the system.

The adaptive scheme can be applied to code still images, or it can be embedded in a video coding system in order to code the (prediction error) frames of a video sequence. Depending on the application, a different set of parameters must be chosen for the adaptive method. The requirement of an application oriented parameter selection is simply related to the fact that natural still images and prediction error frames are image-data of different structure. In this chapter, emphasis will be given to the results of addressing the video application.

6.5 Selection of the Classification Parameters

6.5.1 Two-level classification

One sensitive drawback of most adaptive image coding algorithms is the generation of overhead information. Indeed, part of the compressed bit stream is dedicated to convey control bits that indicate the decoder the mode of operation that was used to compress the image/video data. In general, the higher the degree of adaptability of a coding algorithm, the larger the amount of overhead data that needs to be sent to the decoder. Thus, the number of bits allocated to identify the operation

modes should be as few as possible, so as not to compromise the coding efficiency of the algorithm.

In an adaptive block-size scheme, the overhead information is related to the number of different block sizes and to the segmentation structure of the different classes. Thus, using a minimum number of block-sizes, and the same regular structure at every hierarchical level, should contribute to minimize the overhead information.

For the adaptive scheme reported in this chapter, it was found that for coding prediction error frames, two classes of two-different block sizes, in which the largest block size is 16x16 pixels, was a good compromise between minimum information overhead and maximum compression efficiency. Thus, one single bit is used as overhead information per block of 16x16 pixels, which indicates whether a **B** block is coded as it is, or if it is further subdivided into four 8x8-pixel blocks. Using this simple scheme, the overhead information is kept to a mere (1 bit)/(16x16 pixels) $\approx$ 0.004 bit/pixel.

6.5.2 *Classification metric*

The standard deviation was used as the metric to classify the **B** blocks. Experimentally, it was determined that setting the classification threshold at a value of 3.5 leads to decoded frames of the same quality (subjectively and quantitatively) as those obtained by the non-adaptive scheme. By increasing the value of this threshold, the compression ratios obtained also increase, but so does the difference of PSNR between the reconstructed frames obtained with the adaptive and non-adaptive schemes. The results presented in the next section were obtained by setting a value of 3.5 for the standard deviation (or its equivalent variance value).

6.6 Results

The success of the adaptive algorithm in improving the compression ratio depends on the number of **0** blocks that are selected during the classification stage. The higher the number of selected **0** blocks, the

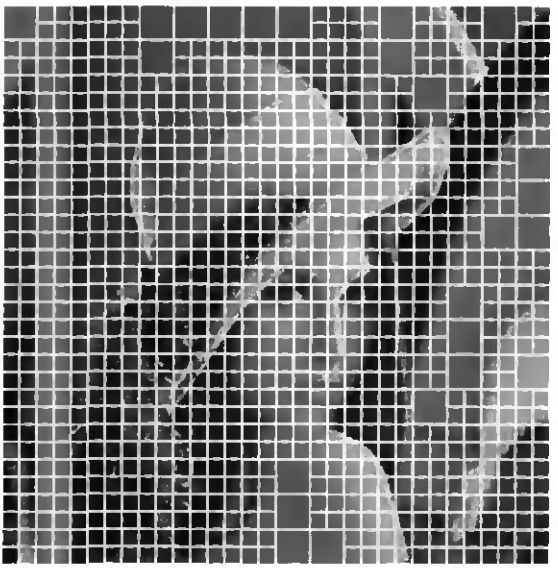


Figure 6.3: Block classification.

higher the improvement of the compression ratio with respect to the non-adaptive scheme.

6.6.1 Still image coding

Coding the image Lena with the proposed scheme gives only a modest improvement of the compression ratio. This is due to the very small number of **0** blocks that are selected during the block classification, as it is shown in Figure 6.3. This was nevertheless the expected result, since the classification parameters were chosen to match the statistics of prediction error frames.

6.6.2 Video coding

Unlike for still images, the number of **0** blocks selected during the block classification for prediction error frames is very high. For the sequences Miss America and Claire, the percentage of selected **0** blocks per frame

is shown in Figure 6.4. As can be observed from these curves, for most of the frames of the sequence Miss America, more than half of the **B** blocks are classified as **0**; while for all the frames (except the first one) of the sequence Claire more than 75% of the **B** blocks are classified as **0**, which leads to a dramatic improvement on the compression efficiency of the system.

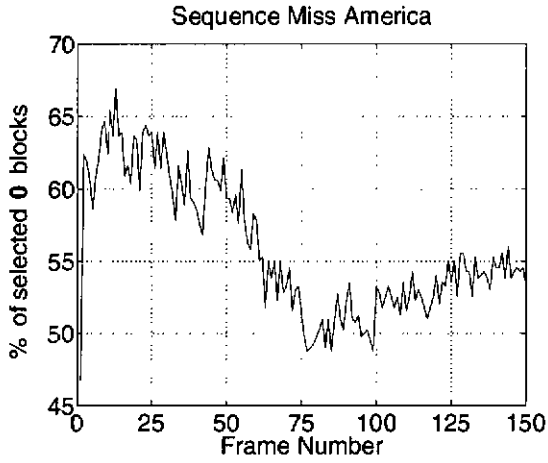
The coding performance of the adaptive scheme in video sequences has been studied by using the coding system shown in Figure 6.5. This is basically a coder based on the H.261 standard [30] without motion compensation.

The curves showing the compression ratio of the adaptive versus the non-adaptive method are shown in Figure 6.6(a) for the sequence Miss America. Displaying for evaluation the video sequence built from the reconstructed frames, shows the same sequence quality for both the adaptive and non-adaptive schemes. The PSNR curve of the adaptive scheme is, on average, 0.081 dB below the corresponding curve of the non-adaptive scheme: a difference that is normally not perceptible by the human visual system [127].

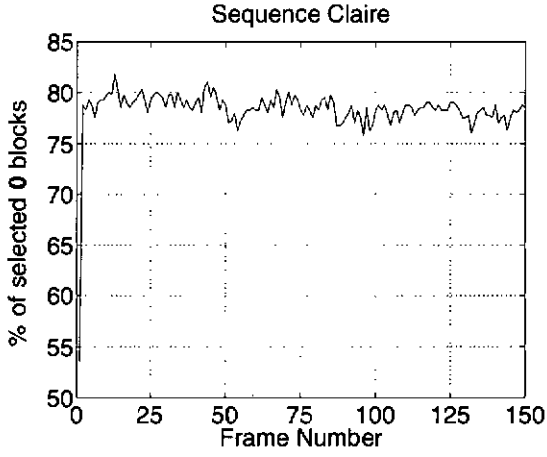
The results of the compression ratio for the sequence Claire are shown in Figure 6.6(b). The conclusions regarding the quality of the reconstructed frames are the same as those pointed out for the sequence Miss America.

Figures 6.7 and 6.8 show an original and its reconstructed frame from the sequences Miss America and Claire respectively. The selected frames are the ones with the highest percentage of selected **0** blocks for each respective sequence.

It is worth noting that the curves in Figure 6.6 do not include motion estimation/compensation. A low power VLSI implementation of the system was considered for the adaptive algorithm. Given that for the original system, more than 62% of the operations are required by the motion estimator (which potentially translates into a roughly proportional allocation of power), it was convenient to dispose of the motion estimation block. Such an approach has been used by other



(a)



(b)

Figure 6.4: Percentage of selected **0** blocks.

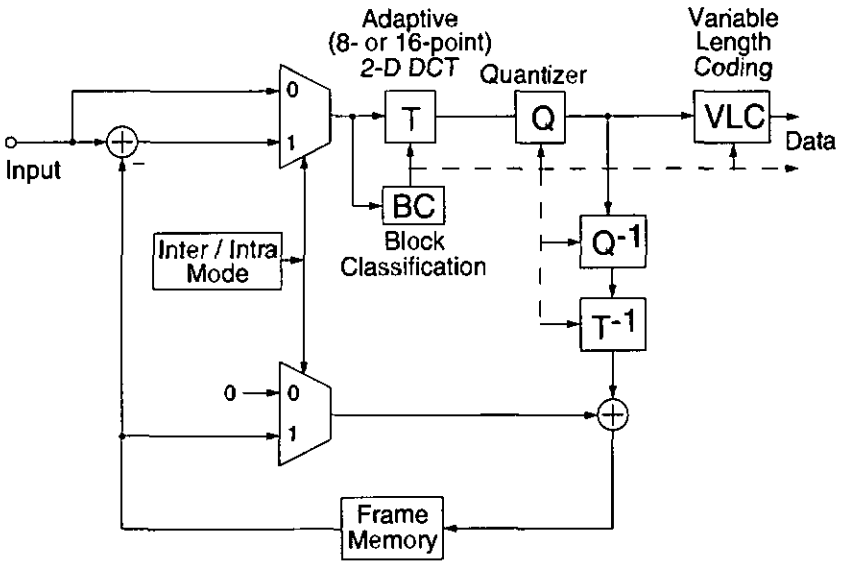


Figure 6.5: Adaptive H.261-based algorithm.

authors [55]. Beyond the low power concern, it is expected that by using a motion compensation technique, the compression ratio will be further improved, since a better prediction would increase the percentage of selected **0** blocks in the prediction error frames.

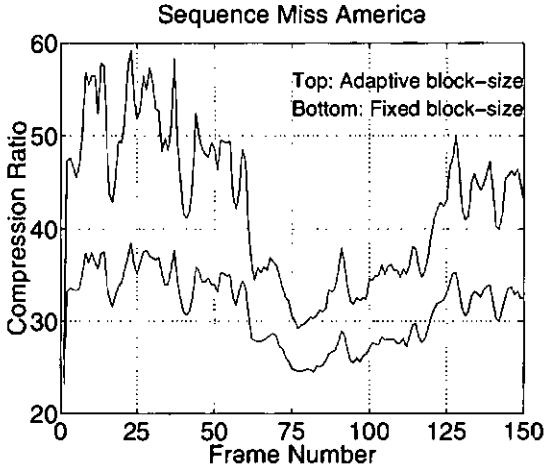
6.7 Computational Complexity

The results presented in the last section are very favorable from a compression efficiency point of view. It is expected that this improvement of the compression ratio is obtained ineluctably in exchange for a huge increase of the computational complexity. Indeed, the original coding system which contained an already complex 8-point DCT module, must now provide, in order to operate in an adaptive mode, additional computational and memory resources to calculate an even more complex 16-point DCT. Furthermore, a non-negligible number of operations are equally required by the block classification algorithm.

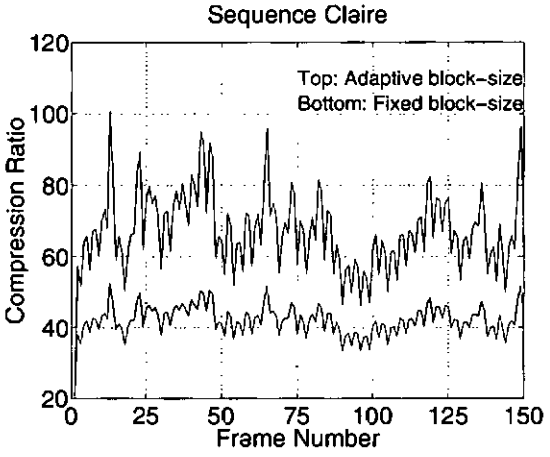
(a) Original

(b) Reconstructed

Figure 6.7: Miss America, frame No. 13.



(a)



(b)

Figure 6.6: Compression ratio.

(a) Original

(b) Reconstructed

Figure 6.8: Claire, frame No. 13.

In the following sections, a strategy to considerably reduce the computational burden of the adaptive system is reported. By following this strategy of implementation, the number of operations can be reduced up to a point that can make the adaptive method more efficient than the non-adaptive scheme, also in terms of computational complexity.

6.7.1 Reducing the complexity of the 16-point DCT

When a B block is classified as a 0 block, it can be taken for granted that this 16x16 block of pixels corresponds to a smooth region of minimum image activity. This assumption led us to make an analysis of the spectrum of the 16x16 transform coefficients of the 0 blocks, and it was noticed that in general, for this kind of blocks, only the first four low-frequency coefficients have a significant magnitude.

This simple fact has a radical effect of the implementation of the system. The heavy computational overhead of a 16-point 2-D DCT is substantially reduced since only these four coefficients are to be calculated for each 0 block. In quantitative terms, for a distributed arithmetic hardware implementation of the DCT (see Chapter 2), when a block is classified as a 0 block a *saving* of about 85% of the operations is achieved with respect to the non-adaptive scheme, which in absence of the classification stage is constrained to compute all the 256 coefficients associated with the four 8x8-pixel blocks.

Since the proportion of selected 0 blocks in a video sequence is typically high (e.g., an average of 56% and 78% for the first 150 prediction error frames of Miss America and Claire respectively) the global reduction of operations is considerable (for example reductions of the order of 38% for the Miss America and 55% for the Claire sequence, including in these figures the computational overhead due to the classification stage). The results presented in the previous section were obtained using this strategy for the reduction of the computational complexity. This important saving of operations is even higher at the decoder where the classification overhead is not present.

6.7.2 Reducing the complexity of the classification stage

The variance $\sigma^2 = \frac{1}{N \cdot N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [x(i,j) - \bar{x}]^2$ (or equivalently the standard deviation) has been widely used as a metric to measure the degree of activity of the pixels of an image. All tbe same, its computational complexity is far from being negligible. For the adaptive system, simulations demonstrated that without compromising the results, a further reduction of the computational complexity is obtained by using the Mean Absolute Deviation (which is also known as Average Deviation) as the classification metric defined as $\text{MADev} = \frac{1}{N \cdot N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |x(i,j) - \bar{x}|$.

It was found, that for a standard deviation, classification threshold value of 3.5 corresponds a MADev threshold value of 2.88. In a large number of test images/frames more that 90% (worst case) of the 0 blocks selected with tbe standard deviation were also selected with the use of the mean absolute deviation.

6.8 Summary

In this chapter it was reported an adaptive block-size transform coding scheme, and a mechanism that provides a significant reduction of its computational complexity. The method features a minimum information-overhead and is based on a transform coding technique that uses two different block sizes. The input image is divided into blocks of 16x16 pixels and each of these blocks is classified according to its image activity. Depending on this classification, eitber four 8-point or a single 16-point 2-D DCT is applied on the block. Tbe purpose of the algorithm is to take advantage of the high energy packing efficiency of a 16-point DCT, applied in a zone in which the stationarity of the image is guaranteed by a classification stage. For the same reconstruction quality, the results of the adaptive algorithm show a significant improvement of the compression ratio with respect to the non-adaptive scheme, particularly to code image sequences.

Chapter 7

Conclusions

The pervasiveness of digital image and video information in today's society has motivated the development of image and video compression techniques. Indeed, given the large number of bits required to represent digital image and video data, their management becomes impractical without a good compression strategy.

Though image compression algorithms have been proposed for more than 25 years, it was not until the relatively recent advances in VLSI technology that it has been possible to exploit data compression to the fullest. This has made VLSI design an indispensable partner of image and video coding, particularly for real-time applications.

The content of this dissertation dealt with image and video compression, both from a VLSI implementation and from an algorithmic point of view. The first part was dedicated in particular to the study of design methodologies and architectures for VLSI implementations. The second part was mainly algorithmic-oriented, keeping a close regard to the computational complexity issues.

7.1 Results

Each of the main contributions in this dissertation was reported in a different chapter, along with examples of application and a presentation of results. A summary of the presented ideas and their results is given in the following paragraphs.

Chapter 2 reported a design methodology for the VLSI implementation of image and video coding algorithms. The utilization of this methodology leads to the efficient transformation of an image coding algorithm into a VLSI circuit. The intermediate results of this methodology could also be used to develop solutions for other kinds of technologies (e.g., DSP, FPGA). A known algorithm for image compression was presented and utilized as an example on which the reported methodology was applied. The results were shown at each of the different phases of the methodology and the resulting ASIC and its characteristics for this particular implementation were also reported.

VLSI for video coding was directly addressed in Chapter 3, where an efficient VLSI architecture for a motion estimation algorithm was reported. A particular implementation scheme was introduced which leads to a significant reduction of the input bandwidth. This implementation scheme in combination with a bit-serial data processing produces excellent results in terms of silicon area. An ASIC developed for a particular application and based on the reported architecture was also presented.

Chapter 4 dealt with the power consumption issue in portable image communications systems. A power management scheme was reported in this chapter, in which an alternative to the traditional trade-off image distortion vs. coding bit rate was formulated. The results presented demonstrated the quality of the reconstructed images and the significant saving of operations, and thus of power, which is obtained by powering-off some modules of the circuit that implements the coding/decoding algorithm.

A bit-rate control method was proposed in Chapter 5. It is suitable for controlling the compression ratio of the JPEG algorithm. Many examples of the application of this method were reported, in which its excellent accuracy was evidenced. Besides the accuracy, this method presents several significant advantages with respect to previously reported methods, among others, it does not induce any degradation to the compressed images (other than that inherent to JPEG), it has a very low computational complexity, and it is fully compliant with the

standard algorithm.

In Chapter 6 an adaptive algorithm for image and video coding was presented. This scheme selects the block-size of a transform-based algorithm according to the degree of image activity of the image region that is being coded. The results presented demonstrated the significant improvement of the adaptive scheme with respect to the fixed block-size algorithm. Equally important, a strategy to obtain a dramatic reduction of the computational complexity of this algorithm was also proposed.

7.2 Perspectives

Perspective work could be envisaged on the base of the work presented in this dissertation. A non-exhaustive list of these possible extensions are itemized per chapter as follows,

- Ch.2: Inclusion of more functionality on the blocks that implement the high-level model routines in order to be able to generate automatically a hardware description language code. As well as the addition of the power consumption constraint at all the levels of the design methodology.
- Ch.3: Study of VLSI architectures for motion estimation algorithms with the low power consumption constraint.
- Ch.4: Extension of the power management schemes for other types of algorithms, and possible inclusion of PM techniques as part of the design methodology of Chapter 2.
- Ch.5: Build more accurate models of the CR-SF characteristic by segmenting the scale-factor range into a higher number of zones.
- Ch.6: Conception of a VLSI architecture for the adaptive scheme, in view of a low power implementation.

Appendix A

JPEG Tables

Range	DC Category	AC Category
0	0	N/A
-1,1	1	1
-3,-2,2,3	2	2
-7,...,-4,4,...,7	3	3
-15,...,-8,8,...,15	4	4
-31,...,-16,16,...,31	5	5
-63,...,-32,32,...,63	6	6
-127,...,-64,64,...,127	7	7
-255,...,-128,128,...,255	8	8
-511,...,-256,256,...,511	9	9
-1023,...,-512,512,...,1023	A	A
-2047,...,-1024,1024,...,2047	B	B
-4095,...,-2048,2048,...,4095	C	C

Table A.1: Baseline JPEG coefficient categories.

Category	Code Length *(cat & cmp)*	Code Word
0	2	00
1	4	010
2	5	011
3	6	100
4	7	101
5	8	110
6	10	1110
7	12	11110
8	14	111110
9	16	1111110
10	18	11111110
11	20	111111110

Table A.2: Huffman table for the (DPCM-ed) DC coefficients.

Run/Category	Length	Code Word
0/0	4	1010 (=**EOB**)
0/1	3	00
0/2	4	01
0/3	6	100
...	...	...
0/A	26	1111111110000011
1/1	5	1100
1/2	7	11011
1/3	10	1111001
...	...	...
1/A	26	1111111110001000
2/1	6	11100
2/2	10	11111001
2/3	13	1111110111
...	...	...
2/A	26	1111111110001110
3/1	7	111010
3/2	11	111110111
3/3	15	111111110101
...	...	...
3/A	26	1111111110010101
4/1	7	111011
4/2	12	1111111000
4/3	19	1111111110010110
...	...	...
4/A	26	1111111110011101
5/1	8	1111010
5/2	13	11111110111
5/3	19	1111111110011110
⋮	⋮	⋮
F/A	26	1111111111111110

Table A.3: Example of a Huffman table for the AC coefficients.

Appendix B

Piecewise-linear Approximation Algorithm

1. Enter the target compression ratio CR_T.

2. Compress the input image using a scale factor of 2, for the JPEG normalization array. Let us call CR1 the resulting compression ratio.

   - If the absolute value of the relative error between CR1 and CR_T is less than 0.05 then the process stops here. Otherwise it will continue in step 3.

3. If (CR_T > CR1) then go to step 4 otherwise go to step 11.

4. Insert the value of CR1 in the equation m : CR1 of region No. 4 in Table 5.2, in order to find the slope of the linear model for the region $2 < sf \leq 5$. Having the value m_4 and the point (2,CR1), the intercept b_4 is implicitly defined and so is the linear model $cr = m_4 \cdot sf + b_4$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = CR_T$. Let us call SF2 this computed scale factor.

   - If (SF2 > 5) then go to step 5 otherwise go to 10.

5. Use the linear model $cr = m_4 \cdot sf + b_4$, in order to find the compression ratio CR_V that corresponds to a scale factor $sf = 5$. This operation defines the point (5,CR_V).

Insert the value of CR1 in the equation m : CR1 of region No. 5 in Table 5.2 in order to find the slope m_5 of the linear model for the region $5 < sf \leq 10$. Having the value m_5 and the point $(5,\mathrm{CR}_V)$, the intercept b_5 is implicitly defined and so is the linear model $cr = m_5 \cdot sf + b_5$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = \mathrm{CR}_T$. Let us call SF2 this computed scale factor. The previous value of SF2 is overwritten, becoming SF2'.

- If (SF2 > 10) then go to step 6 otherwise go to step 9.

6. Use the linear model $cr = m_5 \cdot sf + b_5$, in order to find the compression ratio CR_{VI} that corresponds to a scale factor $sf = 10$. This operation defines the point $(10,\mathrm{CR}_{VI})$.

   Insert the value of CR1 in the equation m : CR1 of region No. 6 in Table 5.2 in order to find the slope m_6 of the linear model for the region $10 < sf \leq 15$. Having the value m_6 and the point $(10,\mathrm{CR}_{VI})$, the intercept b_6 is implicitly defined and so is the linear model $cr = m_6 \cdot sf + b_6$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = \mathrm{CR}_T$. Let us call SF2 this computed scale factor. The previous value of SF2 is overwritten, becoming SF2".

   - If (SF2 > 15) then go to step 7 otherwise go to step 8.

7. Signal that the quality of the compressed image, with the requested compression ratio, will be objectionable. If desired, the image is compressed using SF2 as scale factor.

   - Stop.

8. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio.

   - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise, make $m_t = m_6$ and $b_t = b_6$ and continue in step 18.

9. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio.

   - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise, make $m_t = m_5$ and $b_t = b_5$ and continue in step 18.

10. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio.

    - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise, make $m_t = m_4$ and $b_t = b_4$ and continue in step 18.

11. Insert the value of CR1 in the equation m : CR1 of region No. 3 in Table 5.2, in order to find the slope of the linear model for the region $1.5 \leq sf < 2$. Having the value m_3 and the point (2,CR1), the intercept b_3 is implicitly defined and so is the linear model $cr = m_3 \cdot sf + b_3$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = CR_T$. Let us call SF2 this computed scale factor.

    - If (SF2 < 1.5) then go to step 12 otherwise go to 15.

12. Use the linear model $cr = m_3 \cdot sf + b_3$, in order to find the compression ratio CR_{III} that corresponds to a scale factor $sf = 1.5$. This operation defines the point $(1.5,CR_{III})$.

    Insert the value of CR1 in the equation m : CR1 of region No. 2 in Table 5.2 in order to find the slope m_2 of the linear model for the region $1 \leq sf < 1.5$. Having the value m_2 and the point $(1.5,CR_{III})$, the intercept b_2 is implicitly defined and so is the linear model $cr = m_2 \cdot sf + b_2$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = CR_T$. Let us call SF2 this computed scale factor. The previous value of SF2 is overwritten, becoming SF2'.

    - If (SF2 < 1) then go to step 13 otherwise go to step 16.

13. Use the linear model $cr = m_2 \cdot sf + b_2$, in order to find the compression ratio CR_{II} that corresponds to a scale factor $sf = 1$. This operation defines the point $(1, CR_{II})$.

    Insert the value of CR1 in the equation m : CR1 of region No. 1 in Table 5.2 in order to find the slope m_1 of the linear model for the region $0.5 \leq sf < 1$. Having the value m_1 and the point $(1, CR_{II})$, the intercept b_1 is implicitly defined and so is the linear model $cr = m_1 \cdot sf + b_1$. Use this linear model to find the scale factor that corresponds to a compression ratio $cr = CR_T$. Let us call SF2 this computed scale factor. The previous value of SF2 is overwritten, becoming SF2".

    - If (SF2 < 0.5) then go to step 14 otherwise go to step 17.

14. Compress the input image using a scale factor of 0.5.

    - Stop.

15. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio.

    - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise, make $m_t = m_3$ and $b_t = b_3$ and continue in step 18.

16. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio.

    - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise, make $m_t = m_2$ and $b_t = b_2$ and continue in step 18.

17. Compress the input image using a normalization array scaled by the value of SF2. Let us call CR2 the resulting compression ratio.

    - If the absolute value of the relative error between CR2 and CR_T is less than 0.05 then the process stops here. Otherwise, make $m_t = m_1$ and $b_t = b_1$ and continue in step 18.

18. In order to appropriately approximate CR_T, an intercept adjustment is necessary. The slope m_t and the point (SF2,CR2) implicitly define a more accurate model $cr = m_t \cdot sf + b_{t'}$. In this updated model, find the scale factor that corresponds to a compression ratio $cr = \mathrm{CR}_T$. Let us call SF3 this computed scale factor.

19. Finally, using SF3 to scale the JPEG normalization array, compress the input image.

    - Stop.

Appendix C

Abbreviations and Acronyms

2D	Two dimensional
AC Coeff	Each element of the 2-D DCT coefficients array that is not the DC Coefficient
AD	Absolute Difference
ASIC	Application Specific Integrated Circuit
bpp	Bit per pixel
bps	Bit per second
CAD	Computer Aided Design
CB	Candidate Block
CBM	Candidate Block Memory
CD	Compact Disc
CIF	Common Intermediate Format
CMOS	Complementary Metal-Oxide Semiconductor
CR or cr	Compression Ratio
DAP	Distributed Arithmetic Processor
DC Coeff	The element (0,0) of the 2-D DCT coefficients array
DCT	Discrete Cosine Transform
DPCM	Differential Code Pulse Modulation
DSP	Digital Signal Processing/Processor
DVD	Digital Versatile Disk
EOB	End of Block
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface

HDTV	High Definition Television
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITU	International Telecommunications Union
JPEG	Joint Photographic Experts Group
MAD	Mean Absolute Difference
MHEG	Multimedia Hypermedia Experts Group
M-JPEG	Motion (or Moving) JPEG
MPEG	Moving Pictures Experts Group
PC	Personal Computer
PCM	Pulse Code Modulation
PM	Power Management
PSTN	Public Switched Telephone Network
PSNR	Peak Signal-to-Noise Ratio
RAM	Random Access Memory
RB	Refence Block
RBM	Reference Block Memory
ROM	Read Only Memory
SA	Search Area
SF or sf	Scale Factor
TC	Transform Coding
VLSI	Very Large Scale Integration
VSP	Video Signal Processor
VQ	Vector Quantization

Bibliography

[1] N. Negroponte, *"Being Digital"*, Alfred A. Knopf, New York, USA, 1995.

[2] R. C. Gonzalez and R. E. Woods, *"Digital Image Processing"*, Addison-Wesley, Reading, MA, USA, 1992.

[3] M. Rabbani and P.W. Jones, *"Digital Image Compression Techniques"*, Vol. TT 7, SPIE Optical Engineering Press, Bellingham, WA, USA, 1991.

[4] K. Sayood, *"Introduction to Data Compression"*, Morgan Kaufmann Publishers, San Francisco, CA, USA, 1996.

[5] M. Kunt, G. Granlund, and M. Kocher, *"Traitement Numérique des Images"*, Presses Polytechniques et Universitaires Romandes, Collection Electricité, Traitement de l'Information, Vol. 2, Lausanne, Switzerland, 1993. (In French).

[6] P.G. Howard, *"The design and analysis of efficient lossless data compression systems"*, Ph.D. Thesis, Brown University, May 1993.

[7] N.D. Memon and K. Sayood, "Lossless Compression of Video Sequences", *IEEE Trans. on Communications*, Vol. 44, No. 10, Oct. 1996, pp. 1340–1345.

[8] R. J. Clarke, *"Transform Coding of Images"*, Academic Press, London, Great Britain, 1985.

[9] K. R. Rao and P. Yip, *"Discrete Cosine Transform: Algorithms, Advantages, Applications"*, Academic Press, Boston, MA, USA, 1990.

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

[11] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantization design", *IEEE Trans. on Communications*, Vol. 28, Jan. 1980, pp. 84–95.

[12] J.W. Woods, Ed., *"Subband Image Coding"*, Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[13] T.A. Ramstad, S.O. Aåse, and J.H. Husoy, *"Subband Compression of Images: Principles and Examples"*, Series Advances in Image communications, Elsevier Science Publishers, Amsterdam, The Netherlands, 1995.

[14] M. Vetterli and J. Kovačević, *"Wavelets and Subband Coding"*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

[15] A.E. Jacquin, "Fractal Image Coding: A Review", *Proc. of the IEEE*, Vol. 81, No. 10, Oct. 1993, pp. 1451–1465.

[16] Y. Fisher, Ed., *"Fractal Image Compression: Theory and Applications"*, Springer-Verlag, New York, USA, 1995.

[17] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second generation image coding techniques", *Proc. of the IEEE*, Vol. 73, No. 4, April 1985, pp. 549–574.

[18] G. Karlsson and M. Vetterli, "Three dimensional subband coding of video", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. 2, New York, USA, April 1988, pp. 1100–1103.

[19] T.R. Natarajan and N. Ahmed, "On interframe transform coding", *IEEE Trans. on Communications*, Vol. 25, No. 11, Nov. 1977, pp. 1323–1329.

[20] H.H. Chen, Y.S. Chen, and W.H. Hsu, "Low-rate sequence image coding via vector quantization", *Signal Processing*, Vol. 26, No. 3, March 1992, pp. 265–283.

[21] H.G. Musmann, M. Hotter, and J. Ostermann, "Object-oriented analysis-synthesis of moving images", *Image Communication*, Vol. 1, No. 2, Oct. 1989, pp. 117–138.

[22] D.E. Pearson, "Developments in Model-Based Video Coding", *Proc. of the IEEE*, Vol. 83, No. 6, June 1995, pp. 892–906.

[23] ITU-T Recommendation T.81 *"Digital Compression and Coding of Continuous-tone Still Images"*, Sept. 1992.

[24] ISO/IEC JTC1 CD 11172, *"Coding of Moving Pictures and Associated Audio for Digital Storage Media up to 1.5 Mbits/s"*, International Organization for Standardization (ISO), 1992.

[25] ISO/IEC JTC1 CD 13818, *"Generic Coding of Moving Pictures and Associated Audio"*, International Organization for Standardization (ISO), 1994.

[26] ISO/IEC CD 13522, *"Coded Representation of Multimedia and Hypermedia Information Objects"*, International Organization for Standardization (ISO), 1995.

[27] R. Price, "MHEG: An introduction of the future international standard for hypermedia object interchange",*Proc. 1st ACM Int'l. Conf. on Multimedia*, ACM Press, Anaheim, CA, USA, Aug. 1993, pp. 121–128.

[28] L. Chiariglione, "MPEG and multimedia communications", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 7, No. 1, Feb. 1997, pp. 5–18.

[29] T. Sikora, "The MPEG-4 Video Standard Verification Model", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 7, No. 1, Feb. 1997, pp. 19–31.

[30] ITU-T Recommendation H.261 *"Video Codec for Audiovisual Services at $p \times 64$ kbits"*, 1993.

[31] ITU-T Recommendation H.263 *"Video Codec for Low Bit Rate Communications"*, 1996.

[32] C. Anderson, "The software industry: The birth of a new species", *The Economist*, Vol. 339, No. 7967, May 25th, 1996.

[33] R.J. Offen, Ed., *"VLSI Image Processing"*, Collins & Sons, London, Great Britain, 1985.

[34] Philips, *TriMedia VLIW-Based PCI Multimedia Processor System*, TriMedia White Paper, 1995.

[35] Chromatics Research Inc., *Mpact*, Technical Documentation, 1996.

[36] Texas Instruments, *TMS320C8x*, Technical Documentation, 1995.

[37] Oxford Micro Devices Inc., *A236 Parallel Video DSP*, Technical Documentation, 1997.

[38] *IEEE Trans. of Circuits and Systems for Video Technology*, Special Issue on VLSI for Video Signal Processing, Vol. 5, No. 5, Oct. 1995.

[39] P. Pirsch, Ed., *"VLSI Implementations for Image Communications"*, Series Advances in Image Communications, Elsevier Science Publishers, Amsterdam, The Netherlands, 1993.

[40] C. Piguet and A. Stauffer, *"Synthèse de Circuits ASIC. Des choix méthodologiques aux applications industrielles"*, Dunot, Paris, France, 1990. (In French).

[41] S. Goto, Ed., *"Design Methodologies, Advances in CAD for VLSI"*, Vol. 6, Elsevier Science Publishers, Amsterdam, The Netherlands, 1986.

[42] J. Bracamonte, M. Ansorge, and F. Pellandini, "Design methodology for VLSI implementation of image and video coding algorithms", *Fourth Bayona Workshop on Intelligent Meth. in Sig. Proc. and Comm.*, Bayona-Vigo, Spain, June 1996, pp. 217–220.

[43] J. Bracamonte, M. Ansorge, and F. Pellandini, "Design Methodology for VLSI Implementation of Image and Video Coding Algorithms – A Case Study", Chapter 14 of *Intelligent Methods in Signal Processing and Communications*, Birkauser, Boston, MA, USA, 1997.

[44] W.B. Pennebaker and J.L. Mitchell, *"JPEG Still Image Data Compression Standard"*, Van Nostrand Reinhold, New York, NY, USA, 1993.

[45] H. Lohscheller, "A subjectively adapted image communication system", *IEEE Trans. on Communications*, Vol. 32, Dec. 1984, pp. 1316–1322.

[46] A.B. Watson, "Perceptual optimization of DCT color quantization matrices", *Proc. IEEE Int'l. Conf. on Image Processing* (ICIP), Vol. I, Austin, TX, USA, Nov. 1994, pp. 100–104.

[47] W.-H. Chen and W.K. Pratt, "Scene Adaptive Coder", *IEEE Trans. on Communications*, Vol. 32, 1984, pp. 225–232.

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

[49] D.A. Huffman, "A method for the construction of minimum redundancy codes", *Proc. of the IRE*, Vol. 40, No. 10, 1952, pp. 1098–1101.

[50] G.K. Wallace, "The JPEG still picture compression standard", *IEEE Trans. on Consumer Electronics*, Vol. 38, No. 1, Feb. 1992, pp. xviii–xxxiv.

[51] D. Rasure, D. Arguiro, T. Sauer, and C. William, "A visual language and software development environment for image processing", *Int'l. J. of Imaging Systems and Technology*, Vol. 2, 1990, pp. 183–199.

[52] http://www.khoral.com

[53] Khoral Research, Inc., *Khoros Programmer's Manual*, 1995.

[54] A.P. Chandrakasan, *"Low power digital CMOS design"*, Ph.D. Thesis, Univ. of California, Berkeley, USA, Aug. 1994, Chapter 7.

[55] T.H. Meng, B.M. Gordon, E.K. Tsern, and A.C. Hung, "Portable Video-on-demand in wireless Communication", *Proc. of the IEEE*, Vol. 83, No. 4, July 1995, pp. 659–680.

[56] W.B. Rabiner and A.P. Chandrakasan, "Network-driven motion estimation for wireless video terminals", *IEEE Trans. of Circuits and Systems for Video Technology*, Vol. 7, No. 4, Aug. 1997, pp. 644–653.

[57] P. Denyer and D. Renshaw, *"VLSI Signal Processing: A Bit-serial Approach"*, Addison-Wesley, Microelectronics Systems Design Series, Wokingham, Great Britain, 1985.

[58] S.G. Smith and P.B. Denyer, *"Serial-Data Computation"*, Kluwer Academic Publisher, Boston, MA, USA, 1988.

[59] R.I. Hartley and K.K. Parhi, *"Digit-Serial Computation"*, Kluwer Academic Publishers, Boston, MA, USA, 1995.

[60] A. Peled and B. Liu, "A new hardware realization of digital filters", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-22, No. 6, Dec. 1974, pp. 456–462.

[61] U. Sjöström, *"On the design and implementation of DSP algorithms: An approach using wave digital state-space filters and distributed arithmetic"*, Ph.D. Thesis, University of Neuchâtel, Switzerland, 1993.

[62] S-M. Lei and M-T. Sun, "An entropy coding system for digital HDTV applications", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 1, No. 1, March 1991, pp. 147–155.

[63] COMPASS Design Automation, Inc.: Manuals, COMPASS, San Jose, CA, USA, 1993.

[64] Mentor Graphics, http://www.menthor.com

[65] Cadence, http://www.cadence.com

[66] N. Weste and K. Eshraghian. *"Principles of CMOS VLSI design: A Systems Perspective"*, Addison-Wesley, VLSI System Series, 2nd Edition, Reading, MA, USA, 1993.

[67] C. Henny, *"A VLSI Implementation of a Huffman Coder"*, Diploma Project, University of Neuchâtel, Switzerland, Aug. 1995.

[68] J. Bracamonte, I. Defilippis, M. Ansorge, and F. Pellandini. "Bit-serial parallel processing VLSI architecture for a block matching motion estimation algorithm", *Proc. of the Int'l. Picture Coding Symposium PCS'94*, Sacramento, CA, USA, Sept. 1994, pp. 22–25.

[69] C.D. Kuglin and D.C. Hines, "The phase correlation image alignment method", *Proc. IEEE Int'l. Conf. on Cybernetics and Society*, New York, NY, USA, Sept. 1975, pp. 163–165.

[70] M.H. Ahmad Fadzil and T.J. Dennis, "A hierarchical motion estimation for interframe coding", *IEE Colloquium on Applications of Motion Compensation*, Digest No. 1990/128, Oct. 1990, pp. 3.1–3.6.

[71] A.N. Netravali and J.D. Robbins, "Motion compensated television coding: part 1", *Bell Syst. Tech. Journal*, Vol. 58, No. 3, April 1979, pp. 631–670.

[72] D.R. Walker and K.R. Rao, "Improved pel-recursive motion compensation", *IEEE Trans. of Communications*, Vol. 32, No. 10, Oct. 1984, pp. 1128–1134.

[73] J. Biemond, L. Looijenga, D.E. Boekee, and R.H Plompen, "A pel-recursive Wiener-based displacement estimation algorithm", *Signal Processing*, Vol. 13, No. 4, Dec. 1987, pp. 399–412.

[74] J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Trans. on Communications*, Vol. 29, Dec. 1981, pp. 1799–1808.

[75] V. Seferidis and M. Ghanbari, "Generalized block-matching motion estimation", *Proc. of the SPIE: Visual Communications and Image Processing*, Vol. 1818, Nov. 1992, pp. 110–119.

[76] H.K. Chow and M.L. Liou, "Genetic motion search algorithm for video compression", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 3, No. 6, Dec. 1993, pp. 440–445.

[77] M.T. Orchard, "Comparison of techniques for estimating block motion in image sequence coding", *Proc. of the SPIE: Visual Communication and Image Processing IV*, Vol. 1199, Nov. 1989, pp. 248–258.

[78] F. Dufaux and M. Kunt, "Multigrid based motion estimation for interframe image sequence coding", *Proc. of the EUSIPCO*, Brussels, Belgium, Aug. 1992, pp. 1323–1326.

[79] M. Ghanbari, "The cross-search algorithm for motion estimation", *IEEE Trans. on Communications*, Vol. 38, No. 7, July 1990, pp. 950–953.

[80] A. Puri, H.M. Hang, and D.L. Schilling, "An efficient block-matching algorithm for motion compensated coding", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. 3, Dallas, TX, USA, April 1987, pp. 1063–1066.

[81] G. Madec, "Half pixel accuracy in block matching", *Proc. of the Picture Coding Symposium*, Cambridge, MA, USA, March 1990, pp. 9.12-1–9.12-3.

[82] S.L. Iu, "Comparison of motion compensation using different degrees of sub-pixel accuracy for interfield/interframe hybrid coding of HDTV image sequences", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. 3, San Francisco, CA, USA, March 1992, pp. 465–468.

[83] H.G. Musmann, P. Pirsch, and H. Grallert, "Advances in Picture Coding", *Proc. of the IEEE*, Vol. 73, No. 4, April 1985, pp. 523–548.

[84] U. Sjöström, I. Defilippis, M. Ansorge and F. Pellandini, "A Discrete Cosine Transform Chip for Real Time Video Applica-

tions", *Proc. IEEE Int'l. Symposium Circuits and Systems* (ISCAS), Vol. 2, New Orleans, LA, USA, May 1990, pp. 1620–1623.

[85] T. Komarek and P. Pirsch, "VLSI architectures for block matching algorithms", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Glasgow, Scotland, May 1989, pp. 2457–2460.

[86] L. De Vos, M. Stegherr, and T.G. Noll, "VLSI architectures for the full-search blockmatching algorithm", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Glasgow, Scotland, May 1989, pp. 1687–1690.

[87] P. Pirscb, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression – A survey", *Proc. of the IEEE*, Vol. 83, No. 2, Feb. 1995, pp. 220–246.

[88] J. Yuan and C. Svensson, "High-Speed CMOS Circuit Technique", *IEEE J. of Solid-State Circuits*, Vol. 24, No. 1, Feb. 1989, pp. 62–70.

[89] A. Dahle, "Designing high performance systems to run from 3.3V or lower sources", *Proc. Silicon Valley Personal Computer Design Conf.*, 1991, pp. 685–691.

[90] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low power CMOS design", *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 4, April 1992, pp. 472–484.

[91] J. Bracamonte, M. Ansorge, and F. Pellandini, "VLSI systems for image compression. A power-consumption/image-resolution tradeoff approach", *Proc. Conf. on Digital Compression Technologies & Systems for Video Communications*, SPIE Vol. 2952, Berlin, Germany, Oct. 1996, pp. 591–596.

[92] T. Berger, *"Rate Distortion Theory – A Mathematical Basis for Data Compression"*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.

[93] E. Harris, S. Depp, W. Pence, S. Kirkpatrick, M. Sri-Jayantha, and R. Troutman, "Technology directions for portable computers", *Proc. of the IEEE*, Vol. 83, No. 4, April 1995, pp. 636–658.

[94] S. Gary, C. Dietz, J. Eno, G. Gerosa, S-H. Park, and H. Sanchez, "The PowerPC 603 microprocessor: A low-power design for portable applications", *Tech. Digest of COMPCON 94*, San Francisco, CA, USA, Feb. 1994, pp. 307–315.

[95] S.C. Ellis, "Power management in notebook computers", *Proc. of the Silicon Valley Personal Computer Conf.*, Santa Clara, CA, USA, July 1991, pp. 749–754.

[96] A. Bellaouar and M.I. Elmasry, *"Low-Power Digital VLSI Design. Circuits and Systems"*, Kluwer Academic Press, Boston, MA, USA, 1995.

[97] Herman A.I. Claus, "Digital network for video surveillance and video distribution", *Proc. Conf. on Digital Compression Technologies and Systems for Video Communications*, SPIE Vol. 2952, Berlin, Germany, Oct. 1996, pp. 194–204.

[98] J. Bracamonte, M. Ansorge, and F. Pellandini, *"Procédé de contrôle du taux de compression d'images numériques"*, PCT/CH98/00413, Patent pending, filed in October 1997.

[99] ISO/JTC1/SC2/WG8 N800, *"Initial draft for adaptive discrete cosine transform technique for still picture data compression standard"*, Section 2.1.6 and Annex E: page 49, May 1988.

[100] Goo-man Park, "Image compression method for bit-fixation and the apparatus therefor", U.S. Patent 5,416,604. May 16, 1995.

[101] J. H. Lee, Y.C. Park, and D.H. Yoon, "Variable length-adaptive image data compression method and apparatus", U.S. Patent 5,475,502. Dec. 12, 1995.

[102] A. Razavi, R. Adar, I. Shenberg, R. Retter, and R. Friedlander, "VLSI implementation of an image compression algorithm with a

new bit rate control capability", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. 5, San Francisco, CA, USA, March 1992, pp. 669–672.

[103] A. Habibi and P.A. Wintz, "Image coding by linear transformation and block quantization", *IEEE Trans. on Communications*, Vol. 19, 1971, pp. 50–63.

[104] J. Zdepski, D. Raychaudhuri, and K. Joseph, "Statistically based buffer control policies for constant rate transmission of compressed digital video", *IEEE Trans. on Communications*, Vol. 39, 1991, pp. 947–957.

[105] C.T. Cheng and A. Wong, "A self-governing rate buffer control strategy for pseudoconstant bit rate video coding", *IEEE Trans. on Image Processing*, Vol. 2, 1993, pp. 50–59.

[106] P. F. Farrelle, *"Recursive Block Coding for Image Data Compression"*, Springer-Verlag, New York, USA, 1990.

[107] J. Vaisey and A. Gersho, "Image compression with variable block size segmentation", *IEEE Trans. on Signal Processing*, Vol. 40, No. 8, Aug. 1992, pp. 2040–2060.

[108] J. Bracamonte, M. Ansorge, and F. Pellandini, "Adaptive Block-Size Transform Coding for Image Compression", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. 4, Munich, Germany, April 1997, pp. 2721–2724.

[109] A. Habibi, "Survey of adaptive image coding techniques", *IEEE Trans. on Communications*, Vol. 25, No. 11, Nov. 1977, pp. 1275–1284.

[110] A. Habibi, "An adaptive strategy for hybrid image coding", *IEEE Trans. on Communications*, Vol. 29, No. 12, Dec. 1981, pp. 1736–1740.

[111] W-H. Chen and H. Smith, "Adaptive coding of monochrome and color images", *IEEE Trans. on Communications*, Vol. 25, No. 11, Nov. 1977, pp. 1285–1292.

[112] R.A. Gonsalves, A. Shea, N. Evans, T. Huang, and, E. Delp, "Fixed-error encoding for bandwidth compression", *Proc. Seminar on Applications of Digital Image Processing*, SPIE Vol. 149, San Diego, CA, USA, Aug. 1978, pp. 27–42.

[113] G. Tu, G. Verbeek, J. Rommelaere, and A. Oosterlinck, "Adaptive coding in transform domain using vector quantization and scalar correction", *Proc. Conf. IEEE Region 10*, Seoul, Korea, Aug. 1987, pp. 418–422.

[114] T. Omachi, Y. Takashima, and H. Okada, "DCT-VQ coding scheme using categorization with adaptive band partition", *Proc. Picture Coding Symposium PCS'87*, Stockholm, Sweden, June 1987, pp. 161–162.

[115] K. Aizawa, H. Harashima, and H. Miyakawa, "Adaptive discrete cosine transform coding with vector quantization for color images", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Vol. 2, Tokio, Japan, April 1986, pp. 985–988.

[116] A.A. Adelwahab and S.C. Kwatra, "Image data compression with vector quantization in the transform domain", *Proc. IEEE Int'l. Conf. on Communications* (ICC), Toronto, Canada, June 1986, pp. 1286–1289.

[117] Y.S. Ho and A. Gersho, "Classified transform coding of images using vector quantization", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Glasgow, Scotland, May 1989, pp. 1890–1893.

[118] B. Ramamurthi and A. Gersho, "Classified vector quantization of images", *IEEE Trans. on Communications*, Vol. 34, Nov. 1986, pp. 1105–1115.

[119] J. Vaisey and A. Gersho, "Variable block-size image coding", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Dallas, Texas, USA, April 1987, pp. 1051–1054.

[120] C-T Chen, "Adaptive transform coding via quadtree-based variable blocksize DCT", *Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing* (ICASSP), Glasgow, Scotland, May 1989, pp. 1854–1857.

[121] P.A. Wintz, "Transform Picture Coding", *Proc. of the IEEE*, Vol. 60, No. 7, July 1972, pp. 809–820.

[122] H. Kitajima, "Energy packing efficiency of the Hadamard transform", *IEEE Trans. on Communications*, Vol. 24, 1976, pp. 1256–1258.

[123] P. Yip and K.R. Rao, "Energy packing efficiency for the generalized discrete transform", *IEEE Trans. on Communications*, Vol. 26, Aug. 1978, pp. 1257–1262.

[124] M. Vetterli and H. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations", *Signal Processing*, Vol. 6, 1984, pp. 267–278.

[125] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images", *Trans. of the IEICE*, Vol. E-71, No. 11, Nov. 1988, pp. 1095–1097.

[126] E. Feig and S. Winograd, "Fast Algorithms for the discrete cosine transform", *IEEE Trans. on Signal Processing*, Vol. 40, No. 9, Sept. 1992, pp. 2174–2193.

[127] V. Bhaskaran and K. Konstantinides, *"Image and Video Compression Standards. Algorithms and Architectures"*, Kluwer Academic Publishers, Boston, MA, USA, 1995.