Universidade de Aveiro
2007
Departamento de Electrónica, Telecomunicações e Informática

António José Ribeiro Neves

Compressão sem perdas de imagens com características particulares

Lossless compression of images with specific characteristics

Universidade de Aveiro
2007
Departamento de Electrónica, Telecomunicações e Informática

António José Ribeiro Neves

Compressão sem perdas de imagens com características particulares

Lossless compression of images with specific characteristics

tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Electrotécnica, realizada sob a orientação científica do Doutor Armando José Formoso de Pinho, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Trabalho financiado pela Fundação para a Ciência e a Tecnologia e pelo Fundo Social Europeu no âmbito do III Quadro Comunitário de Apoio.

À minha esposa Patrícia.

o júri

presidente

Doutor Joaquim Manuel Vieira
professor catedrático da Universidade de Aveiro

Doutor Paulo Jorge dos Santos Gonçalves Ferreira
professor catedrático da Universidade de Aveiro

Doutor Fernando Manuel Bernardo Pereira
professor associado com agregação do Instituto Superior Técnico da Universidade Técnica de Lisboa

Doutor Mário Alexandre Teles de Figueiredo
professor associado com agregação do Instituto Superior Técnico da Universidade Técnica de Lisboa

Doutor Armando José Formoso de Pinho
professor associado da Universidade de Aveiro

Doutor Manuel José Cabral dos Santos Reis
professor auxiliar com agregação da Universidade de Trás-os-Montes e Alto Douro

agradecimentos

Ao meu orientador, Doutor Armando Pinho, todo o apoio dado ao longo destes anos. Sem a sua preciosa ajuda, os seus conselhos e a sua paciência não teria sido possível terminar este trabalho.
Ao Doutor Paulo Jorge Ferreira, um agradecimento especial pelos conselhos dados ao longo deste trabalho e pelas sugestões feitas para a escrita desta Tese.
A todos os meus colegas do IEETA, pelo apoio que sempre me deram e por me ajudarem a acreditar que eu seria capaz de terminar este trabalho.
Ao IEETA e ao DETI, por me terem sempre garantido as condições necessárias para a realização deste trabalho. Agradeço também todo o apoio financeiro prestado nas mais diversas ocasiões.
À minha mãe e à minha irmã, que sempre acreditaram em mim, agradeço todo o apoio. À minha cunhada agradeço o incentivo, principalmente nos momentos de mais desalento.

palavras-chave

Compressão de imagem sem perdas, imagens simples, imagens de cor indexada, imagens de microarrays, compactação de histogramas, reordenação da palete de cores.

resumo

A compressão de certos tipos de imagens é um desafio para algumas normas de compressão de imagem. Esta tese investiga a compressão sem perdas de imagens com características especiais, em particular imagens simples, imagens de cor indexada e imagens de microarrays. Estamos interessados no desenvolvimento de métodos de compressão completos e no estudo de técnicas de pré-processamento que possam ser utilizadas em conjunto com as normas de compressão de imagem. A esparsidade do histograma, uma propriedade das imagens simples, é um dos assuntos abordados nesta tese. Desenvolvemos uma técnica de pré-processamento, denominada compactação de histogramas, que explora esta propriedade e que pode ser usada em conjunto com as normas de compressão de imagem para um melhoramento significativo da eficiência de compressão. A compactação de histogramas e os algoritmos de reordenação podem ser usados como pré-processamento para melhorar a compressão sem perdas de imagens de cor indexada. Esta tese apresenta vários algoritmos e um estudo abrangente dos métodos já existentes. Métodos específicos, como é o caso da decomposição em árvores binárias, são também estudados e propostos. O uso de microarrays em biologia encontra-se em franca expansão. Devido ao elevado volume de dados gerados por experiência, são necessárias técnicas de compressão sem perdas. Nesta tese, exploramos a utilização de normas de compressão sem perdas e apresentamos novos algoritmos para codificar eficientemente este tipo de imagens, baseados em modelos de contexto finito e codificação aritmética.

keywords

Lossless image compression, simple images, color-indexed images, microarray images, histogram packing, palette reordering.

abstract

The compression of some types of images is a challenge for some standard compression techniques. This thesis investigates the lossless compression of images with specific characteristics, namely simple images, color-indexed images and microarray images. We are interested in the development of complete compression methods and in the study of preprocessing algorithms that could be used together with standard compression methods. The histogram sparseness, a property of simple images, is addressed in this thesis. We developed a preprocessing technique, denoted histogram packing, that explores this property and can be used with standard compression methods for improving significantly their efficiency. Histogram packing and palette reordering algorithms can be used as a preprocessing step for improving the lossless compression of color-indexed images. This thesis presents several algorithms and a comprehensive study of the already existing methods. Specific compression methods, such as binary tree decomposition, are also addressed. The use of microarray expression data in state-of-the-art biology has been well established and due to the significant volume of data generated per experiment, efficient lossless compression methods are needed. In this thesis, we explore the use of standard image coding techniques and we present new algorithms to efficiently compress this type of images, based on finite-context modeling and arithmetic coding.

Contents

Chapter 1

Introduction

Image compression is the process of converting an input image (the source image) into another image (the output or the compressed image) that has smaller size. Image compression has been one of the most active topics of research in the signal processing area. Both the quantity and the quality of the literature in the field provides an ample proof of this.

In general, data can be compressed if are redundant. In data compression, we try to remove or reduce the redundancy in the data. Image compression can be lossy, i.e., in many cases, image compression may be associated with some loss of irrelevant information. However, in some other cases, losses are unacceptable. Therefore, the primary goal of lossless image compression is to minimize the number of bits required to represent the original image without any loss of information. This is the case of medical applications, color-indexed images, images with highly structured nature, such as text and graphics, and in applications where the image is to be extensively edited and recompressed.

There are many known methods for image compression. They are based on different ideas, are suitable for different types of images and produce different results. Nevertheless, they are based in the same principle: they compress an image by removing redundancy from the original image. This is possible because, for example, adjacent pixels in natural images (i.e., images capturing natural scenes) tend to have similar colors.

The most recent image compression standards that allow lossless compression, namely JPEG-LS and JPEG2000, have been developed with the assumption that images are smooth. In fact, this assumption is verified by most of the images representing natural content. Since this class of images has been traditionally used for testing the performance of the compression

techniques, it is not surprising that these techniques closely match the main characteristics of these images. However, the supremacy of "natural images" has progressively been diminishing, given place to images whose contents also include text, graphical and computer generated materials, besides the usual natural content.

The main purpose of this work is to improve the lossless compression of images with specific characteristics. We are interested in the development of complete compression methods and in the study of preprocessing algorithms that could be used as a means for improving the performance of the existing image compression standards.

In this thesis we investigate the lossless compression of three type of images:

- Simple images;
- Color-indexed images;
- Microarray images.

The concept of **simple image** was first introduced by Yoo *et al.* in [81]. This type of images is characterized by using only a small number of intensities. This property is present in images whose contents include text, graphical and computer generated materials, besides the usual natural content.

Color-indexed images are represented by a matrix of indexes (the index image) and by a color-map or palette [48, 39, 65, 51]. The indexes in the matrix point to positions in the color-map and, therefore, establish the colors of the corresponding pixels. This type of images are obtained by quantizing a full color image to an image with, generally, no more than 256 colors carefully selected. This process is usually considered in two parts: the selection of an optimal color palette and the optimal mapping of each pixel of the image to a color from the palette. Due to the high cost of full color displays in some applications, the necessity of high compressed color images as the case of the World Wide Web and the limited ability of humans to differentiate between the full range of representable colors, color images are often represented as color-indexed images.

The **microarray images** are the result of microarray experiments. The raw data of a microarray experiment consist of a pair of 16 bits per pixel grayscale images. These images are analyzed using a variety of software tools which extract relevant information that is used to evaluate the expression level of individual genes. The DNA microarray technology has become an important tool in the study of gene function, regulation, and interaction across

large numbers of genes, and even entire genomes. It allows the analysis of thousands of genes in a single experience [40, 19].

1.1 Motivation

Generally, the data that represents an image are redundant. However, images might be redundant in different ways. This is why different methods may not perform well for all images and why different methods are needed to compress different image types.

The diversity of contents presented by nowadays images poses some important challenges to the general purpose techniques which have been designed with the aim of compressing natural images. Normally, this problem originates a degradation in the compression rate, affecting both lossy and lossless techniques.

Figure 1.1 presents an example of a natural image and its histogram of intensities. This image is the luminance information of a natural color image. As we can see, the content of this image is smooth and its histogram has almost every possible intensity value. The smoothness of an image can be measured in different ways, for example, considering the entropy of the differences among neighboring pixels.

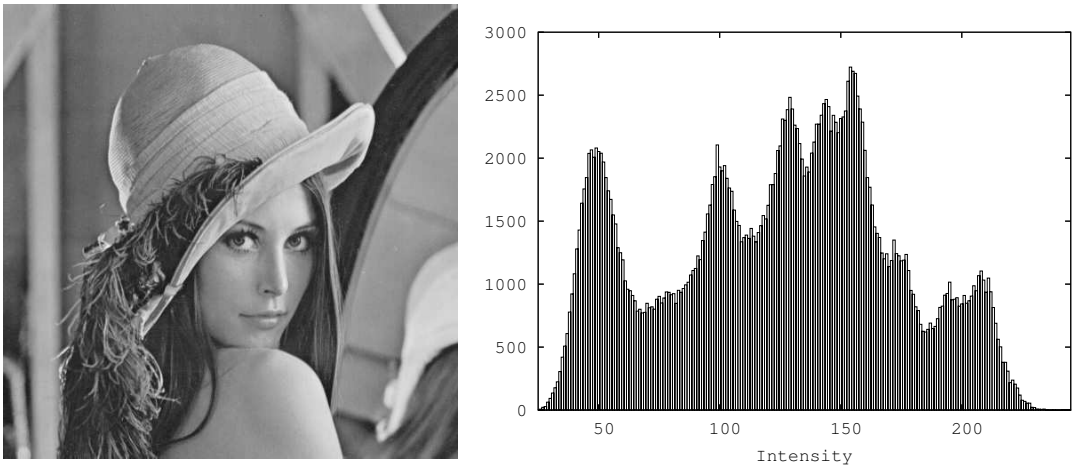


Figure 1.1: On the left, an example of a natural image. On the right, the histogram of the same image.

Figure 1.2 presents the same image, but after applying a color-quantization procedure. This

new image has only 256 colors. The displayed image shows the matrix of indexes (the index image) of the corresponding color-indexed image. These indexes point to positions in the color palette and, therefore, establish the colors of the corresponding pixels. The smoothness property of the original image is lost and, therefore, it is expected a degradation in the compression rate when using a general purpose compression method.

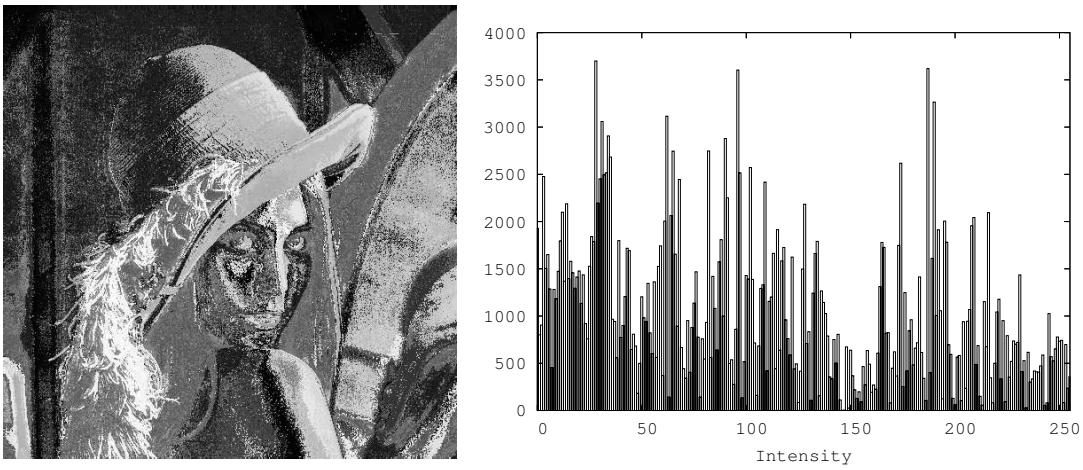


Figure 1.2: On the left, an example of the image of indexes of a color-indexed image. On the right, the histogram of the same image.

Figure 1.3 shows an example of a simple image and its correspondent histogram of intensities. As we can see, this image has only a small number of intensities, but their values are spread in all the scale. This property affects the eventual smoothness of the image.

Figure 1.4 presents an example of the 8 most significant bitplanes of a microarray image and its histogram. As can be seen, these images are highly structured. Moreover, almost all pixels have low intensities implying a highly asymmetrical histogram. These characteristics are not efficiently used by general purpose compression methods and may, in fact, generate a degradation in compression performance.

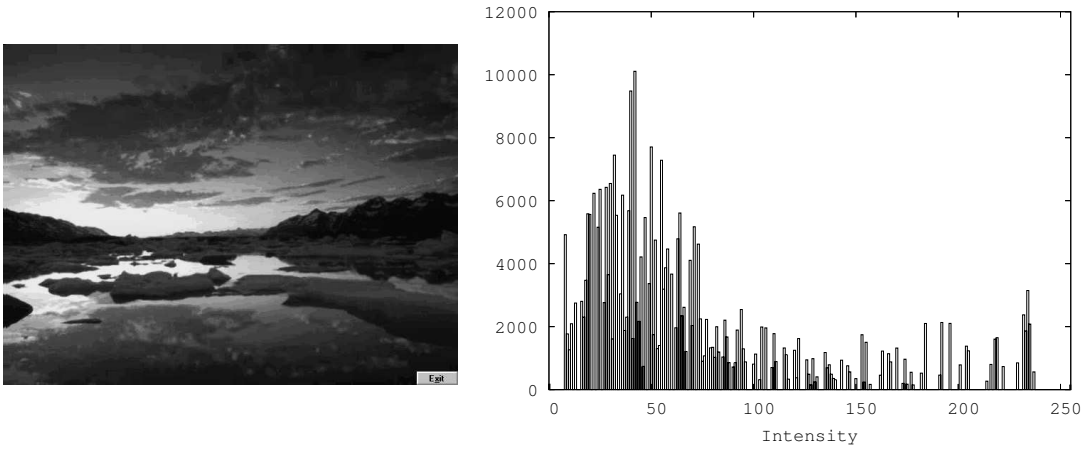


Figure 1.3: On the left, an example of a simple image. On the right, the histogram of the same image.

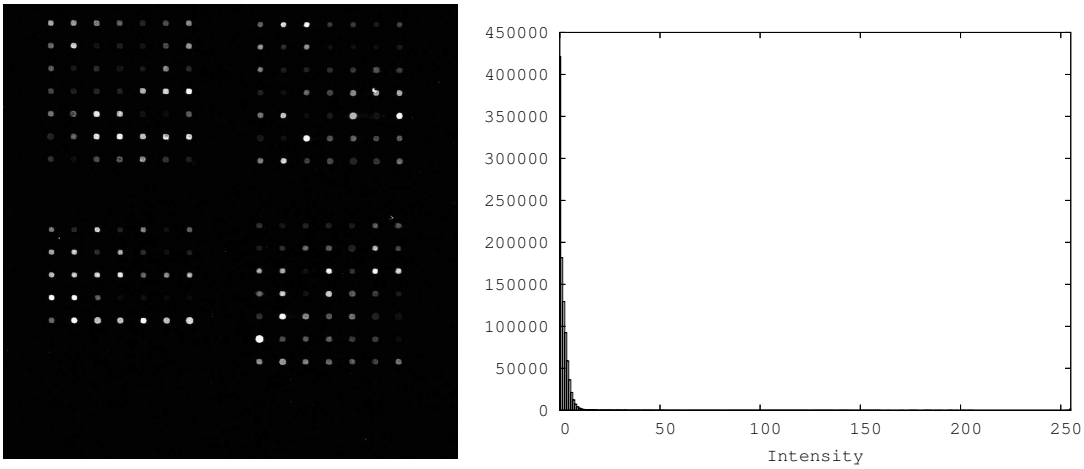


Figure 1.4: On the left, an example of the 8 most significant bitplanes of a microarray image. On the right, the histogram of the same image.

1.2 Main objectives

The main objectives of this thesis is to study the problem of efficiently compress images with specific characteristics, such as those presented in the previous section.

The most recent image compression standards that allow lossless compression have been developed with the aim of efficiently compress images representing natural content. This leads to poor performance when used with images that fail to comply to this characteristic. With this idea in mind, the major goals of this thesis are:

- Review and analyze the performance of standard image compression methods when used to compress the classes of images described in Section 1.1.
- Study of the palette reordering algorithms developed for the lossless compression of color-indexed images.
- Review and analyze the specific image compression algorithms developed for the lossless compression of the classes of images described in Section 1.1.
- Development of efficient preprocessing techniques that can be used before the standard image coding techniques to improve its compression efficiency.
- Development of specific methods for the compression of images with specific characteristics (those described in Section 1.1).

1.3 Thesis structure

This thesis is divided into seven chapters and one appendix:

- Chapter 2 provides a review of the most important lossless image compression standards.
- Chapter 3 presents the impact of histogram sparseness, a property of simple images, in the performance of the standard lossless methods. We propose a preprocessing technique that, when used with the standards, improves their compression performance. We also show specific methods, designed exclusively for the compression of this type of images. We present results showing the efficiency of the preprocessing techniques and specific methods.

- Chapter 4 deals with the problem of compressing color-indexed images. We review the most important specific methods for compression of this type of images, and present our contributions to this topic. We also present a new preprocessing approach that can be used for the same objective, block-based histogram packing, exploring the locally histogram sparseness, an important characteristic of this type of images. We present results showing the efficiency of the preprocessing techniques and specific methods.

- In Chapter 5, we review the most important methods for palette reordering, a preprocessing technique used to improve the compression of color-indexed images, and we present our contributions to this topic. We present results showing the efficiency of these preprocessing techniques.

- Chapter 6 deals with the problem of compressing another important type of images, the DNA microarray images. The use of microarray expression data in state-of-the-art biology has been well established and the widespread adoption of this technology, coupled with the significant volume of data generated per experiment, have led to significant challenges in storage of the data from microarray experiments. We present a comprehensive study regarding the use of image compression standards in the compression of microarray images. We also study the specific methods already developed and we present our approach. We present compression results regarding the use of standards and the specific methods.

- Chapter 7 summarizes the main contributions of this work and presents some possible future work.

- Appendix A provides the images used to evaluate the performance of the compression methods presented in this thesis.

1.4 Original contributions

The main contributions of this thesis can be summarized as follows:

- The development of preprocessing techniques, namely histogram packing techniques, than can be used with standards for lossless compression of synthetic images, presented in Chapter 3:

1. **António J. R. Neves**, Armando J. Pinho, *Improving the JPEG-LS compression of images with locally sparse histograms.* Proc. of the 12th Portuguese Conf. on Pattern Recognition, Recpad-2002, June 2002, Aveiro, Portugal.
2. Armando J. Pinho, **António J. R. Neves**, *Improvement of the lossless compression of images with quasi-sparse histograms*, Proc. of the 11th European Signal Processing Conference, EUSIPCO-2002, September 2002, Toulouse, France, Vol. II, pp. 467-470.

- A comprehensive study of the palette reordering algorithms already developed for the lossless compression of color-indexed images, presented in Chapter 5:

1. Armando J. Pinho, **António J. R. Neves**, *On the Relation Between Memon's and the Modified Zeng's Palette Reordering Methods*, Elsevier Image and Vision Computing, no. 24, 2006, pp. 534-540.
2. Armando J. Pinho, **António J. R. Neves**, *A survey on palette reordering methods for improving the compression of color-indexed images*, IEEE Trans. on Image Processing, vol. 13, no. 11, November 2004, pp. 1411-1418.
3. Armando J. Pinho, **António J. R. Neves**, *On the efficiency of luminance-based palette reordering of color-quantized images*, Proc. of the 1st Iberian Conference on Pattern Recognition and Image Analysis, ibPRIA-2003, June 2003, Puerto de Andratx, Spain, pp. 766-772.

- The development of preprocessing techniques, namely histogram packing and palette reordering techniques, than can be used with standards for lossless compression of color-indexed images presented in Chapter 4 and in Chapter 5:

1. Armando J. Pinho, **António J. R. Neves**, *A note on Zeng's technique for color re-indexing of palette-based images*, IEEE Signal Processing Letters, vol. 11, no. 2, February 2004, pp. 232-234.
2. **António J. R. Neves**, Armando J. Pinho, *A Bit-Plane Approach for Lossless Compression of Color-Quantized Images*, Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2006, May 2006, Toulouse, France, Vol. II, pp. 429-432.
3. **António J. R. Neves**, Armando J. Pinho, *Lossless compression of color-quantized images using block-based palette reordering*, Proc. of the International Conference

on Image Analysis and Recognition, ICIAR 2004, September 2004, Oporto, Portugal, Vol. I, pp. 277-284.

4. Armando J. Pinho, **António J. R. Neves**, *Palette reordering under an exponential power distribution model of prediction residuals*, Proc. of the IEEE International Conference on Image Processing, ICIP 2004, October 2004, Singapore, pp. 501-504.
5. Armando J. Pinho, **António J. R. Neves**, *Variable size block-based histogram packing for lossless compression of color quantized images*, Proc. of the 4th IASTED International Conference on Visualization, Imaging, and Image Processing, VIIP 2004, September 2004, Marbella, Spain, pp. 778-782.
6. Armando J. Pinho, **António J. R. Neves**,*JPEG 2000 coding of color-quantized images*, Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2003, September 2003, Barcelona, Spain, Vol. II, pp. 181-184.
7. Armando J. Pinho, **António J. R. Neves**, *Block-based histogram packing of color-quantized images*, Proc. of the IEEE Int. Conf. on Multimedia and Expo, ICME-2003, July 2003, Baltimore, MD, Vol. I, pp. 341-344.

- The study and development of specific methods for compression of color-indexed images, presented in Chapter 4:

1. Armando J. Pinho, **António J. R. Neves**, *A context adaptation model for the compression of images with a reduced number of colors*, Proc. of the IEEE International Conference on Image Processing, ICIP 2005, September 2005, Genoa, Italy, Vol. II, pp. 738-741.

- A comprehensive study of the image compression standards applied to the compression of microarray images, presented in Chapter 6:

1. Armando J. Pinho, António R. C. Paiva, **António J. R. Neves**, *On the use of standards for microarray lossless image compression*, IEEE Trans. on Biomedical Engineering, vol. 53, no. 3, March 2006, pp. 563-566.

- The study and development of specific methods for compression of microarray images, presented in Chapter 6:

1. **António J. R. Neves** and Armando J. Pinho, *Lossless compression of microarray images*, Proc. of the IEEE International Conference on Image Processing, ICIP

2006, October 2006, Atlanta, GA, pp. 2505–2508.

2. **António J. R. Neves**, Armando J. Pinho, *Lossless bit-plane compression of microarrays images using 3D context models*, Proc. of the 5th IASTED International Conference on Visualization, Imaging, and Image Processing, VIIP 2005, September 2005, Benidorm, Spain, pp. 253-258.

Chapter 2

Lossless image compression standards

In many cases, image compression may be associated with some loss of information, for the following reasons:

- Significant loss can often be tolerated by the human visual system without interfering with perception of the scene content;
- The human visual system doesn't detect some type of losses and this can be exploited by the encoding algorithm;
- In most cases, digital input to the compression algorithm is itself an imperfect representation of the real-world scene;
- Lossless compression is usually incapable of achieving the high compression requirements of many storage and distribution applications.

However, in some cases, losses are unacceptable. Therefore, the primary goal of lossless image compression is to minimize the number of bits required to represent the original image without any loss of information. Lossless compression is often required in medical applications so as to avoid legal disputation over the significance of errors introduced into the imagery. Lossless compression is also often applied in cases where it is difficult to determine how to introduce an acceptable loss. On color quantized images, for example, a small error in the index values may have a drastic effect upon the color representation. Images with highly

structured nature, such as text and graphics, are usually compressed using lossless methods. Finally, lossless compression may be appropriate in applications where the image is to be extensively edited and recompressed, so that the accumulations of errors from multiple lossy compression operations may become unacceptable.

In this chapter, we present the state-of-the-art standards that allow lossless coding of digital images, namely JPEG-LS, JPEG2000, JBIG and PNG. They have been developed with different goals in mind: JPEG-LS is dedicated to the lossless compression of continuous-tone images; JPEG2000 was designed with the aim of providing a wide range of functionalities; JBIG is more focused on progressive lossless compression of binary and low-precision gray-level images; PNG was developed for lossless compression of computer graphics images, however supporting also grayscale and true-color images.

2.1 The JPEG-LS standard

JPEG-LS[24, 26, 80] is the state-of-the-art International Standard for lossless and near-lossless coding of continuous tone still images. It has been developed by the Joint Photographic Experts Group (JPEG) with the aim of providing a low complexity lossless image standard that could be able to offer better compression efficiency than lossless JPEG [24, 80, 76]. Part 1 of this standard was finalized in 1999. The core of JPEG-LS is based on the LOw COmplexity LOssless COmpression for Images (LOCO-I) algorithm [79], that relies on prediction, residual modeling and context-based coding of the residuals. Most of the low complexity of this technique comes from the assumption that prediction residuals follow a two-sided geometric probability distribution and from the use of Golomb codes, which are known to be optimal for this kind of distributions. Besides lossless compression, JPEG-LS also provides a lossy mode where the maximum absolute error can be controlled by the encoder. This is known as near-lossless compression or L_∞-constrained compression. The basic block diagram of JPEG-LS is given in Fig. 2.1.

A. Modeling and Prediction

Modeling in lossless image compression can be formulated as an inductive inference problem. In a raster scan, after having scanned past data, one infers the next pixel value by assigning a conditional probability distribution to it. In state-of-the-art lossless image compression schemes, this probability assignment is generally broken into the following components: (1) a prediction step, in which a deterministic value $\hat{x}_{t+1}$ is guessed for the next sample x_{t+1} based

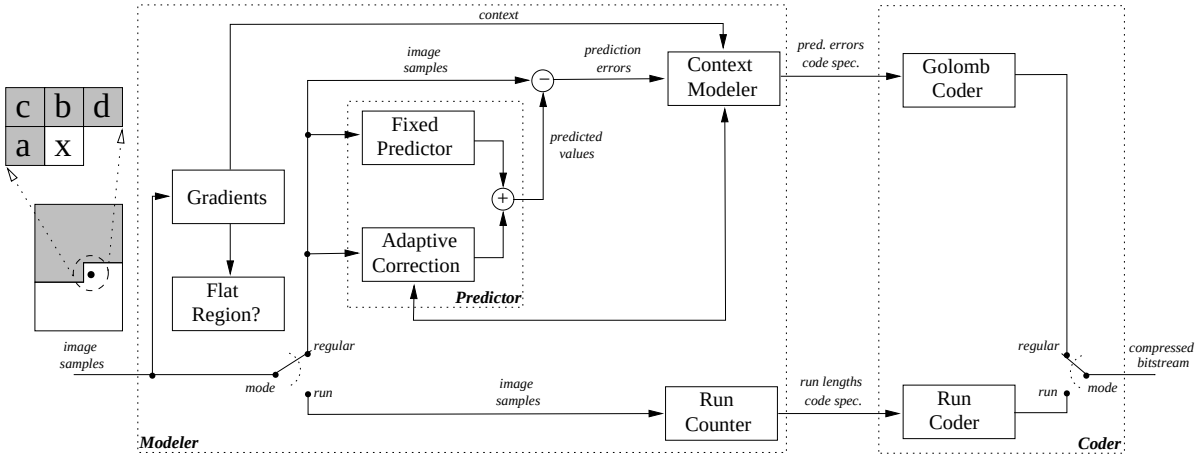


Figure 2.1: JPEG-LS: Block Diagram [80].

on a finite subset (a causal template) of the available past sequence x^t; (2) the determination of a context in which x_{t+1} occurs; (3) a probabilistic model for the prediction residual (or error signal) $\epsilon = x_{t+1} - \hat{x}_{t+1}$ (this model determines how the residual is compressed).

The prediction and modeling units in JPEG-LS are based on the causal template depicted in Fig. 2.2 where x denotes the current sample, and a, b, c and d are neighboring pixels in the relative positions shown in the figure.

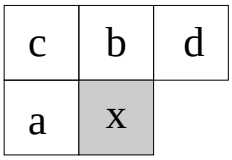


Figure 2.2: The causal template used in JPEG-LS for prediction and modeling.

The fixed predictor used in JPEG-LS

$$\hat{x}_{MED} = \begin{cases} \min(a,b), & \text{if } c \geq \max(a,b) \\ \max(a,b), & \text{if } c \leq \min(a,b) \\ a+b-c, & otherwise. \end{cases} \tag{2.1}$$

performs a simple test to detect vertical or horizontal edges. The predictor switches between

three simple predictors: it tends to choose b in the case where a vertical edge exists left of the current location, a in cases of an horizontal edge above the current location, or $a+b-c$ if no edge is detected. The latter choice would be the value of x if the current pixel belongs to the plane defined by the three neighboring pixels with heights a, b and c. This expresses the expected smoothness of the image in the absence of edges.

B. Context Modeling

It is an accepted observation, adopted by JPEG-LS, that the global statistics of residuals from a fixed predictor in continuous tone images are well modeled by a two-sided geometric distribution (TSGD) centered at zero. For context-conditioned predictors, this distribution has an offset, and this is addressed by JPEG-LS as well. For each context, there is a need to estimate the exponential decay value and center of the distribution.

The context that conditions the encoding of the current prediction residual in JPEG-LS is built out of the differences $g_1 = d - b$, $g_2 = b - c$ and $g_3 = c - a$. These differences represent an estimate of the local gradient, thus capturing the level of activity (smoothness, edginess) surrounding a sample, which governs the statistical behavior of prediction errors. For further model size reduction, each difference g_1, g_2 and g_3 is quantized into a small (fixed) number of approximately equiprobable connected regions. This aims to maximize the quantizer mutual information between the current sample value and its context, an information-theoretic measure of the amount of information provided by the conditioning context on the sample value to be modeled.

C. Coding

To encode bias corrected prediction residuals distributed according to the TSGD, JPEG-LS uses a minimal complexity family of optimal prefix codes for TSGD, sequentially selecting the code among this family.

Golomb codes were first described in [16] as a means for encoding run lengths. Given a positive integer parameter m, the Golomb code G_m encodes an integer $y \geq 0$ in two parts: a unary representation of $\lfloor y/m \rfloor$ and a modified binary representation of $y \mod m$ (using $\lfloor \log_2 m \rfloor$ bits if $y < 2^{\lceil \log_2 m \rceil} - m$ and $\lceil \log_2 m \rceil$ bits otherwise). Golomb codes are optimal for one-sided geometric distributions of the nonnegative integers [15], i.e., distributions of the form $(1-\theta)\theta^y$, where $0 < \theta < 1$. Thus, for every θ there exists a value of such leading to the shortest average code length over all uniquely decipherable codes for the nonnegative integers.

The special case of Golomb codes with $m = 2^k$ leads to very simple encoding/decoding procedures. The code for y is constructed by appending the k least significant bits of y to the unary representation of the number formed by the remaining higher order bits of y. The length of the encoding is $k + 1 + \lfloor y/2^k \rfloor$.

In order to use these codes, the TSGD has to be first mapped into one-sided geometric distributions. In JPEG-LS, this mapping is done using the following equation:

$$M(\epsilon) = 2|\epsilon| - u(\epsilon),$$

where function $u(\epsilon) = 1$ if $\epsilon < 0$ or 0 otherwise.

To encode flat regions, JPEG-LS addresses the problem by embedding an alphabet extension into the context conditioning. Specifically, the encoder enters in a differently encoded "run" mode when a context with $a = b = c = d$ is detected, as this indicates a flat region. Since the central region of quantization for the gradients g_1, g_2, g_3, is the singleton $\{0\}$, the run condition is easily detected in the process of context quantization by checking for the quantizer context $[g_1, q_2, q_3] = [0, 0, 0]$.

D. Near-Lossless Compression

JPEG-LS offers a lossy mode of operation, termed "near-lossless", in which every sample value in a reconstructed image component is guaranteed to differ from the corresponding value in the original image by up to a preset (small) amount, δ. JPEG-LS is the only standard currently supporting this mode of operation.

The basic technique employed for achieving this near-lossless or controlled-lossy in JPEG-LS is the traditional DPCM loop, where the prediction residual (after correction and possible sign reversion, but before modulo reduction) is quantized into quantization bins of size $2\delta+1$, with reproduction values at the center of the interval (thereby giving a maximal error of δ).

Context modeling and prediction are based on reconstructed values, so that the decoder can mimic the operation of the encoder. The condition for entering the run mode is relaxed to require that the gradients g_i, $i = 1, 2, 3$ satisfy $|g_i| \leq \delta$. This relaxed condition reflects the fact that reconstructed sample differences up to δ can be the result of quantization errors. Moreover, once in run mode, the encoder checks for runs within a tolerance of δ while reproducing the value of the reconstructed sample at a. Consequently, the run interruption contexts are determined according to whether $|a - b| \leq \delta$ or not. The relaxed condition for the run mode also determines the central region for quantized gradients, which is $|g_i| \leq \delta$,

$i = 1, 2, 3$. Thus, the size of the central region is increased by 2δ. Consequently, the default thresholds for gradient quantization are scaled accordingly.

E. Multicomponent and Palettized images

For encoding images with more than one component (e.g., color images), JPEG-LS supports combinations of single-component and multicomponent scans. For multicomponent scans, a single set of context counters (namely, A, B, C and N for regular mode context) is used across all components in the scan. Prediction and context determination are performed as in the single component case and are component independent. Thus, the use of possible correlation between color planes is limited to sharing statistics, collected from all planes. For some color spaces (e.g., RGB), good decorrelation can be obtained through simple lossless color transforms [5] as a preprocessing step to JPEG-LS. The data in a multicomponent scan can be interleaved either by lines (line-interleaved mode) or by pixels (pixel-interleaved mode).

The JPEG-LS data format also provides tools for encoding palletized images in an appropriate index space (i.e., as an array of indexes to a palette table), rather than in the original color space. To this end, the decoding process may be followed by a so-called sample-mapping procedure, which maps each decoded sample value (e.g., and 8-bit index) to a reconstructed sample value (e.g., an RGB triplet) by means of mapping tables. Appropriate syntax is defined to allow embedding of these tables in the JPEG-LS bit stream. Many of the assumptions for the JPEG-LS model, targeted at continuous-tone images, do not hold when compressing an array of indexes. However, an appropriate reordering of the palette table can sometimes alleviate this deficiency, as we present in Chapter 5.

2.2 The JPEG2000 standard

JPEG2000 [25, 73, 11, 76] is the most recent international standard for still image compression (Part 1 was published as an International Standard in the year 2000). This standard is based on wavelet technology and embedded block coding (EBCOT) of the wavelet coefficients [77, 75], providing very good compression performance for a wide range of bit rates, including lossless coding. Moreover, JPEG2000 allows the generation of embedded codestreams, meaning that from a higher bit rate stream it is possible to extract lower bit rate instances without the need for re-encoding.

This compression system allows great flexibility, not only for the compression of images but

also for the access into the compressed data. The codestream provides a number of mechanisms for locating and extracting data for the purpose of retransmission, storage, display or editing. This access allows storage and retrieval of data appropriate for a given application, without decoding.

The block diagram of the JPEG2000 encoder is illustrated in Fig. 2.3. The discrete wavelet transform (DWT) is first applied to the source image data. The transform coefficients are then quantized and entropy coded, before forming the output codestream (bitstream). The decoder is the reverse of the encoder: the codestream is first entropy decoded, dequantized and inverse discrete transformed, thus resulting in the reconstructed image data. Before proceeding with the details of each block of encoder in Fig. 2.3, it should be mentioned that the standard works on image tiles.

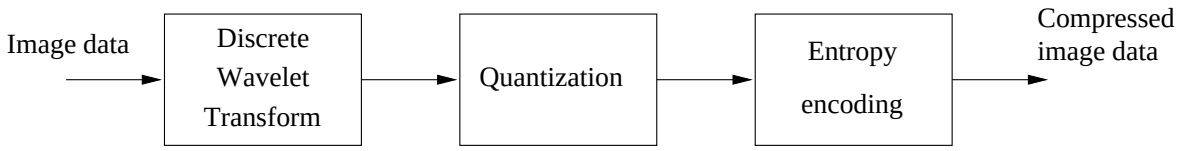


Figure 2.3: JPEG2000: Block Diagram.

The term "tiling" refers to the partition of the original (source) image into rectangular non-overlapping blocks (tiles), which are compressed independently, as though they were entirely distinct images (see Fig. 2.4). Prior to computation of the discrete wavelet transform on each image tile, all samples of the image tile component are DC level shifted by subtracting the same quantity (i.e., 2^{b-1}, where b is the component depth). DC level shifting is performed on samples of components that are unsigned only. Arithmetic coding is used in the last part of the encoding process. The binary MQ-coder is adopted in JPEG2000. This coder is basically similar to the QM-Coder adopted in the original JPEG standard. The MQ-coder is also used in the JBIG2 standard [27].

A. Discrete Wavelet Transform

The tile components are decomposed into different levels using the DWT. The resulting subbands contain coefficients that describe the horizontal and vertical characteristics of the original tile component. This process of applying DWT is then repeated a number of times on the low-resolution image block using the dyadic decomposition represented in Fig. 2.5.

To perform the DWT, the standard uses a 1-D subband decomposition of a 1-D set of samples

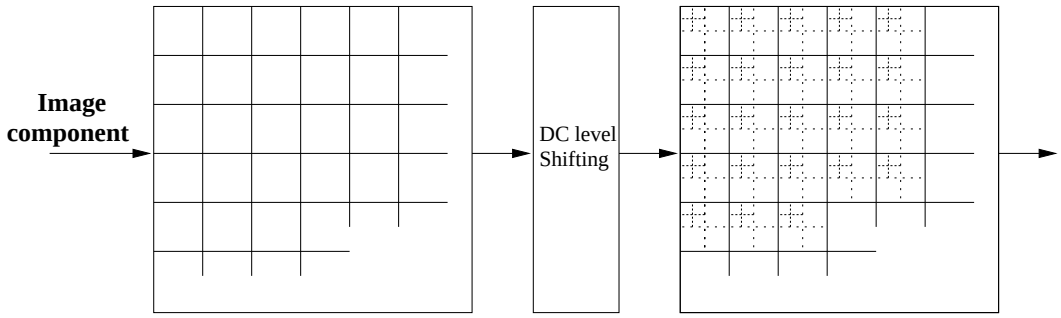


Figure 2.4: Tiling, DC level shifting and DWT of each image component.

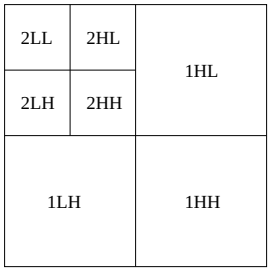


Figure 2.5: A representation of the dyadic decomposition (L indicates low-pass filtering, whereas H means high-pass filtering).

into low-pass samples and high-pass samples. Low-pass samples represent a downsampled low-resolution version of the original set. High-pass samples represent a downsampled residual version of the original set, needed for the perfect reconstruction of the original set from the low-pass set. The DWT can be irreversible or reversible. The default irreversible transform is implemented by means of the Daubechies 9/7 filter. The default reversible transformation is implemented by means of the 5/3 filter with integer coefficients.

B. Quantization

After transformation, all coefficients are quantized. Scalar quantization is used in Part I of the standard. Quantization is the process by which the coefficients are reduced in precision. This operation is lossy, unless the quantization step is 1 and the coefficients are integers, as produced by the reversible integer 5/3 wavelet. Each of the transform coefficients $a_b(u, v)$ of

the subband b is quantized to the value $q_b(u,v)$ according to

$$q_b(u,v) = \text{sign}(a_b(u,v)) \left\lfloor \frac{|a_b(u,v)|}{\Delta_b} \right\rfloor .$$

The quantization step Δ_b is represented relative to the dynamic range R_b of subband b, by the exponent ϵ_b and mantissa μ_b as

$$\Delta_b = 2^{R_b - \epsilon_b} \left(1 + \frac{\mu_b}{2^{11}}\right) .$$

The dynamic range R_b depends on the number of bits used to represent the original image tile component and on the choice of the wavelet transform. All quantized transform coefficients are signed values even when the original components are unsigned. These coefficients are expressed in a sign+magnitude representation prior to coding. For reversible compression, the quantization step size is required to be 1. This implies that $\mu_b = 0$ and $R_b = \epsilon_b$.

C. Arithmetic coding

After quantization, each subband is divided into rectangular blocks. Three spatially consistent rectangles (one from each subband at each resolution level) comprise a packet partition. Each packet partition location is further divided into non-overlapping rectangles, called "code-blocks", which form the input to the entropy coder. The individual bitplanes of the coefficients in a code-block are coded within three coding passes. Each of these coding passes collects contextual information about the bitplane data. An arithmetic coder uses this contextual information and its internal state to decode a compressed bit-stream. Different termination mechanisms allow different levels of independent extraction of this coding pass data.

The coded data of each code-block is distributed across one or more layers in the codestream. Each layer consists of a number of consecutive bitplane coding passes from each code-block in the tile, including all subbands of all components for that tile. The number of coding passes in the layer may vary from code-block to code-block and may be as little as zero for any or all code-blocks. Each layer successively and monotonically improves the image quality, so that the decoder is able to decode the codeblock contributions contained in each layer in sequence. For a given code-block, the first coding pass in layer n is the coding pass immediately following the last coding pass for the code-block in layer $n - i$, if any.

Each bitplane of a code-block is scanned in a particular order. Starting from the top left, the first four bits of the first column are scanned. Then the first four bits of the second column, until the width of the code-block is covered, Then the second four bits of the first column are

scanned and so on. A similar vertical scan is continued for any leftover rows on the lowest code-blocks in the subband.

Code-blocks are then coded a bitplane at a time starting from the most significant bitplane with a non-zero element to the least significant bitplane. For each bitplane in a code-block, a special code-block scan pattern is used for each of three coding passes. Each coefficient bit in the bitplane is coded in only one of the three coding passes. The three coding passes are: significance propagation, magnitude refinement, and cleanup. For each pass, contexts are created which are provided to the arithmetic coder.

In the JPEG2000 standard, all coding is done using context dependent binary arithmetic coding . The recursive probability interval subdivision of Elias coding is the basis for the binary arithmetic coding process. With each binary decision, the current probability interval is subdivided into two sub-intervals, and the codestream is modified (if necessary) so that points to the base (the lower bound) of the probability sub-interval assigned to the symbol, which occurred. Since the coding process involves addition of binary fractions rather than concatenation of integer codewords, the binary decisions more probable can often be coded at a cost of much less than one bit per decision.

JPEG2000 uses no more than 9 contexts for any given type of bit. This allows rapid probability adaptation and decreases the cost of independently coded segments. The context models are always reinitialized at the beginning of each code-block and the arithmetic coder is always terminated at the end of each block (i.e., once at the end of the last sub-bitplane). This is useful for error resilience also.

In addition to the above, a coding mode is used to reduce the number of symbols that are arithmetically coded. According to this mode, after the fourth bitplane is coded, the first and second pass are included as raw (uncompressed data), while only the third coding pass of each bitplane employs arithmetic coding.

2.3 The JBIG standard

Joint Bi-level Image Experts Group (JBIG) [23] was issued in 1993 by the International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) and Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T) for the progressive lossless compression of binary images. The major advantages of JBIG over other existing standards, such as FAX Group 3/4, are its capability of progressive

encoding and its superior compression efficiency [18, 41, 70]. The term "progressive encoding" means that the image is saved in several "layers" in the compressed stream. When an image is decompressed and viewed, the viewer first sees an imprecise image (first layer) followed by improved versions (higher layers).

Even though JBIG was designed for bi-level images, it is possible to apply it to grayscale images by separating the bitplanes and compressing each individually, as if it was a bi-level image. In this case, the use of Gray Code, instead of the standard binary code, may improve the compression efficiency [1].

The core of JBIG consists of an adaptive finite-context model followed by arithmetic coding. A finite-context model (see Fig. 2.6) of an information source assigns probability estimates to the symbols of an alphabet $\mathcal{A}$, according to a conditioning context computed over a finite and fixed number, M, of past outcomes (order-M finite-context model) [68, 67, 72]. At time t, we represent these conditioning outcomes by $c^t = x_{t-M+1}, \ldots, x_{t-1}, x_t$. The number of conditioning states of the model is $|\mathcal{A}|^M$, dictating its complexity (or model cost).

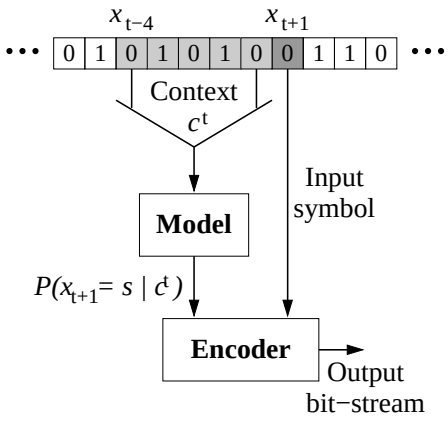


Figure 2.6: Finite-context model: the probability of the next outcome, x_{t+1}, is conditioned by the M last outcomes. In this example, $M = 5$.

In practice, the probability that the next outcome, x_{t+1}, is $s \in \mathcal{A}$, is obtained using the following estimator:

$$P(x_{t+1} = s|c^t) = \frac{n(s, c^t) + \delta}{\sum_{a \in \mathcal{A}} n(a, c^t) + |\mathcal{A}|\delta},$$

where $n(s, c^t)$ represents the number of times that, in the past, the information source generated symbol s having c^t as the conditioning context. The parameter $\delta > 0$, besides allowing fine tuning the estimator, avoids generating zero probabilities when a symbol is encoded for

the first time. The counters are updated each time a symbol is encoded. Since the context template is causal, the decoder is able to reproduce the same probability estimates without needing additional information.

The context model used in JBIG rely on 1024 contexts when operating in sequential mode or on low resolution layers of the progressive mode, or 4096 contexts when encoding high resolution layers. For each pixel, JBIG examines a template made of the 10 neighboring pixels (see Fig. 2.7), marked as "X" and "A", and based on the value of these pixels choose the respective statistical model that will be used to encode the current pixel, marked as "?".

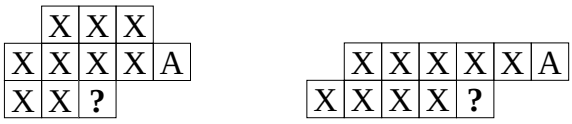


Figure 2.7: Templates for the lowest resolution layer. On the left, the three lines template. On the right, the two lines template.

Figure 2.7 shows the two templates used for the sequential mode and for the low resolution mode. The encoder decides whether to use the three-line or the two-line template and indicates this choice in the bitstream (the two-line template results in a somewhat faster execution and the three-line template produces slightly better compression). The template pixel labeled "A" is called *adaptive pixel* (AP). The encoder is allowed to use as AP a pixel outside the template and it uses two parameters in each layer to indicate the position of the AP in that layer.

More recently, a new version, named JBIG2, has been published [27], introducing additional functionalities to the standard, such as multipage document compression, two modes of progressive compression, lossy compression and differentiated compression methods for different regions of the image (e.g., text or halftones) [70].

2.4 The PNG standard

Portable Network Graphics (PNG) [22] is an extensible file format for the lossless, portable, well-compressed storage of raster images. Color-indexed, grayscale, and truecolor images are

supported, with optional transparency (alpha channel). The images can have sample depths range from 1 to 16 bits.

PNG is designed to work well in online viewing applications, such as the World Wide Web, allowing a progressive display option using a 2-D interlacing algorithm. This algorithm, named Adam7, uses seven passes to send the complete picture. In the first pass only 1 out of 64 pixels is transmitted, which results in a good approximation of the original image. PNG is robust, providing both full file integrity checking and simple detection of common transmission errors. Also, PNG can store gamma and chromaticity data for improved color matching on heterogeneous platforms.

The core of PNG's compression scheme is a descendant of the LZ77 algorithm [87] known as the deflate algorithm [42]. Deflate is comparable to LZW in both encoding and decoding speed and generally compresses better. In simplest terms, deflate uses a sliding window of up to 32 kilobytes, with a Huffman encoder [70] on the back end. Encoding involves finding the longest matching string (or at least a long string) in the 32 KB window immediately prior to the current position, storing it as a pointer (distance backward) and a length, and advancing the current position and the window accordingly.

Deflate limits match-lengths to between 3 and 258 bytes. One of the consequences of the length limits is that there must be some alternate mechanism to encode sequences of less than three bytes — particularly single bytes. In order to prime the sliding window and to accommodate bytes in the input stream that don't appear anywhere in the sliding window, the algorithm must be able to encode plain characters, or "literals". It means that there are three kinds of symbols rather than two: lengths, distances, and literals. These three alphabets are the input for the Huffman stage of the deflate engine. Deflate actually merges the length and literal codes into a single alphabet of 286 symbols. A similar approach is used for the distance alphabet. The two alphabets, lenghts/literals and distances, are fed to the Huffman encoder and compressed with either fixed or dynamic Huffman codes.

PNG also supports a pre-compression step called filtering. Filtering is a method of reversible transforming the image data so that the main compression engine can operate more efficiently. As a simple example, consider a sequence of bytes increasing uniformly from 1 to 255. Since there is no repetition in the sequence, it compresses either very poorly or not at all. But a trivial modification of the sequence — namely, leaving the first byte alone but replacing each subsequent byte by the difference between it and its predecessor — transforms the sequence into an extremely compressible set of 255 identical bytes, each having the value 1.

Actual image data is rarely that perfect, but filtering does improve compression in grayscale and truecolor images, and it can help on some palette images as well. PNG supports five types of filters, and an encoder may choose to use a different filter for each row of pixels in the image:

- None: each byte is unchanged;
- Sub: each byte is replaced with the difference between it and the "corresponding byte" to its left.
- Up: each byte is replaced with the difference between it and the byte above it (in the previous row, as it was before filtering).
- Average: each byte is replaced with the difference between it and the average of the corresponding bytes to its left and above it, truncating any fractional part.
- Paeth: each byte is replaced with the difference between it and the Paeth predictor of the corresponding bytes to its left, above it, and to its upper left.

The last method requires some explanation. Invented by Alan Paeth [50], the Paeth predictor is computed by first calculating a base value, equal to the sum of the corresponding bytes to the left and above, minus the byte to the upper left. Then, the difference between the base value and each of the three corresponding bytes is calculated, and the byte that gave the smallest absolute difference — that is, the one that was closest to the base value — is used as the predictor and subtracted from the target byte to give the filtered value. In case of ties, the corresponding byte to the left has precedence as the predicted value, followed by the one directly above. Note that all calculations to produce the Paeth predictor are done using exact integer arithmetic. The final filter calculation, on the other hand, is done using base-256 modular arithmetic; this is true for all of the filter types.

Though the concept is simple, there are quite a few subtleties in the actual mechanics of filtering. Most important among these is that filtering always operates on bytes, not pixels. For images with pixels smaller than eight bits, this means that the filter algorithms actually operate on more than one pixel at a time; for example, in a 2-bit palette or grayscale image, there are four pixels per byte. This approach improves the efficiency of decoders by avoiding bit-level manipulations.

2.5 Software

All the experimental results presented in this thesis, regarding the use of the standard image compression methods described in this chapter, have been obtained using the following implementations:

- JPEG-LS: we used the implementation provided by the Signal Processing & Multimedia Group at the University of British Columbia (SPMG / JPEG-LS V.2.2 codec). A copy of this package can be found in `ftp://ftp.ieeta.pt/~ap/codecs`.
- JPEG2000: we used the implementation provided by the Jasper project, an open-source initiative to provide a free software-based reference implementation of the codec specified in the JPEG-2000 Part-1 standard. In our experiments we used the version 1.700.2. This project can be found in `http://www.ece.uvic.ca/~mdadams/jasper/`.
- PNG: we used the implementation provided by the Netpbm project. Netpbm is a package with several converters and tools for image manipulation. We used the version 10.0. This project can be found in `http://netpbm.sourceforge.net/`.
- JBIG: we used the implementation provided by the JBIG-KIT project. JBIG-KIT implements the specification of International Standard ISO/IEC 11544:1993 and ITU-T Recommendation T.82(1993), commonly referred to as the JBIG1 standard. We used the version 1.6. This project can be found in `http://www.cl.cam.ac.uk/~mgk25/jbigkit/`.

Chapter 3

Lossless compression of simple images

The majority of the image compression techniques strongly rely on the assumption that images are locally smooth. In fact, this assumption is verified by most of the images representing natural content. Since this class of images has been traditionally used for testing the performance of the compression techniques, it is not surprising that these techniques closely match the main characteristics of the images. However, the supremacy of "natural images" has progressively been diminishing, given place to images whose contents also include text, graphical and computer generated materials, besides the usual natural content.

This diversity of contents poses some important challenges to the general purpose techniques which have been designed with the aim of compressing natural images. Normally, this problem originates a degradation in the compression rate, affecting both lossy and lossless techniques.

Special purpose compression techniques, such as Embedded Image-Domain Adaptive Compression (EIDAC) [81, 82] or Runs of Adaptive Pixel Patterns (RAPP) [66], which have been designed specifically for compressing "simple images". This designation is used to indicate images that do not use all available intensities. These methods generally attain better compression results on this class of images, comparatively to the results obtained using standards. However, they are not widely available as standards are. Moreover, as we will show in this chapter, the overall compression results obtained with the help of preprocessing techniques can substantially improve the compression results obtained using standards like JPEG-LS [24, 80] or lossless JPEG2000 [25, 37].

In this chapter, we are specially interested in studying preprocessing methods that can be used together with standard image coding techniques with the aim of improving the lossless compression of images that do not fall into the "natural" class. We are particularly interested in images having a sparse histogram of intensities. Moreover, we go further, and we expand the analysis from a global and strictly sparse perspective, to a more general analysis, incorporating concepts such as local sparseness and quasi-sparseness of the histograms of image intensities. We show that even very simple preprocessing techniques are able to cut down the lossless compression ratios provided by state-of-the-art image coding techniques, such as JPEG-LS or lossless JPEG2000, sometimes dramatically.

3.1 Motivation

In this section, we provide evidence to show why this type of images originates a degradation in the compression rate when standards like JPEG-LS or JPEG2000 are used. Moreover, this evidence is the motive for the development of the preprocessing techniques described in Section 3.3.

3.1.1 Images with sparse histograms

Figure 3.1 shows a gray-scale image, "gate", and its histogram. As can be observed, the histogram of this image is reasonably sparse: only 69 different intensities are used, out of the possible 256. Moreover, because they spread all over the $[0, 255]$ interval, a total of 48 holes can be found in the histogram. We consider that a histogram has a hole whenever between two consecutive non-zero bins there are one or more contiguous zero bins. Using the JPEG-LS standard for compressing the "gate" image we obtain a compressed file size of 27 656 bytes. However, if prior to compression we apply an order-preserving mapping from the 69 intensity values used in the image into the subset of the integers $\{0, 1, \ldots, 68\}$, then the compressed file size is reduced to 20 647 bytes. After taking into account the overhead required for storing the mapping, (in this case 71 bytes), we still obtain an overall reduction in the compressed file size of about 25%. Similar results would have been obtained if, instead of using a JPEG-LS coder, other general purpose lossless image compression encoders had been used, such as lossless JPEG2000.

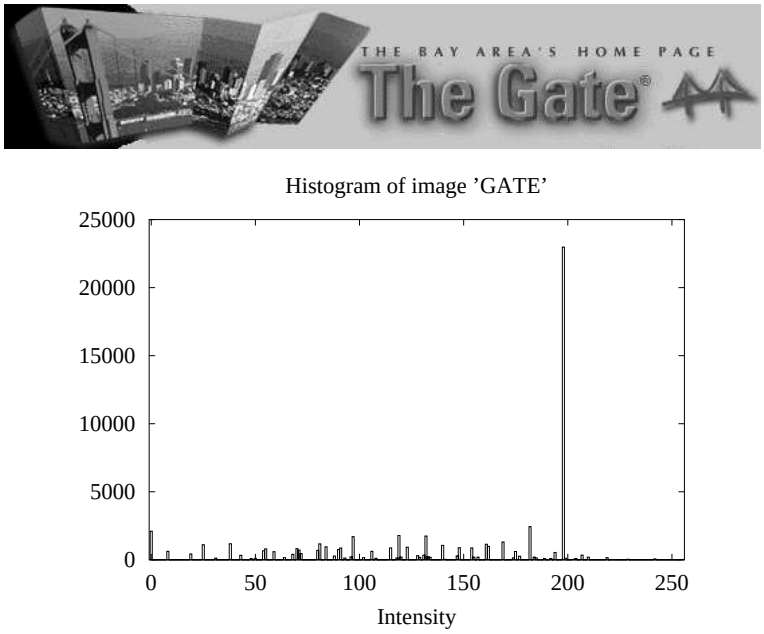


Figure 3.1: The "gate" image (108 rows × 564 columns, 69 different intensity values). The histogram of this image is also displayed.

3.1.2 Images with quasi-sparse histograms

Besides the images with sparse histograms, we also have images where some intensity values appear only once or just a few number of times in the image. We can say that these images have a "quasi-sparse" histogram.

Figure 3.2 shows a 8 bits per pixel, 59×460 image, "yahoo", which uses 156 different intensity values. After applying an order-preserving mapping from the 156 intensity values used in the image into the subset of the integers $\{0, 1, \ldots, 156\}$, this image occupies 8 401 bytes, instead of 8 822 bytes when encoded directly with JPEG-LS (a gain of 4.8%). The analysis of the packed histogram of the "yahoo" image (Fig. 3.3) reveals that it still maintains a sparse appearance, even after been packed. However, in a strict sense, it is not sparse, because all bins concerning intensities lower that 156 are non-zero, although some of them account only for a few occurrences.

Based on this observation, we can formulate the following questions:

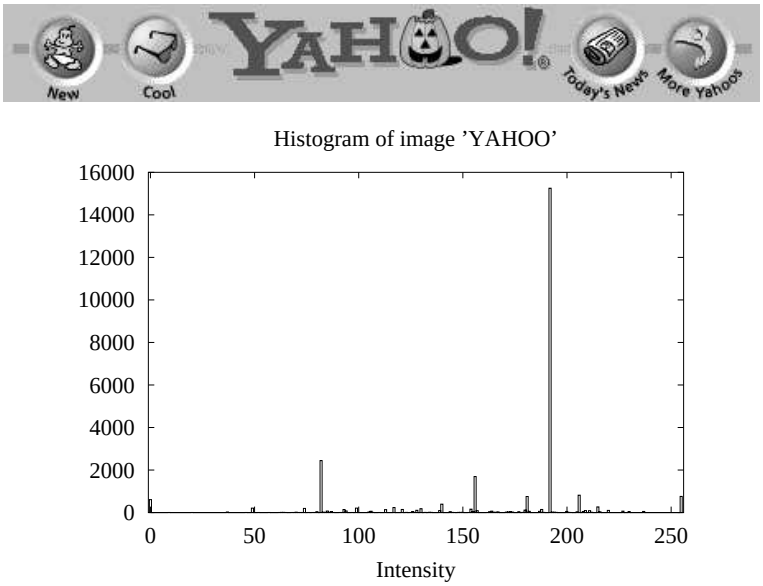


Figure 3.2: The "yahoo" image (59 rows × 460 columns, 156 different intensity values). The histogram of this image is also displayed.

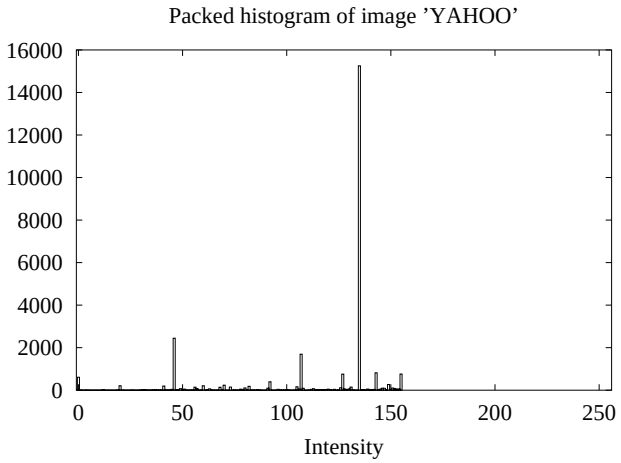


Figure 3.3: The packed histogram of the "yahoo" image.

- Is there a way to avoid the intensities that have a very low number of occurrences

and, therefore, to transform the quasi-sparse histograms in histograms that are strictly sparse?

- If this is possible, is it advantageous in terms of overall compression gain?

In Section 3.4 we try to answer these questions.

3.2 Specialized image compression techniques for simple images

The compression of simple images can be addressed using two different approaches. One of these approaches relies on the use of standard lossless image coding techniques combined with an appropriate preprocessing technique, as we will study in detail in Section 3.3. The second approach is the development of specialized coding techniques. Among the most successful specialized methods we can point out, for example, Embedded Image-Domain Adaptive Compression (EIDAC) [81, 82] and Runs of Adaptive Pixel Patterns (RAPP) [66]. EIDAC compresses the image in a bitplane basis, going from the most significant bitplane to the least significant bitplane, using a context model with pixels from the bitplane under compression and pixels from the bitplanes already encoded. RAPP uses relative pixel patterns within four pixel template. We now discuss these methods and their characteristics with more detail.

3.2.1 Embedded Image-Domain Adaptive Compression (EIDAC)

Yoo *et al.* proposed a context-based coding technique, named Embedded Image-Domain Adaptive Compression (EIDAC) [81]. This technique, based on bitplane coding, was designed to efficiently exploit the characteristics of simple images.

EIDAC compresses the bitplanes of the image successively, from the most significant bitplane (MSB) to the least significant bitplane (LSB), using an adaptive binary arithmetic coder. The causal context model used by the adaptive arithmetic coder uses two sets of bits: one set containing neighboring bits of the bitplane currently being encoded, C_{intra}, and other containing bits from the above bitplanes already coded, C_{inter}. The reasons behind using separate C_{intra} and C_{inter} are explained next.

On one hand, the strong spatial correlation among image pixels extends to their truncated versions, (e.g., when only the MSB's are used). Thus, assuming that we are about to code the

i^{th} bit of a given pixel, if this pixel has strong correlation with the neighboring pixels, then it i^{th} bit can be efficiently predicted using the i^{th} bit of its causal neighbors. This motivates the use C_{intra}, i.e., causal context information of the current bitplane.

On the other hand, C_{inter} is particularly useful for simple images. Consider, for example, an 8-bit image obtained by scanning a bi-level (black-and-white) image, for which the two most likely intensity levels are 00000000 and 11111111. There are possibly a few levels that differ only in the least significant bits from these two predominant levels. If a particular pixel's MSB is 1 for this image, it is highly likely that its other bits are also 1. Thus, it is possible to achieve compression gain by using information about the upper (or more significant) bitplanes. This justifies the potential benefits of C_{inter}. The situation is similar for other images as long as their active gray-scale levels are still much less than the maximum of 256.

Yoo *et al.* implemented the EIDAC algorithm using 4 bits for C_{intra}, as we can see in Fig. 3.4. The number of bits used in C_{inter} depends on the bitplane and can be as many as 15 bits (Fig. 3.5).

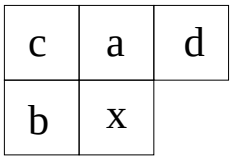


Figure 3.4: C_{intra} context configuration of the EIDAC method.

For instance, any bits in the upper bitplanes (i.e., the bitplanes already encoded) can be used to define C_{inter}, while a larger causal neighborhood can be considered to define C_{intra}. In fact, for images with strong spatial correlation, it is possible to achieve additional compression gain by including the surrounding pixels in the upper bitplanes. However, the number of bits used for context modeling cannot arbitrarily increase due to the associated increase of computational complexity and also due to the context dilution problem [7]. Once the context model has been selected, the compression algorithm processes each bitplane in the usual raster scan order.

To improve the compression efficiency of the algorithm, Yoo *et al.* suggests a preprocessing technique, denoted histogram compaction. In this case, it is possible to use side information to specify which pixel values are in use and to represent each pixel intensity with a reduced number of bits. For example, suppose that only 29 active pixel values are used in a given

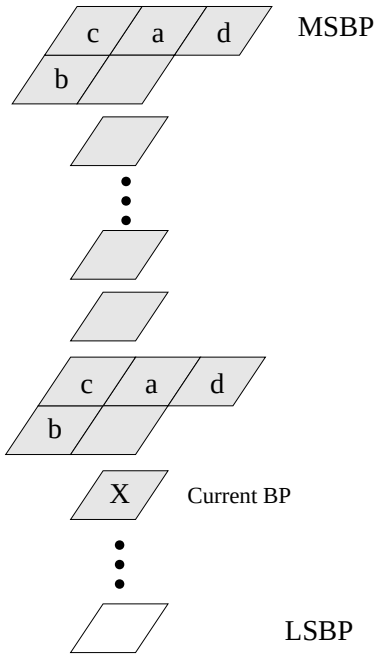
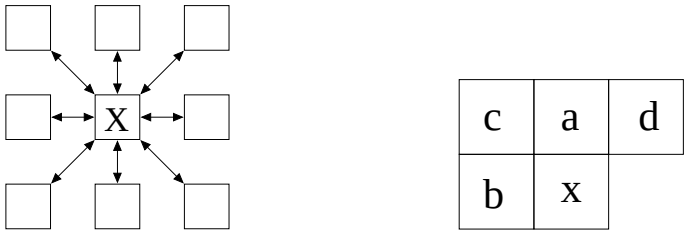


Figure 3.5: C_{inter} context configuration of the EIDAC method [81].

image. Then, it is possible to represent the original image using only 5 bits per pixel (i.e, it is only necessary to encode 5 bitplanes, instead of 8) after specifying those 29 levels explicitly as side information.

Later, Yoo *et al.* proposed a new approach [82], based on the previously described method, where it is used a more sophisticated context model. This improved context model is designed to exploit the shape information from the upper bitplanes when encoding the bits in the k^{th} bitplane. Figure 3.6 defines the 8-way connectivity model for C_{inter}. Instead of the bits from the upper bitplanes, the upper bitplane connectivity for the pixel pair is used in this case (i.e., 1 for having the same bit values in the upper bitplanes and 0 otherwise). This type of C_{inter}, along with C_{intra} in Fig. 3.6, can provide a more versatile context model in the presence of varying pixel values and complex shape information in the pixel's neighborhood.

Yoo *et al.* uses a different context model for the MSB, since it cannot exploit C_{inter}. For the MSB it is employed a more complex C_{intra} to take into account the connectivity model that allows to utilize the shape information from the neighboring bits in the MSB. The difference between the current bit and its neighboring bits (i.e., 1 when two bits are the same and 0



(a) (b)

Figure 3.6: (a) The 8-way connectivity model for C_{inter} (b) The context model used for C_{intra} [82].

otherwise), rather than the bit value, is used for the input to EIDAC, as it is found more advantageous for efficient compression of MSB. This special context C_{intra} for the MSB is shown in Fig. 3.7. It uses 8 connectivity values (indicated by the arrows) and 4 neighboring bits (indicated by letters a, b, c and d), thus having 212 possible contexts.

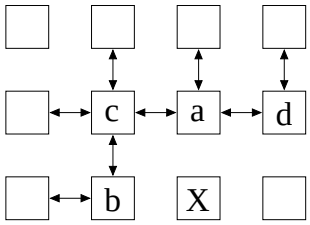


Figure 3.7: C_{intra} for the MSB bitplane [82].

For the other bitplanes, a test is made to find matching pixels in the causal neighborhood so that the current bit is encoded differently. In this test, matching pixels for the current bitplane are defined as the pixels having the same bit values in all of the upper bitplanes. When a matching pixel is detected, instead of the current bit value, the matching indicator bit is encoded, i.e., "1" indicates that the match extends to the current bitplane, while "0" indicates a mismatch. The reason to include this matching test is the following: when two pixels are identical in the upper bitplanes, we have a matching bit also in the current bitplane with high probability. For the upper bitplane, the west, north, and north-west pixel locations are tested in order. If there are multiple matches, the first detected matching pixel is used to

encode the matching indicator bit based on the context for the matching indicator. If there is no match, the current bit value is encoded using the C_{inter} and C_{intra} introduced earlier.

Instead of using histogram compaction for images that have a small number of intensities, in this case Yoo *et al.* use a new method denoted Scalable Bit-plane Reduction (SBR). SBR finds the reduced bitplane codebook by growing a binary tree for recursive bitplane partition, from the MSB towards the LSB. Each partition splits the given pixel set of the current bitplane into two disjoint subsets in the next significant bitplane, such that it minimizes the overall distortion when representing each subset by a reconstruction pixel value for the selected distortion metric. A simple MINMAX metric is used to measure the distortion, i.e., the optimal partition is the one that minimizes the maximum distortion for the resulting pixel subsets. The MINMAX metric does not require to transmit the reduced-bitplane codebook or the pixel statistics for reproduction of the codebook at the decoding end.

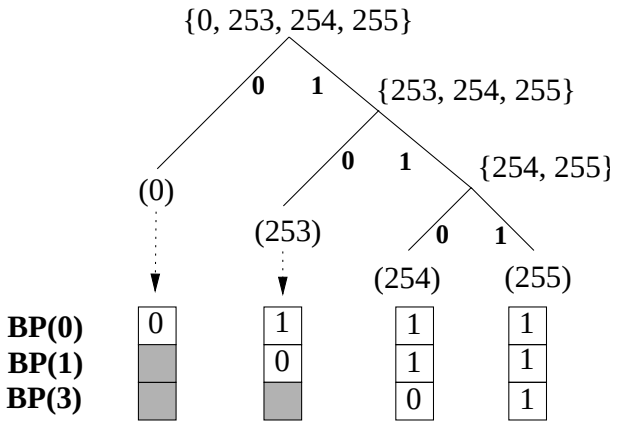


Figure 3.8: Binary tree to obtain the codebook to represent the pixel values 0, 253, 254, 255 in the reduced bit-plane space [82].

As shown in Fig. 3.8, the binary tree can be asymmetric, resulting different codeword lengths for different pixel values. EIDAC can take the advantage of this variable-length codebook by skipping encoding, for instance the "0" pixels, except in the lower bitplanes.

3.2.2 Image compression with Runs of Adaptive Pixel Patterns (RAPP)

Viresh Ratnakar developed a lossless coding method named Runs of Adaptive Pixel Patterns (RAPP) [66]. This method uses patterns of neighboring pixels (instead of their values) to

predict and code a pixel. A pixel pattern attempts to capture edges and uniform regions and their orientations. For each pattern of neighboring pixels, a prediction rule, which is adaptively learned from the image itself by both the encoder and the decoder, is used to predict the current pixel. For each pixel to be coded, the pattern of its already-coded neighboring pixels is determined. Ratnakar considered the four neighbors of the pixel to be coded, denoted X: W, NW, N and NE as showed in Fig. 3.9.

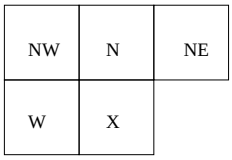


Figure 3.9: Neighbors used in the RAPP method. X denotes the pixel to be coded.

The set of basic patterns for the four-neighbor case consists of the 15 possible ways of labeling these pixels with at most four labels. Denoting the labels by the letters A, B, C and D, the set of basic patterns is: AAAA, AAAB, AABA, ABAA, ABBB, AABB, ABAB, ABBA, AABC, ABAC, ABCA, ABBC, ABCB, ABCC, ABCD. Each basic pattern is denoted by a string of four letters identifying the labels of the W, NW, N and NE neighbors, respectively. For example, the color pattern labeled AABC represents the case when the W-neighbor and NW-neighbor are identical, while the N-neighbor and NE-neighbor are distinct from them as well as from each other.

Each pattern has four probabilities (one for each label in the basic pattern) associated with it, determining the prediction and which are updated after the prediction to incorporate the actual value of the pixel. For example, suppose that the neighboring pattern of the current pixel is (AABC,(1, 1, 1, 1)). This pattern has three probabilities associated: p_A , p_B and p_C. If p_A is the greatest of these three, the predicted value for the current pixel is the same of its W-neighbor (which is the same as the NW-neighbor). If p_B is the greatest, then the prediction value is the same of its N-neighbor, otherwise, the NE-neighbor is used.

The encoder works as follows. The pixels are scanned in raster order. For each pixel, the neighborhood pattern is determined and the corresponding probabilities are used to predict its value. If the prediction is correct, then the symbol SUCCESS is sent to an output list called *TrendList*. If the prediction is incorrect, then a special symbol is inserted in *TrendList*. This symbol can have several possible values, depending on the basic pattern of the current

neighborhood. The symbol OTHER-n-m is used to indicate the case when the current pixel is equal to one of its neighbors, but not to the one that was predicted. When the current pixel is predicted incorrectly and it is not the same as any of the four neighbors, the value ANOMALY-n is inserted in *TrendList* and the value of the actual anomalous pixel is added to an other list, called *AnomalyList*.

After applying the RAPP-Transform, the lists *TrendList* and *AnomalyList* are entropy coded. Ratnakar uses Huffman or arithmetic coding for the *TrendList* and the deflate algorithm to encode the *AnomalyList*.

3.3 Preprocessing techniques

In this section we present some preprocessing methods that can be used together with standard image coding techniques with the objective of improving the lossless compression of simple images. These methods are designed for images that have a sparse histogram of image intensities.

3.3.1 Off-line histogram packing

Off-line histogram packing is a preprocessing method capable of producing improvements if applied prior to the lossless compression of images having sparse histograms. Basically, off-line histogram packing (Fig. 3.10) is obtained through the construction of an one-to-one order-preserving mapping, h, from the image intensity values, $\mathcal{I}$, into a contiguous subset of $\mathbb{N}_0$:

$$h = (I_0 \mapsto 0, I_1 \mapsto 1, \ldots, I_{N-1} \mapsto N-1),$$

where $I_i \in \mathcal{I}$, and where we assume, with loss of generality, that $I_i < I_j, \forall_{i<j}$.

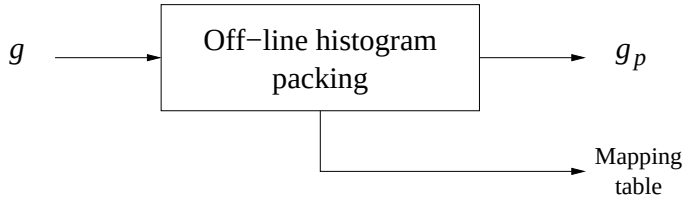


Figure 3.10: Preprocessing of image g using off-line histogram packing.

Therefore, after constructing and applying this mapping to an image g, a new image $g_p = h(g)$ is generated, hopefully more "compression-friendly" than g.

Compression gains are obtained if the storage required by the compressed version of g_p, together with the mapping table h (required for inversion of the process), is smaller than the storage required by the compressed version of g.

Off-line histogram packing requires *a priory* knowledge of the sparseness of the image histogram or, if this knowledge is not available, it requires a two-pass operation, which may not be possible or desirable in some applications.

3.3.2 On-line histogram packing

To overcome the drawback of the preprocessing technique described in the previous section (off-line histogram packing), an on-line technique was proposed in [52]. Basically, the order-preserving mapping is constructed on-the-fly, by introducing the new intensities whenever they appear for the first time in the image (based on some predefined image scanning strategy), and then the mapping is rearranged accordingly. Let us assume that the packing procedure is going to process sample[1] x_t and that it has already found $n \leq N_I$ different intensity values, $\mathcal{I}^t = \{I_0, I_1, \ldots, I_{n-1}\}$. This means that $x_k \in \mathcal{I}^t$ for $k = 1, \ldots, t-1$. Without loss of generality, we also assume that $I_i < I_j, \forall_{i<j}$. Then, the mapping h^t, used to map sample x_t, would be

$$h^t = (I_0 \mapsto 0, I_1 \mapsto 1, \ldots, I_{n-1} \mapsto n-1).$$

If $x_t \in \mathcal{I}^t$, then symbol $h^t(x_t)$ is generated by the histogram packing procedure. If not, i.e., if $x_t \notin \mathcal{I}^t$, then the symbol that is generated is n, i.e., the smallest integer not in the codomain of h^t. The purpose of this symbol is to act as an escape symbol.

Since $x_t \notin \mathcal{I}^t$, then this new intensity, $\tilde{I} = x_t$, has to be stored (for example, in an auxiliary file), in order to allow reversing the packing procedure. Moreover, the occurrence of a new intensity also implies the rearrangement of the mapping, such that, if $I_i < \tilde{I} < I_{i+1}$, then the new mapping h^{t+1} should be

$$h^{t+1} = (\ldots, I_i \mapsto i, \tilde{I} \mapsto i+1, I_{i+1} \mapsto i+2, \ldots, I_{n-1} \mapsto n),$$

which maintains the one-to-one order-preserving characteristic. In practice, this implies that the mapping values of all intensities greater than $\tilde{I}$ are increased by one unit.

[1] Assumed that the image samples have been transformed into a sample sequence, x_t, using some image scanning strategy. In this method, we used a raster-scanning approach.

As can be seen, the operation of this on-line method differs only from the off-line version until the last intensity has been found. From that point onwards, i.e., when $n = N_I$, both methods operate identically. As the off-line histogram packing method, this on-line approach operates also completely detached from the encoders, which means that it does not require any modification of the encoding algorithms. Moreover, it is also very effective, sometimes even better than its off-line counterpart. This is due to the fact that until the last intensity has been found, the mapping has a smaller number of intensities than the off-line version, which could be explored by the encoders.

3.4 Proposed preprocessing techniques

To answer the questions formulated in the begining of this chapter we propose a preprocessing method that can be used to explore the local sparseness and quasi-sparseness of the histograms of image intensities.

3.4.1 Histogram packing with a limited number of symbols

One of the major drawbacks of global off-line or on-line histogram packing is that if even most of the intensity values appear only once or just a few number of times in the image, they will be considered by the histogram packing procedure as having as much importance as those that occur most frequently. In other words, images having "quasi-sparse" histograms cannot benefit from this method. To investigate how this characteristic can be used to improve compression we propose the following approach [53, 43].

Let us denote by $\mathcal{I}$ the set of all different intensity values used by a given image, and by $S \in \mathbb{N}$ some pre-defined value. During the processing of sample x^t, which generates a transformed sample y^t at time instant t, we assume that a previously constructed subset of $\mathcal{I}$, $\mathcal{I}_n^t$, is available:

$$\mathcal{I}_n^t = \{I_0, I_1, \ldots, I_{n-1}\}, \quad n \leq S.$$

Moreover, we assume, without loss of generality, that $I_i < I_j, \forall_{i<j}$, and that the following one-to-one, order-preserving mapping in $\mathbb{N}_0$ is also available:

$$h^t = (I_0 \mapsto 0, I_1 \mapsto 1, \ldots, I_{n-1} \mapsto n-1).$$

Sample x^t is processed as follows. If $x^t \in \mathcal{I}_n^t$, then $y^t = h^t(x^t)$. However, if $x^t \notin \mathcal{I}_n^t$, then $y^t = n$, which is the first element of $\mathbb{N}_0$ that does not belong to $h^t(\mathcal{I}_n^t)$. In this case, the

intensity value $\tilde{I} = x^t$ is stored into a file, which is used for later recovery of the original image intensity values. We call this file the "recovery file".

The occurrence of an intensity not belonging to $\mathcal{I}_n^t$ also implies the rearrangement of the mapping, which depends on whether $n = S$ or not. If $n < S$, and assuming that $I_i < \tilde{I} < I_{i+1}$, then the new mapping is:

$$h^{t+1} = (\ldots, I_i \mapsto i, \tilde{I} \mapsto i+1, I_{i+1} \mapsto i+2, \ldots, I_{n-1} \mapsto n).$$

As can be seen, $\tilde{I}$ is inserted in the mapping in such a way that the one-to-one, order-preserving property is maintained. On the other hand, if $n = S$, in addition to the inclusion of $\tilde{I}$ in the mapping, as described above, it is also required the deletion of one of the members of $\mathcal{I}_n^t$ (i.e., the cardinality of the set is kept equal to S). Also in this case, the mapping has to be rearranged in order to obey to the one-to-one, order-preserving property. The algorithm is presented in Fig. 3.11.

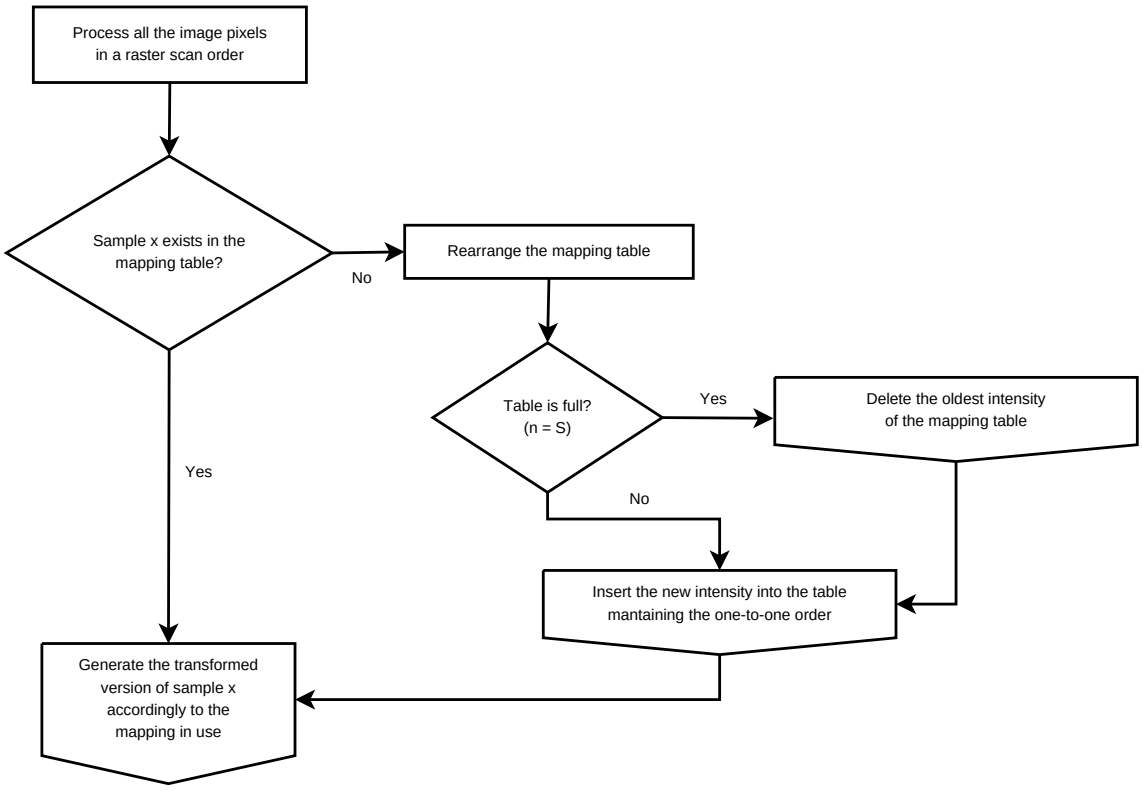


Figure 3.11: Schematic view of histogram packing with a limited number of symbols.

Decoding is performed using a similar strategy as encoding. When decoding sample y^t, if $y^t < S$, then $x^t = (h^t)^{-1}(y^t)$. [2] Otherwise, an intensity value, $\tilde{I}$, is fetched from the recovery file and $x^t = \tilde{I}$. The mapping is always reorganized following the same procedures as those performed by the encoder.

The success of this method depends, fundamentally, on how the increase in bit-rate generated due to storing the values of $x^t \notin \mathcal{I}$ is compensated by a more "compression-friendly" histogram-packed image, g_p. Figure 3.12 illustrates this tradeoff for the case of the "yahoo" and "benjerry" images. The curves represent compression gain in relation to normal JPEG-LS encoding. The black dots indicate the points where the highest compression gains were obtained, whereas the squares on the rightmost end of the curves indicate the gains attained with off-line histogram packing.

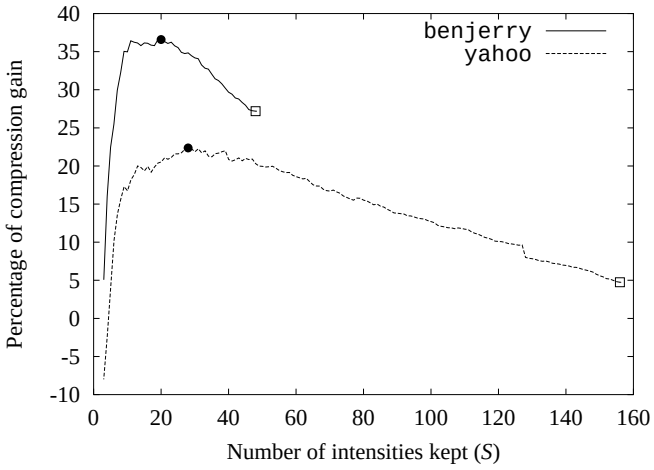


Figure 3.12: Analysis of the quasi-sparse characteristics of the histograms of the "benjerry" and "yahoo" images.

Until now, we left some important issues open, namely:

1. How the intensity values are stored in the recovery file;
2. How do we choose which intensity value is deleted from the mapping when a new value has to be inserted and $\mathcal{I}_n^t$ is not allowed to increase further (i.e., $n = S$);
3. How do we find the optimum value of S.

[2] Notation $(h^t)^{-1}$ indicates a mapping that is the inverse of h^t.

Concerning the first question, i.e., how the values are stored in the recovery file, for simplicity we store these values directly, without any kind of compression. This means that, for 8 bits per pixel images, each intensity value is stored in one byte.

The second question, i.e., how the intensity values are substituted in $\mathcal{I}_n^t$ when $n = S$, lead us to several approaches. Among them, we just point out two of the most obvious and simple: one of them calls for the removal of the oldest unused intensity; the other relies on the removal of the least used intensity. Both methods have been tested and the best results have been obtained using the removal of the oldest unused intensity.

Finally, the question regarding the optimum value of S. This image-dependent and also encoding-method-dependent parameter plays a crucial role in this preprocessing technique. In fact, depending on the value of S, we may obtain substantial compression improvements or, if incorrectly chosen, we may end up with a degradation in the compression ratio. Examples of some curves representing the compressed size of the images as a function of S can be observed in Fig. 3.13.

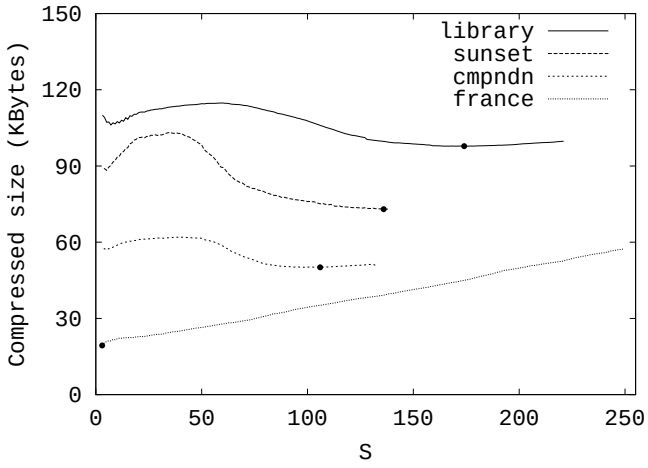


Figure 3.13: Examples of the dependence of the compressed size of the images with parameter S. These values have been obtained with the JPEG-LS encoder. The dots mark the optimum values.

To overcome the drawback of choosing the optimum value of S, we improved the described method in order to adapt the number of symbols accordingly to the image information. The method starts with a predefined number of symbols and update this number while the image

is being read. The algorithm looks at the last K processed pixels and decides what to do depending on the operations made in the mapping table. We can configure the following parameters: initial value of S, increment value, window size, and decision parameters.

3.4.2 Experimental results

In this section we present experimental results obtained with the methods described in this chapter.

To show the efficiency of the proposed technique we used three sets of images. Two of them were those used by Yoo *et al.* to test the efficiency of EIDAC [82]. The first set (corresponding to the first group of images in Tables 3.1, 3.2, and 3.3) is a gray-scale-converted version of a set used by Ausbeck in its PWC coder [4], a compression method designed for compressing palette images, that we describe with more detail in Section 4.1.

The second set (second group in Tables 3.1, 3.2, and 3.3) comprises several natural images and has the objective of testing the compression performance of the methods in images that are not "simple" (this set was also used in [82]). The third set (last group of images in Tables 3.1, 3.2, and 3.3) is composed of five images taken from the BragZone archive[3]. This set was used to illustrate the poor performance of JPEG-LS in compressing this type of images [8].

Table 3.1 presents the compression results obtained with four image compression standards, namely JPEG-LS, JPEG2000, JBIG and PNG used directly in simple images. It also contains the compression results obtained with the two most important specialized methods that have been developed for this type of images, namely EIDAC and RAPP.

According to the experimental results presented in Table 3.1 and regarding the standard methods, we reached the conclusion that for the first and third set of images PNG provides the best results. This confirms the performance of this standard when used in simple images. In addition, JPEG-LS gives the best results for the second set, which contains natural images, confirming the performance of this standard with natural images. Moreover, we can see that JBIG gives results close to the ones obtained with PNG in all sets. Regarding the specialized methods, RAPP is the best in the third set and EIDAC outperforms RAPP in the second and third sets. Finally, we can observe that the specialized methods outperforms the standards in the first and third sets, which is expected due to the fact that these methods have been designed for simple images.

[3]From `http://links.uwaterloo.ca/BragZone`.

Image	Intensities	JPEG-LS	JPEG2000	JBIG	PNG	EIDAC	RAPP
benjerry	48	1.919	4.062	1.967	1.315	1.187	0.762
books	7	5.601	6.181	2.075	1.515	1.432	1.360
cmpndd	133	1.454	2.329	1.361	1.454	1.195	1.175
cmpndn	132	1.193	2.191	1.137	1.148	0.947	0.868
gate	69	3.632	4.339	3.411	3.243	2.610	2.209
music	8	2.943	5.570	1.457	1.053	1.097	0.516
netscape	27	2.777	4.038	2.601	2.514	1.695	1.460
sea_dusk	43	0.206	0.424	0.110	0.078	0.057	0.033
sunset	138	2.175	3.102	2.063	1.996	1.674	1.513
winaw	10	1.309	2.311	0.706	0.494	0.392	0.377
yahoo	156	2.600	4.099	2.603	1.979	1.794	1.209
Average	–	**1.673**	**2.587**	**1.379**	**1.304**	**1.072**	**0.975**
aerial2	256	5.288	5.441	5.748	5.425	5.079	5.817
bikeh	220	5.516	6.083	5.627	5.838	5.319	6.207
bike	220	4.355	4.528	4.632	4.821	4.195	5.453
cafe	220	5.090	5.352	5.535	5.575	5.219	6.550
goldhill	220	4.712	4.838	5.260	5.008	5.206	6.539
lena	215	4.245	4.317	4.736	4.665	4.693	6.437
woman	220	5.387	5.458	5.536	5.793	5.346	6.455
Average	–	**4.890**	**5.088**	**5.274**	**5.268**	**4.836**	**5.981**
france	249	1.409	2.020	0.630	0.402	0.315	0.285
frog	102	6.049	6.258	6.094	6.023	3.732	3.815
library	221	5.099	5.698	4.865	5.095	4.301	4.682
mountain	110	6.421	6.701	6.429	6.606	5.381	5.213
washsat	35	4.129	4.432	4.413	3.686	2.163	2.538
Average	–	**4.529**	**4.916**	**4.378**	**4.235**	**3.041**	**3.132**

Table 3.1: Compression results, in bits per pixel (bpp), using four standars, namely JPEG-LS, JPEG-2000, JBIG and PNG, and using two specific method, EIDAC and RAPP.

Tables 3.2 and 3.3 show the comparison of the compression results (in bits/sample) obtained with JPEG-LS and JPEG2000 respectively, applied directly to the images ("Normal"), applied after off-line packing ("Off-line packing"), applied after on-line packing ("On-line packing") and applied after using the proposed on-line packing with a reduced symbol-set ("Reduced symbol-set"). Percentages represent compression gains in relation to the "Normal" values. The results presented in Tables 3.2 and 3.3 include, besides the size of encoded image, all required overhead information needed for recovering the original image.

The analysis of Tables 3.2 and 3.3 shows clearly how off-line histogram packing can improve,

Image	Intensities	Normal	Off-line packing		On-line packing		Reduced symbol-set		
		bps	bps	%	bps	%	bps	%	S
benjerry	48	1.919	1.396	27.2	1.729	9.9	1.368	28.7	26
books	7	5.601	1.882	66.4	1.881	66.4	1.638	70.8	5
cmpndd	133	1.454	1.270	12.6	1.361	6.4	1.255	13.6	108
cmpndn	132	1.193	1.050	12.0	1.146	3.9	1.043	12.5	106
gate	69	3.632	2.721	25.1	2.852	21.5	2.848	21.6	64
music	8	2.943	1.134	61.5	1.529	48.1	1.229	58.2	8
netscape	27	2.777	1.724	37.9	1.696	38.9	1.693	39.0	26
sea_dusk	43	0.206	0.176	14.3	0.188	8.8	0.073	64.4	3
sunset	138	2.175	1.963	9.7	1.947	10.5	1.946	10.5	136
winaw	10	1.309	0.546	58.3	0.571	56.4	0.524	60.0	7
yahoo	156	2.600	2.476	4.8	3.036	-16.8	2.175	16.3	3
Average	—	**1.673**	**1.230**	**26.5**	**1.293**	**22.7**	**1.201**	**28.2**	—
aerial2	256	5.288	5.289	0.0	5.051	4.5	4.292	18.8	117
bike	220	4.355	3.968	8.9	3.979	8.7	3.973	8.8	220
bikeh	220	5.516	5.139	6.8	5.272	4.4	5.201	5.7	220
cafe	220	5.090	4.941	2.9	4.945	2.8	4.941	2.9	220
goldhill	220	4.711	4.719	-0.2	4.753	-0.9	4.748	-0.8	204
lena	215	4.245	4.252	-0.2	4.278	-0.8	4.276	-0.7	209
woman	220	5.387	5.188	3.7	5.205	3.4	5.200	3.5	220
Average	—	**4.890**	**4.702**	**3.8**	**4.647**	**5.0**	**4.441**	**9.2**	—
france	249	1.411	1.406	0.3	1.415	-0.3	0.475	66.3	3
frog	102	6.048	5.175	14.4	5.043	16.6	4.566	24.5	11
library	221	5.100	5.006	1.8	5.061	0.8	4.904	3.8	174
mountain	110	6.421	5.245	18.3	5.206	18.9	5.175	19.4	95
washsat	35	4.129	2.006	51.4	2.020	51.1	2.007	51.4	35
Average	—	**4.530**	**3.653**	**19.3**	**3.627**	**19.9**	**3.263**	**28.0**	—

Table 3.2: Compression results, in bits per pixel (bpp), obtained with JPEG-LS when used after the preprocessing techniques presented in this chapter.

sometimes dramatically, the lossless compression of images that have sparse histograms, which is the case of the images in the first group and also of some images in the third group. Moreover, the degradation in the compression rates of the images that are not "simple" (those of the second group) is generally negligible.

As can be seen in Table 3.2, the results attained by the on-line packing procedure are close to those obtained with the off-line method, sometimes even better. Relatively to the JPEG2000 (Table 3.3), the on-line packing is always better than the offline conversion. This lead us to conclude that the on-line histogram packing technique can be used as a good replacement of its off-line counterpart, with the great advantage of not requiring two passes over the image.

From the observation of the "Average" rows in Tables 3.2 and 3.3, we can immediately conclude that the proposed method, histogram packing method using a limited number of symbols ("Reduced symbol-set"), provides globally better results than the off-line packing and on-line packing methods. Looking at individual images, we can notice some dramatic improvements relatively to the other methods. That is the case of image "france", which gets a compression improvement of 75% if using the JPEG2000 codec and 66% if using JPEG-LS. The compression of image "sea_dusk" improves 80% (using JPEG2000) and 64% (using JPEG-LS). Also, a considerable 19% improvement, attainable by both compression standards, can be noticed in the compression of image "aerial2". This percentages represent improvements over the use of the standard image coding methods directly in the images. Other less dramatic but also important improvements can be observed in other images.

Image	Intensities	Normal	Off-line packing		On-line packing		Reduced symbol-set		
		bps	bps	%	bps	%	bps	%	S
benjerry	48	4.027	2.765	31.3	2.690	33.2	1.922	52.3	4
books	7	6.164	2.152	65.1	2.028	67.1	1.766	71.3	5
cmpndd	133	2.326	2.009	13.6	1.620	30.4	1.572	32.4	129
cmpndn	132	2.189	1.883	13.9	1.486	32.1	1.344	38.6	3
gate	69	4.323	3.193	26.1	3.192	26.2	3.171	26.6	64
music	8	5.491	2.064	62.4	2.003	63.5	1.911	65.2	3
netscape	27	4.022	2.338	41.9	2.307	42.6	2.298	42.8	26
sea_dusk	43	0.417	0.299	28.2	0.240	42.5	0.084	79.7	3
sunset	138	3.099	2.760	10.9	2.714	12.4	2.705	12.7	125
winaw	10	2.307	0.917	60.2	0.846	63.3	0.681	70.5	5
yahoo	156	4.062	3.832	5.7	3.752	7.6	2.537	37.5	3
Average	—	**2.491**	**1.877**	**27.3**	**1.672**	**35.2**	**1.547**	**40.0**	—
aerial2	256	5.440	5.441	0.0	5.199	4.4	4.416	18.8	119
bike	220	4.528	4.154	8.3	4.159	8.1	4.159	8.1	220
bikeh	220	6.078	5.737	5.6	5.760	5.2	5.760	5.2	220
cafe	220	5.352	5.203	2.8	5.204	2.8	5.204	2.8	220
goldhill	220	4.834	4.841	-0.1	4.876	-0.9	4.876	-0.9	218
lena	215	4.313	4.320	-0.2	4.347	-0.8	4.346	-0.8	212
woman	220	5.455	5.250	3.8	5.259	3.6	5.259	3.6	220
Average	—	**5.087**	**4.905**	**3.6**	**4.844**	**4.8**	**4.635**	**8.9**	—
france	249	2.017	2.016	0.1	1.838	8.9	0.500	75.2	3
frog	102	6.254	5.297	15.3	5.158	17.5	4.679	25.2	55
library	221	5.692	5.596	1.7	5.569	2.2	5.506	3.3	193
mountain	110	6.698	5.429	18.9	5.393	19.5	5.356	20.0	97
washsat	35	4.428	2.233	49.6	2.235	49.5	2.235	49.5	35
Average	—	**4.911**	**3.983**	**18.9**	**3.898**	**20.6**	**3.451**	**29.8**	—

Table 3.3: Compression results, in bits per pixel (bpp), obtained with JPEG2000 when used after the preprocessing techniques presented in this chapter.

3.5 Final remarks

The standard JPEG-LS has been developed with more emphasis on general continuous-tone images. Its context modeling strategies are not expected to take the advantage of the sparse histograms existent in simple images. Moreover, for simple images, a transform-domain representation is not necessarily a useful compression tool, as it is for natural images. JPEG2000 is a good example. The strong correlation among the pixel values in continuous-tone images, which can facilitate data source modeling and adaptive coding techniques, is not present in simple images.

In spite of PNG has been designed to compress computer generated images and the experimental results confirm its supremacy comparing with the other standard methods, the algorithm used in the core of the standard does not explore the two dimensional information inherent to the images. This motivate the ideia that more can be done to compress this type of images, as we have shown in this chapter.

The compression of simple images can be addressed using two different approaches. One of this approaches relies on the use of standard lossless image coding techniques combined with an appropriate preprocessing technique. The second approach is the development of specialized coding techniques.

The histogram packing using a limited number of symbols proposed in this thesis is a preprocessing technique which is capable of producing compression improvements on images that have histograms that, although not strictly sparse, are quasi-sparse. In this case, the off-line packing approach is unsuitable. However, by reducing the size of the symbol-set used by the packing procedure, we attained globally better results, some of which, individually, are quite dramatic.

The histogram packing, being a preprocessing approach, does not imply any modification of the particular compression technique to which we want to associate it. This characteristic is of particular importance when the use of standards or well-established general-purpose compression techniques is required or desirable.

Chapter 4

Lossless compression of color-indexed images

In Chapter 3, we have studied a class of images that, due to its special characteristics, is not well tolerated by standard image coding methods, usually designed for natural images. In this chapter, we address the lossless compression of color-indexed images, another class of images that requires special attention.

Color-indexed images are represented by a matrix of indexes (the index image) and by a color-map or palette. The indexes in the matrix point to positions in the color-map and, therefore, establish the colors of the corresponding pixels. For a particular image, the mapping between index values and colors (typically, RGB triplets) is not unique — it can be arbitrarily permuted, under the condition that the corresponding index image is changed accordingly.

The compression of color-indexed images is a challenging task to most general purpose continuous-tone image coding techniques. Figure 4.1 shows the index image of a color-indexed image. This index image requires 257 037 bytes for JPEG-LS encoding (297 277 if lossless JPEG2000 is used), whereas the same image only needs 128 634 bytes when using the specific method presented in Section 4.2. This represents a gain of 50% when comparing with JPEG-LS and 57% when comparing with lossless JPEG2000.

In this chapter, we are interested in the development of specific techniques that can be used to efficiently encode this type of images images. Moreover, we are also interested in studying preprocessing methods that can be used together with standard image coding techniques with the aim of improving the lossless compression of color-indexed images.

Figure 4.1: An example of the index image of a color-indexed image.

One of the preprocessing methods studied in this chapter is histogram packing, which tries to explore the local histogram sparseness of color-indexed images. Another technique is the search of an optimal permutation of the color palette, such that the resulting image of indexes is more amenable for compression. The latter technique will be presented in Chapter 5 due to the extensive work done regarding these methods.

4.1 Specialized image compression techniques for color-indexed images

In this section, we address the compression of color-indexed images, presenting the most successful methods designed specially for this type of images. Among them we point out, for example, Piecewise-constant Image Model (PWC) [2, 3, 4] and the more recent new method that has been proposed by Chen *et al.* [10, 9]. PWC uses edge maps and other techniques to reduce the number of contexts. Chen's method uses a binary tree structure to represent the image colors and an arithmetic encoder with variable size context.

4.1.1 Piecewise-constant Image Model (PWC)

Paul J. Aubeck Jr. developed a technique for lossless compression of color-indexed images named Piecewise-constant Image Model (PWC) [2, 3, 4].

Whether synthetically produced or obtained from continuous tone natural images, palette images are characterized by the following properties:

- They tend to contain fewer colors than pixels;
- Pixels of the same color tend to be contiguous;
- The color of a pixel is statistically related to the colors of its neighbors.

The PWC captures these characteristics in a two pass model. In the first image pass, boundaries between constant color pieces (or domains) are established. In the second pass, the color of each domain is evaluated.

The PWC coding language is composed of four decisions that we present next:

- $D1$ Is the color of the current pixel identical to that of a specified rectilinear connect neighbor?
- $D2$ Is the color of the current pixel identical to that of a specified diagonally connect neighbor?
- $D3$ Is the color of the current pixel identical to a guessed value?
- $D4$ What is the color of the current pixel?

$D1$ decisions are used to establish the boundaries between constant color domains. The decisions $D2$ to $D4$ are used to establish the color information of each domain. The decisions $D1$ to $D3$ are binary and $D4$ is a composition of binary decisions.

In the following, we present the two main steps of the algorithm: the boundary coding and the color coding.

Boundary coding

The edge map used in this work represents boundary information via the introduction of imaginary edges between pixels. Each pixel is assigned one vertical and one horizontal edge in a separator lattice. In the edge map representation, binary decisions can be naturally used to determine whether or not a particular lattice site is full.

PWC populates its edge map boundary model in raster order. At each pixel location, X, the state of vertical separator site is determined first, followed by the horizontal site. Population decisions are made using D1 decisions from the PWC language. In Fig. 4.2, the two rectilinear

separator decisions are labeled $D1v$ and $D1h$, respectively. Due to connectivity constraints, $D1h$ can often be determined deterministically. For example, if none of the three causal edges touching the left end of separator site $D1h$ is full, then $D1h$ is deterministically empty. If only one of the causal edges is full, then D1h is deterministically full.

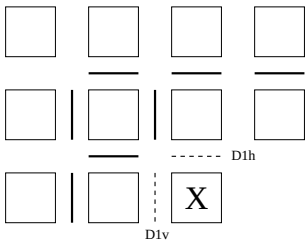


Figure 4.2: Edge context model in PWC.

For coding the edge map, PWC uses an arithmetic coder with a context formed by the neighboring edge segments. For each pixel X, $D1$ is posed against the vertical dotted edge location, as shown on Fig. 4.2, under the context determined by the eight solid edge segments on the presented in the same figure. The western edge together with the same surrounding edges form a context to pose $D1$ against the northern edge. The number of model parameters associated with this scheme is $256 + 512 = 768$, and the number of decisions is two per pixel. The number of model parameters and decisions of the model can be reduced (in these case the model parameters are reduced to 512) by taking advantage of boundary connectivity constraints.

Color coding

For color coding PWC uses the $D2$, $D3$ and $D4$ decisions. The color coding can be separated in three steps: first PWC tries to establish diagonal connectivity; if that fails, a color guessing process is attempted and finally, if color guessing fails, the color is established using predictive coding. In the following we explain these steps with more detail.

1. Diagonal Connectivity:

   $D2$ decisions are used to establish diagonal connectivity in PWC. Diagonal connectivity is only defined at lattice intersections where there is no rectilinear connectivity. The context model for $D2$ decisions uses both orientation and color information. In the Fig. 4.3 the two orientations are shown by the left and right representations and the

propagating color is labeled as C.

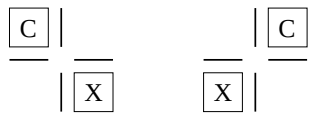


Figure 4.3: Diagonal context in PWC method.

2. Color Guessing:

   PWC language element $D3$, is designed to model the neighboring color relationships in an image while using a controlled number of model parameters. A guess is simply some color that has occurred previously in the coding process.

   The size of a guess model is proportional to D^S, where D is the palette depth of the image and s is the number of neighboring colors used in the model. To maintain a reasonably size of the model, the number of neighboring domain colors used to condition $D3$ must be limited. For 256 color images, it is usually only beneficial to include one neighboring color in the coding context, for example the previous domain color in the raster order.

   The number of guesses maintained simultaneously by the model is limited to some fixed number. When limiting guesses, a mechanism is needed to maintain only good guesses: guesses that are mostly correct. One way to achieve this is through guess competition within a guess pool. The competitive mechanism used by PWC is a least recently used (LRU) chain. In this application, a context is moved to the front of the LRU any time its associated guess is correct. When a new guess is added to the pool and the pool has reached its maximum size, the guess at the end of the LRU chain is sacrificed.

3. Guess failures:

   When diagonal connectivity and color guessing fail, PWC makes $D4$ decisions to introduce new colors in its model. $D4$ decisions are made using predictive coding. The predictor used is the same of JPEG-LS. Prediction residuals are coded using a method based on Golomb Coding.

Paul J. Aubeck Jr. has also developed a streaming approach for PWC [3]. Instead of making two complete image passes, the basic strategy of streaming PWC is to make two passes

through each image scanline. The first pass makes $D1$ decisions to establish rectilinear connectivity within the scanline and to the previous scanline. The second pass determines the color of each domain stripe by either propagating the color from the previous scanline or, failing that, by making decisions $D2$-$D4$.

Another development made in PWC was the introduction of a Skip-Innovation model. The basic idea is to skip over uniform areas entirely if possible and to partially skip them if not. This is made introducing another decision into the PWC coding model.

4.1.2 Chen's method

The coding method proposed by Chen *et al.* [10, 9] was inspired by the work of Orchard *et al.* in color quantization [49] and is based on the construction of a binary tree, where the root node represents all image colors and the leaf nodes represent each individual color. During encoding, the tree is traversed in a specific order, with the aim of minimizing the reconstruction error. For each node, the encoder sends the weighted average color of each of its two children nodes and the location of the pixels having a color change. The encoding of these pixel locations is performed by means of a context-based arithmetic encoder with variable size contexts.

We start by describing how the binary tree is constructed and then we explain how the required information is efficiently encoded.

Constructing the tree

We denote by M the number of different colors in the image, by $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_M\}$ the set of colors (RGB triplets, in our case), and by $\mathcal{S}_0 = \{1, 2, \ldots, M\}$ the set of color indexes used for identifying the colors, where index i represents color $\mathbf{c}_i$. The number of pixels associated with color $\mathbf{c}_i$ is given by p_i.

Each node, j, of the binary tree represents a certain subset of the colors of the image. We refer to this node using the corresponding set of color indexes, $\mathcal{S}_j$. Moreover, we associate to each node a representative color, $\mathbf{q}_j$, which is given by the weighted average color of the node, i.e.,

$$\mathbf{q}_j = \frac{\sum_{i \in \mathcal{S}_j} p_i \mathbf{c}_i}{\sum_{i \in \mathcal{S}_j} p_i}.$$

Whenever $|\mathcal{S}_j| > 1$, the node is split in two, in such way that the two new subsets of colors are

separated by a plane which is perpendicular to the principal direction of the data and passes through the average color value, $\mathbf{q}_j$. The principal direction of the data corresponds to the eigenvector associated with the largest eigenvalue of the covariance matrix of the weighted colors in $\mathcal{S}_j$, $\mathbf{C}_j$, where

$$\mathbf{C}_j = \sum_{i \in \mathcal{S}_j} p_i (\mathbf{c}_i \mathbf{c}_i^t - \mathbf{q}_j \mathbf{q}_j^t).$$

Traversing the tree

Although, in our explanation, we separated the construction of the tree from the part of traversing the tree and encoding the associated information, in the case of Chen's method these two operations can be done simultaneously. The first node to be encoded is the root node, for which the value of $\mathbf{q}_0$ is transmitted. Then, until the whole tree is traversed, the following procedure is iterated:

1. From the nodes available for processing, the node with the largest associated eigenvalue of $\mathbf{C}_j$ is chosen. Let us call it node m.

2. Encode the identification of this node, i.e., the value of m (the numbering of nodes is such that it can be reproduced by the decoder without additional side information) and the representative colors of its two children, $\mathbf{q}_m^l$ and $\mathbf{q}_m^r$, corresponding to subsets $\mathcal{S}_m^l$ and $\mathcal{S}_m^r$ ($\mathcal{S}_m = \mathcal{S}_m^l \cup \mathcal{S}_m^r$).

3. Encode the location of the pixels with color belonging to $\mathcal{S}_m^l$. Notice that since the location of the pixels with color belonging to $\mathcal{S}_m$ is known by the decoder, then we only need to encode, for each of those pixels, if its color belongs now to $\mathcal{S}_m^l$ or to $\mathcal{S}_m^r$.

Encoding pixel locations

For encoding the location of the pixels that have their colors modified due to a node splitting, Chen *et al.* proposed context-based arithmetic coding [10]. Contexts are constructed based on the template shown in Fig. 4.4, where the context pixels are numbered according to their distance to the encoding pixel. The context is constructed using a sequence of bits, $b_1 b_2 \ldots b_k$, where

$$b_i = \begin{cases} 0, & \text{if} \quad \|\mathbf{q}^i - \mathbf{q}_m^l\| \leq \|\mathbf{q}^i - \mathbf{q}_m^r\| \\ 1, & \text{otherwise} \end{cases},$$

and where $\mathbf{q}^i$ denotes the current color of the pixel in the reconstructed image corresponding to position i of the context template.

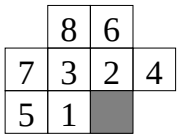


Figure 4.4: Context template used by Chen *et al.*

Chen *et al.* noticed that better results could be obtained if the size of the context template, i.e., the value of k, was progressively reduced during encoding [10]. Moreover, they proposed the relation

$$k(n) = 9 - \lfloor \log_2 n \rfloor \tag{4.1}$$

for doing this adaptation, where $n-1$ indicates the number of colors already encoded. Therefore, since the information used for calculating the contexts is binary, then the total number of contexts is given by

$$C(n) = 2^{k(n)}.$$

4.2 Proposed approach based on specialized methods

4.2.1 Modifications of Chen's method

In this section, we address the model for context adaptation proposed by Chen *et al.* and used by the encoding method proposed in [10]. Moreover, we propose a modification of this model that provides increased performance at virtually no additional computational cost [60].

One of the drawbacks of the context adaptation model proposed by Chen is that it does not take into account the size of the image being encoded. In fact, it seems reasonable to admit that larger images might allow larger contexts, because they provide more data for adapting the underlying probability models.

Having this motivation in mind, we investigated the appropriateness of a model that could make use of this information. First, we note that (4.1) can be rewritten as

$$k(n) = 9 - \lfloor \log_2 n \rfloor = \lceil 9 - \log_2 n \rceil = \left\lceil \log_2 \frac{512}{n} \right\rceil. \tag{4.2}$$

We intend to study the following generalization of (4.2),

$$k(n) = \lceil \alpha(N) - \log_2 n \rceil \,, \tag{4.3}$$

where N denotes the number of pixels of the image and

$$\alpha(N) = m \log_2 N + b, \tag{4.4}$$

i.e.,

$$k(n) = \left\lceil \log_2 \frac{N^m 2^b}{n} \right\rceil . \tag{4.5}$$

Moreover, and in order to accommodate wider variations of $k(n)$ that are due to the $\alpha(N)$ term, we introduced four additional pixel locations in the context template (see Fig. 4.5).

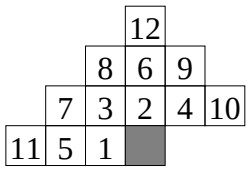


Figure 4.5: Context template used in this method.

The function $\alpha(N)$ was adjusted using values taken from the kodak set of 23 images (presented in Appendix A) quantized to 256, 128 and 64 colors, and for sizes of 768×512 (the original size), 384×256, 192×128 and 96×64. For each of these images, the best value of $\alpha(N)$ was found. Figure 4.6 shows the fitting of the model to the data obtained from the training set, resulting in $m = 0.671$ and $b = -0.859$.

4.2.2 Experimental results

In this section, we present experimental results obtained with some standard image compression methods, namely JPEG-LS, JPEG2000, JBIG and PNG. We also present results with the two most important specific compression methods developed for lossless compression of color-indexed images, namely PWC and Chen's method.

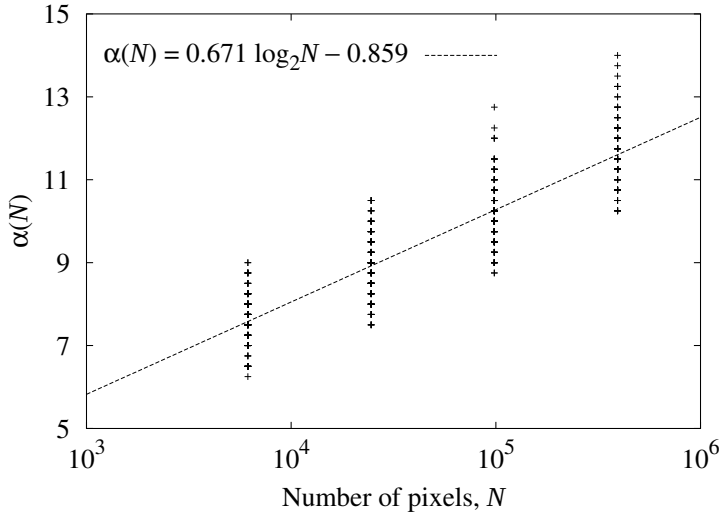


Figure 4.6: Plot showing the fitting of the model to the training data. Note that each '+' mark in the graphic generally indicates several superimposed data points.

According to the experimental results presented in Table 4.1, we reached the conclusion that globally JBIG performs best when compared with the other standards. Moreover, PNG performs better than JPEG-LS and JPEG2000. This confirms the performance of this standard when used in color-indexed images. Regarding the specialized methods, the modified Chen's method (mChen) is globally the best. Regarding individual results, Chen's method outperforms PWC in the images having a larger number of colors, while PWC is the best in the images with a small number of colors.

To evaluate the appropriateness of the context adaptation model presented in Section 4.2, we performed tests using the same collection of color-indexed images that have been used for the other methods. These are images from both synthetic and natural origins and of various sizes and number of colors.

Table 4.1 also presents the results obtained using the context adaptation model originally presented in Section 4.2. As can be observed, from the 30 test images, the proposed context adaptation model provides worse results for only three of them. The percentages represent compression gains obtained by the new model when compared with Chen's method. However, in overall terms, the impact of these negative results are negligible, since these images represent a very small fraction (about 0.3%) of the total number of bytes required to encode

Image	Colors	JLS	J2K	JBG	PNG	PWC	Chen	mChen	Gain
pc	6	0.812	1.082	0.321	0.501	0.230	0.313	0.273	12.8
books	7	2.043	2.236	1.562	1.514	1.151	1.138	1.130	0.7
music	8	1.156	1.693	0.665	1.049	0.470	0.538	0.538	0.1
winaw	10	0.551	0.859	0.483	0.493	0.297	0.354	0.352	0.5
party8	12	0.344	0.584	0.161	0.328	0.113	0.177	0.183	-3.0
netscape	32	2.027	2.823	1.915	2.508	1.481	1.460	1.460	0.0
sea_dusk	46	0.187	0.152	0.059	0.077	0.065	0.076	0.081	-6.5
benjerry	48	1.338	2.375	1.220	1.331	0.876	0.814	0.814	0.0
gate	84	3.457	4.118	3.392	3.298	2.124	2.013	2.009	0.2
descent	122	3.716	4.669	3.374	3.549	1.970	2.238	2.231	0.3
sunset	204	2.838	4.178	2.473	2.292	1.375	1.557	1.549	0.5
yahoo	229	2.499	3.470	2.139	2.005	1.338	1.699	1.701	-0.1
airplane	256	6.583	6.927	5.988	6.475	4.225	3.705	3.696	0.2
anemone	256	6.717	7.435	6.149	5.901	4.296	3.870	3.789	2.1
arial	256	7.000	7.278	6.649	6.759	5.397	5.259	5.204	1.0
baboon	256	7.482	7.593	7.288	7.444	6.244	5.436	5.425	0.2
bike3	256	5.366	6.233	4.922	5.211	3.304	3.202	3.106	3.0
boat	256	7.159	7.328	6.671	7.108	5.565	4.596	4.533	1.4
clegg	256	6.324	6.952	5.393	5.176	3.633	3.843	3.649	5.0
cwheel	256	4.121	4.863	3.000	3.028	1.610	2.049	1.803	12.0
fractal	256	7.020	7.228	6.536	6.824	5.489	4.981	4.912	1.4
frymire	256	4.078	5.171	3.045	3.002	1.438	2.095	1.854	11.4
ghouse	256	6.387	7.037	5.696	5.549	3.684	3.581	3.477	2.9
girl	256	7.035	7.250	6.590	6.786	4.803	4.095	4.039	1.4
house	256	6.185	6.391	5.996	5.989	4.489	4.017	4.010	0.2
lena	256	6.823	7.133	6.270	6.444	4.551	3.847	3.784	1.6
monarch	256	5.472	6.342	4.782	4.982	2.928	2.834	2.760	2.6
peppers	256	7.080	7.393	6.315	6.645	4.068	3.591	3.474	3.2
serrano	256	4.075	5.496	3.168	3.280	1.579	2.071	1.907	8.0
tulips	256	6.056	6.855	5.271	5.371	3.229	2.957	2.880	2.6
Average	—	**3.962**	**4.528**	**3.358**	**3.463**	**2.265**	**2.267**	**2.176**	**4.0**

Table 4.1: Compression results of color-quantized images using four standards and three specific methods. The "mChen" column refers to the method presented in this section. Percentage column shows the gain attained by "mChen" in relation to Chen's method.

the whole set of images. The average lossless compression gain attained was 4%.

4.3 Proposed approaches based on histogram packing

4.3.1 Region histogram packing

Generally, image data are not stationary. Therefore, a (global) histogram may not express correctly how intensities are used in different parts of the image. To illustrate this problem we refer to image "france" (496 rows × 672 columns, 249 different intensity values), which is displayed in Fig. 4.7, along with its (global) histogram.

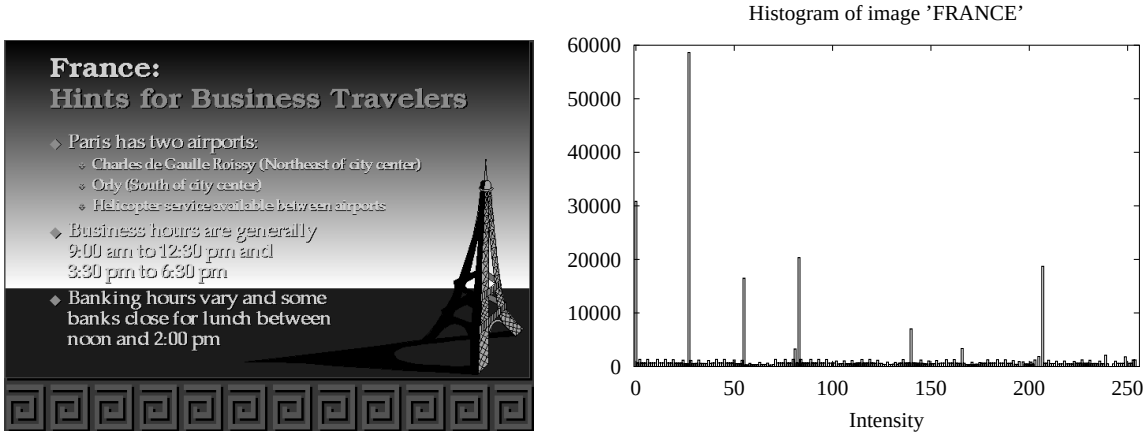


Figure 4.7: On the left, the "france" image. On the right, the histogram of the same image.

A simple analysis of this histogram would probably lead us to the conclusion that, due to its quasi-sparse appearance, the method described in the Chapter 3 would be the most appropriate, in order to obtain improvements concerning the compression of this image. In fact, it is not. Unfortunately, a simple inspection of the degree of sparseness that a given histogram exhibits is not enough to infer the impact of histogram packing in the lossless compression of the image.

Figure 4.8 shows several curves representing the number of intensities used in different parts of the image (the horizontal axis indicates relative position in relation to the end of the image, according to a raster-scanning approach), measured periodically (512 pixels of period) and using analysis windows of various lengths W_S: 1 024, 4 096, 16 384 and 65 536 pixels. As can be observed, using analysis windows of 1 024 or 4 096 pixels, the mean number of different intensities that is used simultaneously is significantly smaller (less than 10) than the number

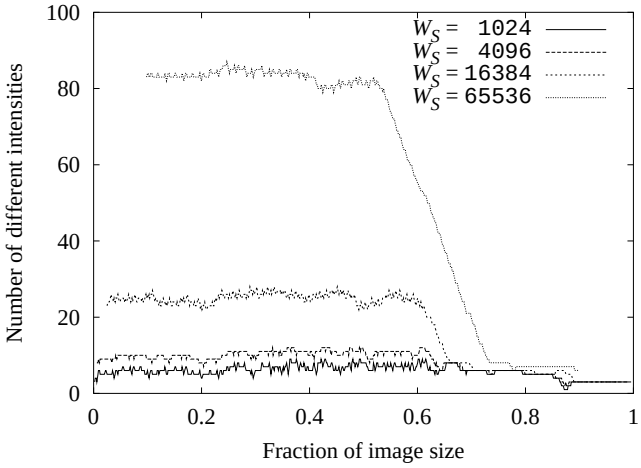


Figure 4.8: Analysis of local histograms of the "france" image.

given by a global histogram analysis (249). For larger window sizes this number grows rapidly.

Another class of images that have locally sparse histograms is color indexed images, obtained through color quantization of full color images. As we will study in Chapter 5, several techniques have been proposed to improve the lossless compression of this type of images. Basically, they rely on finding a suitable reordering of the color table in such a way that the corresponding image of indexes becomes more amenable to compression. As we can see in Fig. 4.9, the reordered version of a color quantized image have a locally sparse histogram. Although, as can be observed, the global histogram is dense. In this section, we present some methods to improve the compression of this type of images.

To explore the characteristics of images with locally sparse histograms, we developed a packing procedure which, basically, performs off-line histogram packing (see Section 3.3.1) on consecutive image regions of a predefined size and geometry. The choice of a particular partition has implications on the overall performance of the method, i.e., on the final compression rate. However, it is clearly impractical to search for the best partition among all the possibilities.

The major difference in relation to the (global) off-line histogram packing described in Section 3.3.1 is that, in this case, instead of one, we have to handle several mapping tables (one per image region). The overhead needed for storing the (uncompressed) list of intensities that occur inside each image block can be efficiently represented using 256 bits (32 bytes). Notice

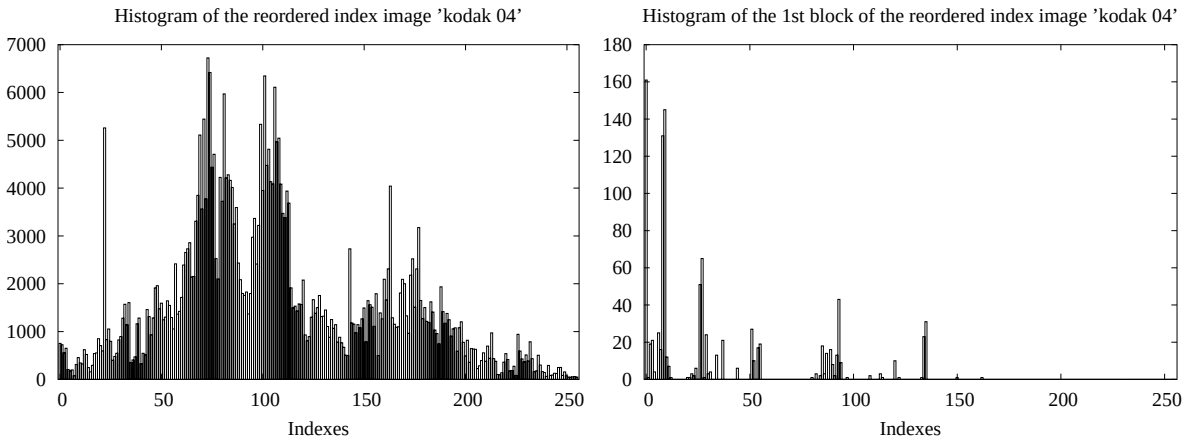


Figure 4.9: On the left, the histogram of the reordered "kodak 04" index image. On the right, the histogram of a 32×32 block of the same image.

that, to invert the process, we only need to know if a particular intensity exists in that block. This can be flagged using only one bit for each of the 256 possible intensities. This was the main motive to choose the off-line packing procedure to be used in the proposed method. In the on-line approach we need to store one byte for each intensity found in the image, which is a large amount of data when considered for each block.

This block-based histogram procedure consists on applying off-line histogram packing to each block of an image (see Fig. 4.10). In Fig. 4.11 we present experimental results showing the effect of changing the block size in the block-based histogram packing. The graphic presents global results (in bits per pixel) regarding the use of JPEG-LS after applying the region histogram packing algorithm in the 23 images of the kodak set. As we can see, the best results are obtained using a block size of 32×32 pixels. This was the size used in the work reported in [54, 55].

Figure 4.12 shows an example of the described algorithm. In this case, the overall compression gain (i.e., including the overhead for storing the list of intensities used by each block), was around 20%.

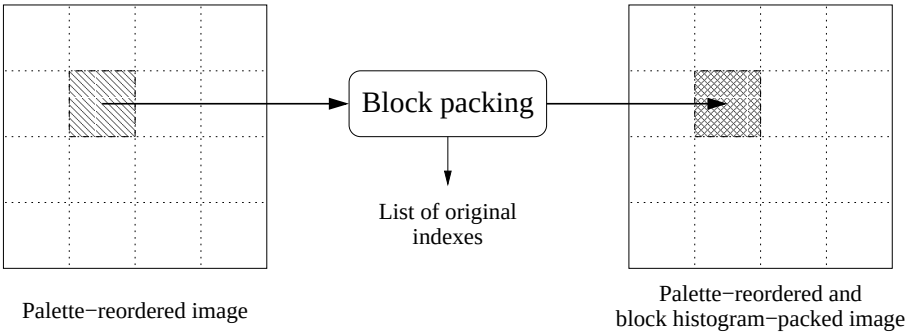


Figure 4.10: Schematic overview of the local histogram packing method.

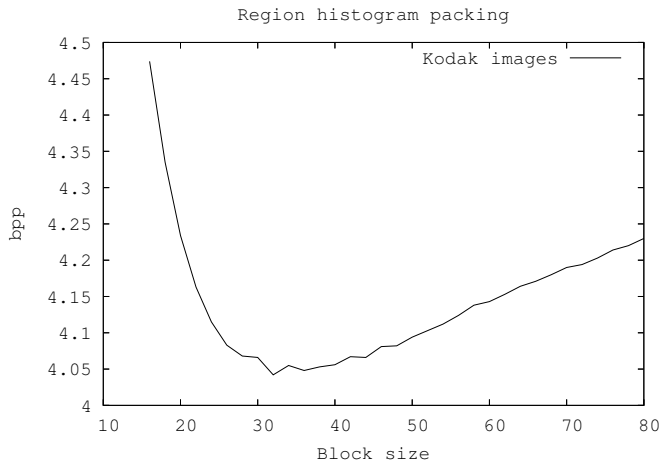


Figure 4.11: Analysis of the compression performance of region histogram packing varying the block size.

4.3.2 Region-adaptive histogram packing

One of the problems overlooked by strategies based on fixed-size block processing is the fact that characteristics of image data vary across the image. This means, for example, that a given block size may be appropriate for some region of the image, but might be too large or too small for some other region.

A way to overcome this limitation is by allowing the processing of regions of arbitrary shape and size. However, this is generally avoided in order to alleviate the overhead required by

(a) (b)

Figure 4.12: (a) Index image corresponding to image "07" of the kodak set after color quantization with dithering and after color table reordering based on luminance; (b) The same image after block-based histogram packing using square blocks of 32×32 pixels.

the representation of the regions, i.e., the number of bits needed for partition coding. A somewhat intermediate approach relaxes the arbitrary shape and size requirement into a less bit-expensive variable size and possibly variable shape approach. Here, "variable" means a reasonable, usually small, number of possibilities.

The method described in this section relies on a process that sequentially aggregates elementary blocks, of a given predefined size, in order to construct a larger region where histogram packing is performed. Figure 4.13 sketches the idea, showing how a new elementary block **B** might be aggregated to region $\mathbf{R}_k^n$ (n identifies the region being created, whereas k indicates how many elementary blocks already belong to that region).

The Block **B** is aggregated to the current region if the relation (4.6) holds. This relation measures the balance between an estimate of the number of bits required to encode the region if the block is aggregated, (b_{k+1}^n), and the estimate of the number of bits needed if not, $(b_k^n + b)$.

$$b_{k+1}^n - (b_k^n + b) < 0, \tag{4.6}$$

These estimates are calculated using the entropy of the residuals of a first order predictor, assuming that the pixels are raster scanned, and taking into account the overhead required

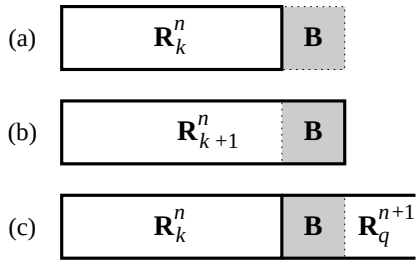


Figure 4.13: Construction of regions for histogram packing: (a) the elementary block **B** is the next candidate to integrate region $\mathbf{R}_k^n$; (b) if (4.6) is verified, them **B** is aggregated to region $\mathbf{R}_k^n$, which is relabeled to $\mathbf{R}_{k+1}^n$; (c) in this case (4.6) is not verified, triggering the beginning of a new region, $\mathbf{R}_q^{n+1}$ (at this stage, the value of q is unknown).

for storing the mapping table (M bits, for M-color images). Therefore, we may rewrite (4.6) as

$$(k+1)N^2H_{k+1}^n + o - (kN^2H_k^n + o + b) < 0$$

or

$$(k+1)N^2H_{k+1}^n - kN^2H_k^n - b < 0 \tag{4.7}$$

where H_{k+1}^n and H_k^n are, respectively, the entropies (calculated as mentioned above) of the region with $k+1$ and k elementary blocks, and o is the overhead required by the mapping table. We assume, without loss of generality, that the elementary blocks are squares of $N \times N$ pixels.

Term b in (4.6) and (4.7) denotes an estimate of the number of bits that block **B** would claim if, instead of being aggregated to region $\mathbf{R}_k^n$ (see Fig. 4.13b), it would be allocated to a new region, $\mathbf{R}_q^{n+1}$ (situation depicted in Fig. 4.13c). The main problem in obtaining this estimate is that, based only on past data, we are unable to know the size of the next region, i.e., the value of q.

In fact, the size of the next region is needed not only to estimate its entropy, but also to know how the overhead of the mapping table will be distributed among the pixels of the region, i.e., the value of o/q. Therefore, deciding when stop growing a given region $\mathbf{R}^n$ depends on the knowledge of the size of the next region, $\mathbf{R}^{n+1}$, which, by itself, depends on the size of region $\mathbf{R}^{n+2}$, and so on. At each decision step, this recursion goes until the end of the image, generally leading to a computational problem not solvable in practical time.

To alleviate the computational burden involved in the computation of b, we impose two limitations in the algorithm: (1) the maximum size of the region is limited to some value K, typically 16 elementary blocks; (2) the recursion does not proceed until the end of the image but, instead, until a point which is Q elementary blocks ahead of $\mathbf{B}$, typically 8. Notice that limitation (2) has the role of imposing an artificial end-of-image to the recursion process. On the other hand, restriction (1) has the additional goal of limiting the number of bits required to encode the size of the regions, which is $\lceil \log_2 K \rceil$.

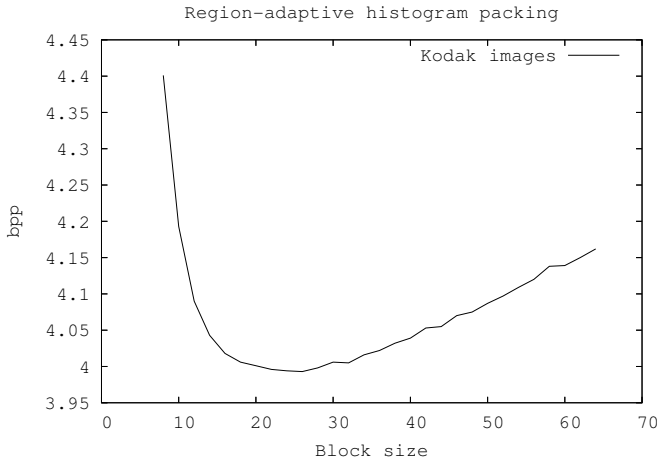


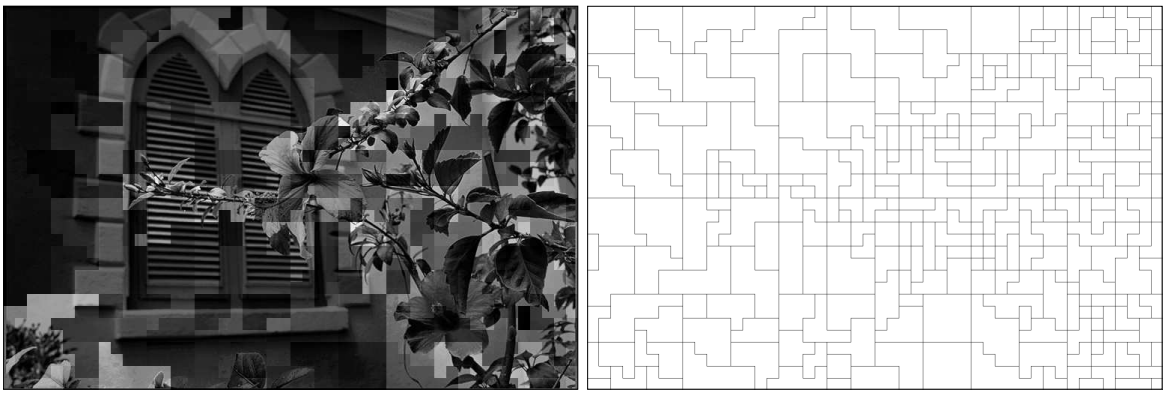
Figure 4.14: Analysis of the compression performance of region-adaptive histogram packing varying the block size.

In Fig. 4.14 we present experimental results showing the effect of changing the elementary block size in the region-adaptive histogram packing. The graphic presents global results (in bits per pixel) regarding the use of JPEG-LS after applying the region-adaptive histogram packing algorithm in the 23 images of the kodak set. These results were obtained using $K = 16$ and $Q = 8$. As we can see, the best results are obtained using an elementary block size in the range 16×16 to 28×28 pixels.

The Fig. 4.15 shows an example of the described algorithm.

4.3.3 Experimental results

The experimental results that we present in this section are based on a set of 23 true color images (768 columns × 512 rows) that we refer to as the "Kodak" images. Using version



(a) (b)

Figure 4.15: Example of the region-adaptive histogram packing applied to the Kodak image "07". (a) The luminance-reordered image of indexes after adaptive packing. (b) The regions created by the algorithm.

1.2.3 of the "`gimp`" program, each image was color-quantized based on an image-dependent palette of 256 colors with and without Floyd-Steinberg color dithering.

After color quantization the index images have been reordered using a luminance based method. For comparison, compression results obtained with the unordered index images, i.e., as obtained after color quantization, are also given. The compression results that are presented include, besides the size of the encoded index image, also the (uncompressed) size of the color table and, for the region based histogram packed approach, the overhead needed for storing the (uncompressed) list of intensities that occur inside each image block (256 bits for each block).

The results referring to region histogram packing were obtained using a partition of the image into 384 squared blocks of 32×32 pixels. Each of these blocks was (independently) remapped in order to eliminate the holes in the corresponding histogram. Therefore, for images with the above mentioned geometry, the overhead introduced by this operation is of 98 304 bits (0.25 bpp). The results for region-adaptive histogram packing were obtained with $K = 16$, $Q = 8$ and elementary block size of 16×16 pixels.

Table 4.2 shows compression results obtained for each of the images of the test set. Average values are also provided. From the analysis of Table 4.2 we can point out some important observations. First, using the region-based histogram packing, we can attain improvements of

Image	Without dithering					With dithering				
	W/o packing	Fixed packing	% gain	Adapt. packing	% gain	W/o packing	Fixed packing	% gain	Adapt. packing	% gain
01	5.973	5.286	11.5	5.267	11.8	6.151	5.518	10.3	5.498	10.6
02	5.762	5.383	6.5	5.371	6.8	5.991	5.684	5.1	5.663	5.4
03	3.633	2.428	33.3	2.344	35.6	4.779	3.430	28.3	3.349	30.0
04	4.771	3.726	21.9	3.672	23.1	5.279	4.219	20.1	4.173	21.0
05	5.667	4.612	18.6	4.579	19.2	6.000	5.101	15.0	5.078	15.4
06	4.956	4.268	13.9	4.226	14.7	5.230	4.619	11.7	4.579	12.4
07	4.202	3.255	22.6	3.207	23.7	4.951	3.971	19.8	3.911	21.0
08	5.566	4.869	12.5	4.827	13.3	5.699	4.993	12.4	4.954	13.0
09	4.510	3.650	19.1	3.577	20.7	4.846	4.089	15.6	4.014	17.2
10	4.652	3.805	18.2	3.776	18.8	4.999	4.315	13.7	4.286	14.3
11	5.119	4.119	19.6	4.068	20.6	5.439	4.484	17.5	4.454	18.1
12	4.327	3.298	23.8	3.217	25.7	4.849	3.965	18.3	3.921	19.2
13	6.452	5.764	10.6	5.744	10.9	6.535	5.987	8.4	5.967	8.7
14	5.347	4.219	21.1	4.159	22.2	5.616	4.586	18.4	4.532	19.3
15	4.058	3.241	20.2	3.175	21.8	4.575	3.824	16.4	3.757	17.9
16	4.669	3.913	16.2	3.875	17.0	5.092	4.417	13.3	4.388	13.8
17	4.550	3.951	13.1	3.915	14.0	4.805	4.360	9.3	4.309	10.3
18	5.899	4.918	16.6	4.874	17.4	6.026	5.195	13.8	5.149	14.5
19	5.080	4.105	19.2	4.045	20.4	5.332	4.467	16.2	4.420	17.1
20	3.371	3.195	5.2	3.110	7.7	3.557	3.456	2.8	3.391	4.7
21	5.257	4.333	17.6	4.291	18.4	5.494	4.693	14.6	4.642	15.5
22	5.223	4.361	16.5	4.313	17.4	5.534	4.708	14.9	4.660	15.8
23	3.599	2.624	27.2	2.557	29.0	4.865	3.634	25.3	3.564	26.8
Average	**4.898**	**4.058**	**17.1**	**4.008**	**18.1**	**5.289**	**4.510**	**14.7**	**4.467**	**15.5**

Table 4.2: Compression results, in bits per pixel, of the color-indexed "Kodak" images. For comparison, we provide compression results concerning three different approaches: (1) luminance-reordered images, without histogram packing; (2) luminance-reordered images with fixed block size histogram packing; (3) luminance-reordered images with region-adaptive histogram packing. The percentage gain refers to the compression without histogram packing. Color table sizes as well as the overheads generated by the histogram packing procedure are included in the compression results.

17.1% for images without dithering and 14.7% for images with dithering. Second, regarding the region-adaptive histogram packing, the compression performance has always improved. As expected, the gain varies from image to image, showing the ability of the region-adaptive histogram packing technique for exploiting zones of stationarity, where large blocks can be used, but without compromising zones where small blocks are required.

From the results, we observe that larger gains are generally attained for images having less

colors. This is also an expected behavior, because the gains provided by the proposed method are due to a well-balanced tradeoff between mapping table overheads and effectiveness of the histogram packing operation: on one hand, larger blocks imply less bits of overhead, but may impair the effectiveness of histogram packing; on the other hand, small blocks provide better compression results of the packed image, but imply more bits to represent the larger number of mapping tables that are required. Since the overhead in constant for fixed-size block-based histogram packing, then the impact of better using the overhead bits will be potentially larger for images with less colors.

4.4 Final remarks

In this chapter, we presented specific techniques that can be used to efficiently encode color-indexed images. We also presented preprocessing methods that can be used together with standard image coding techniques with the aim of improving the lossless compression of color-indexed images.

In Section 4.2 we presented a new context adaptation model for the color-indexed image coding method of Chen *et al.* This new model provided an average lossless compression gain of 4% on the used test set, when compared with Chen's method.

With the region-based histogram packing we try to explore the special properties of the histograms of color-indexed images. In spite of the global histogram being dense, locally these images use only a small number of intensities. With the region-based histogram packing procedure we can efficiently encode this type of images using the standard compression methods. The presented experimental results show the effectiveness of the region-based histogram packing procedure.

Chapter 5

Palette reordering methods for lossless compression of color-indexed images

Despite the existence of specialized approaches for coding color-indexed images, as we showed in the previous chapter, it is nevertheless important to ensure that general purpose coding techniques, such as JPEG-LS [24, 80] or lossless JPEG2000 [25, 73], perform appropriately when called to encode this class of images. In Chapter 4, we described a preprocessing method with that aim. In this chapter we address a class of preprocessing methods generally known as palette reordering.

For a particular image, the mapping between index values and colors is not unique — it can be arbitrarily permuted, under the condition that the corresponding index image is changed accordingly. However, for most continuous-tone image coding techniques these alternative representations are far from being equivalent. In Fig. 5.1 we illustrate this problem. On the left, an unordered index image obtained after color quantization of the 512×512 "Lena" image. On the right, the index image of the same color-indexed image, but after palette reordering using a luminance-based approach. The former index image requires 234 992 bytes for JPEG-LS encoding (241 116 if lossless JPEG2000 is used), whereas the latter index image only needs 169 158 bytes, i.e., 28% less (174 385 for lossless JPEG2000, i.e., 27.7% less).

Palette reordering is a class of preprocessing methods aiming at finding a permutation of the color palette such that the resulting image of indexes is more amenable for compression.

Figure 5.1: On the left, an (unordered) index image obtained after color quantization. On the right, the index image of the same color-indexed image, but after palette reordering using a luminance-based approach. The image on the right can be encoded with 28% less bytes than the image on the left.

These preprocessing techniques have the advantage of not requiring post-processing and of being costless in terms of side information. However, if the optimal configuration is sought, then the computational complexity involved can be high. In fact, the number of possible configurations for a table of M colors corresponds to the number of permutations of M objects, which equals $M!$. Clearly, exhaustive search is impractical for most of the interesting cases, which motivated several sub-optimal, lower complexity, proposals [59].

In this chapter, we provide a detailed study of palette reordering methods, investigating their ability to improve the lossless compression rates of some general purpose image coding techniques, namely, JPEG-LS, JPEG2000 and JBIG. Our study addresses two classes of images which we believe do cover a large percentage of the images usually represented as color-indexed images: color-indexed natural images, both with dithering and without dithering, and computer-generated images.

This chapter is organized as follows. In Section 5.1, we give a description of the palette reordering methods that fall under a class to which we refer as "color-based methods". These techniques are characterized by relying only on the information provided by the color palette. In Section 5.2, we describe a class of techniques that we call "index-based methods". The methods belonging to this class rely only on the statistical information conveyed by the index

image to perform the reordering operation, independently of its meaning in terms of color representation. In Section 5.4, we present the methods that we propose. In Section 5.5, we provide experimental results showing the effectiveness of our methods. Finally, in Section 5.6, we draw some conclusions.

5.1 Color-based methods

The techniques based on color information are characterized by relying only on the information provided by the color palette. The main idea behind color-based methods for palette reordering is that colors that are close in the color space, according to some measure, should have close indexes.

5.1.1 Luminance

Among the methods based only on the information provided by the color palette, reordering by luminance is the simplest (in fact, it is the simplest among all methods addressed in this chapter) [56]. This method was proposed by Zaccarin *et al.* [83] in the context of lossy compression. It relies on the assumption that, generally, a given pixel has neighbors of similar luminance and, therefore, colors with similar luminance should have similar indexes. Reordering is performed by sorting the colors according to its luminance, with the luminance computed according to

$$Y = 0.299R + 0.587G + 0.114B,$$

where R, G and B denote the intensities of the red, green and blue components, respectively. Note that Y is a weighted ℓ_1 norm in the RGB space, and only the distance to the origin (according to this norm) matters. This operation can be performed in $\mathcal{O}(M)$ operations if we use, for example, counting sort. Zaccarin claims in his work that the complexity of the reordering procedure is $\mathcal{O}(M \log M)$.

5.1.2 Po's method

Po *et al.* [64] proposed a reordering technique, to which they referred as the "closest pairs ordering", with the aim of assigning close indexes to colors that are close in three-dimensional color space. The proposed algorithm starts by assigning index zero to the color closest to the

origin of the color space and then proceeds by assigning index i to the color which is closest to the color corresponding to index $i-1$.

The idea of assigning close indexes to colors that are close in color space was also exploited by Hadenfeldt *et al.* [17] and by Spira *et al.* [74]. Basically, this strategy can be formulated as the problem of finding the shortest Hamiltonian path in a complete weighted graph $G(V, E, w)$, where each vertex in $V = \{v_1, v_2, \ldots, v_{|V|}\}$ ($|V| = M$) corresponds to a palette color, and $w(v_i, v_j), (v_i, v_j) \in E$ corresponds to the metric distance (usually Euclidean) measured between the colors associated with the graph vertices v_i and v_j. We seek to find an one-to-one mapping $\sigma : \{1, 2, \ldots, |V|\} \rightarrow V$ satisfying

$$\sigma = \arg\min_{\sigma_n} \sum_{i=1}^{|V|-1} w\big(\sigma_n(i), \sigma_n(i+1)\big). \tag{5.1}$$

The problem of finding σ in (5.1) is similar to the well-known traveling salesman problem (TSP), which is known to be NP-hard. Due to the intractability of the problem, in practice, exact solutions can only be found for small values of M, and, therefore, approximate solutions have to be sought. In fact, the method proposed by Po *et al.* [64] is known as the Nearest Neighbor Algorithm for the TSP, which is a simple technique, but, generally, generates poor solutions.

5.1.3 Hadenfeldt's method

Hadenfeldt *et al.* [17] proposed two strategies in order to approximately solve (5.1). One is similar to that proposed by Po *et al.* [64], with the variant of choosing the best starting point among all M possible starting colors. The other strategy relies on simulated annealing, a stochastic technique for combinatorial optimization, which aims at finding minimum (or maximum) values of a cost (or objective) function, usually non-linear and of many independent variables. This function is used to represent the appropriateness of the candidate solutions and it is often characterized by having a great number of local minima.

Generally, the complexity of the cost function renders techniques strictly based on steepest descent approaches useless, and simulated annealing provides a solution to the problem of getting trapped in local minima. This is possible because it not only allows the acceptance of solutions that reduce the cost function, but also of some that increase the cost function, thereby providing a way to escape local minima. The probability of accepting one of these solutions is controlled by a cooling procedure: at high temperatures they are frequently

accepted, at low temperatures only sporadically. New solutions are generated by producing (usually small) perturbations in the current solution.

It can be shown that, depending on certain conditions, where the cooling strategy plays a key role, the global minimum can be attained, although this may imply a large computing time. Nevertheless, even in the cases where the global optimum is missed, reasonable cooling strategies usually lead to good solutions.

5.1.4 Spira's method

Spira *et al.* [74] proposed a reindexing scheme that also falls under the formulation described in (5.1). In fact, they concluded that the colors should be ordered according to the distances between them in three-dimensional color space, based on the assumption that images contain objects and that objects are constructed from pixels with similar colors. As in [17], the proposed approach relies on finding an approximate solution to the TSP. However, in this case, they proposed to use the Farthest Insertion Algorithm, a $\mathcal{O}(M^3)$ heuristic for solving the TSP.

5.2 Index-based methods

The methods belonging to this class rely only on the statistical information conveyed by the index image to perform the reordering operation, independently of its meaning in terms of color representation.

The main idea behind index-based methods for palette reordering is that colors that occur frequently close to each other should have close indexes. Therefore, based on this principle, the assignment of the indexes is usually guided by some function, $C(i, j)$, measuring the number of occurrences corresponding to pixels with index i that are spatially adjacent to pixels with index j, according to some predefined neighborhood.

5.2.1 Waldemar's method

In the context of lossy compression of palettized images, Waldemar *et al.* [78] proposed a palette sorting algorithm, to which they referred as "sorting by color correlation". The algorithm that they proposed starts by creating a list of indexes (the sorting list) sorted by their number of occurrences in the image. Then, it expands the list (creating a second row)

by attaching, to each of the indexes in the first row, the index most frequently found in its neighborhood (they used a 8-connected neighborhood for that purpose). Therefore, for each index i in the first row of the sorting list, a pair is formed with the index satisfying $\arg\max_j C(i,j)$.

They proposed the construction of a weighted graph using a weighting scheme based on the number of times the most frequently occurring color occurs in a pair and also based on the assumption that a pair of colors is more likely to be visually close the more frequent is the corresponding index in the first row of the sorting list. The new ordering of the colors is obtained by traversing the graph.

Due to the complexity of this algorithm, Waldemar *et al.* proposed an alternative color correlation sorting algorithm, based on the creation of color groups, and ensuring that within each group colors from a pair taken from the sorting list lie close to each other. These groups are then sorted in order to create the new palette [78].

5.2.2 Memon's method

Memon *et al.* formulated the problem of palette reordering within the framework of linear predictive coding [38]. In that context, the objective is to minimize the zero-order entropy of the prediction residuals. They noticed that, for image data, the prediction residuals are often well modeled by a Laplacian distribution and that, in this case, minimizing the absolute sum of the prediction residuals leads to the minimization of the zero-order entropy of those residuals. For the case of a first-order prediction scheme, the absolute sum of the prediction residuals reduces to

$$\sum_{i=1}^{M}\sum_{j=1}^{M} C(i,j)|i-j|, \tag{5.2}$$

where, in this case, $C(i,j)$ denotes the number of times index i is used as the predicted value for a pixel whose color is indexed by j.

The problem of finding a palette reordering that minimizes (5.2) can be formulated as the optimization version of the linear ordering problem (also known as the minimum linear arrangement), whose decision version is known to be NP-complete [38]. In fact, if we consider a complete non-directed weighted graph $G(V,E,w)$, where each vertex in $V=\{v_1,v_2,\ldots,v_{|V|}\}$ ($|V|=M$) corresponds to a palette color, and $w(v_i,v_j)=C(i,j)+C(j,i), (v_i,v_j)\in E$ corresponds to the weight associated to the edge defined between vertices v_i and v_j, then the goal

is to find an one-to-one mapping (permutation) $\sigma : V \to \{1, 2, \ldots, |V|\}$ satisfying

$$\sigma = \arg\min_{\sigma_n} \sum_{(v_i,v_j)\in E} w(v_i, v_j)|\sigma_n(v_i) - \sigma_n(v_j)|. \tag{5.3}$$

With the aim of seeking approximate solutions for (5.3), Memon *et al.* proposed two heuristics: one based on simulated annealing, the other, faster to compute, based on a technique called "pairwise merge".

Essentially, the pairwise merge heuristic is based on repeatedly merging ordered sets of colors until obtaining a single (reordered) set. Initially, each color (graph vertex) is assigned to a different set. Then, each iteration consists of two steps. First, the two sets A and B maximizing

$$\sum_{v_i\in A}\sum_{v_j\in B} w(v_i, v_j)$$

among all possible pairs of ordered sets are chosen. Then, a number of merging combinations of the sets A and B are tested (ideally, all possible combinations), and the one minimizing

$$\sum_{i=1}^{k}\sum_{j=1}^{k} w(u_i, u_j)|i - j|,$$

is selected. Here $u_1, u_2, \ldots, u_k$ is the combined ordered set under evaluation. To alleviate the computational burden involved in selecting the best way of merging the two ordered sets, Memon *et al.* proposed to use a reduced number of configurations [38]. If $a_1, a_2, \ldots, a_r$ and $b_1, b_2, \ldots, b_s$ are the two ordered sets under evaluation, and if $r, s > 1$, then the following configurations are considered:

$$\begin{aligned} &a_1, a_2, \ldots, a_r, b_1, b_2, \ldots, b_s \\ &a_r, \ldots, a_2, a_1, b_1, b_2, \ldots, b_s \\ &b_1, b_2, \ldots, b_s, a_1, a_2, \ldots, a_r \\ &b_1, b_2, \ldots, b_s, a_r, \ldots, a_2, a_1 \end{aligned} \tag{5.4}$$

Alternatively, if one of the sets has size one, then the following configurations are tried (without loss of generality, we consider $r = 1$):

$$\begin{aligned} &a_1, b_1, b_2, \ldots, b_s \\ &b_1, a_1, b_2, \ldots, b_s \\ &b_1, b_2, a_1, \ldots, b_s \\ &\qquad\vdots \\ &b_1, b_2, \ldots, b_s, a_1 \end{aligned} \tag{5.5}$$

5.2.3 Zeng's method

The palette re-indexing method proposed by Zeng *et al.* [84] is based on an one-step look-ahead greedy approach, which aims at increasing the lossless compression efficiency of color-indexed images.

The algorithm starts by finding the index that is most frequently located adjacent to other (different) indexes, and the index that is most frequently found adjacent to it. This pair of indexes is the starting base for an ordered set, $\mathcal{S}$, that will be constructed, one index at a time, during the operation of the re-indexing algorithm. We denote by v_i the indexes already assigned to the ordered set (i indicates the position of the index in the ordered set and, therefore, its distance to the left end side of the set) and by u those still unassigned. Therefore, just before starting the iterations, $\mathcal{S} = \{v_1, v_2\}$, where v_1 and v_2 are the two indexes mentioned above. New indexes can only be attached to the left or to the right extremity of the ordered set.

The algorithm then proceeds as follows. For each iteration, compute u_L and u_R according to:

$$u_L = \arg\max_{u \notin \mathcal{S}} D_L(u), \tag{5.6}$$

where

$$D_L(u) = \sum_{v_i \in \mathcal{S}} \alpha_i \, C(u, v_i), \tag{5.7}$$

and

$$u_R = \arg\max_{u \notin \mathcal{S}} D_R(u), \tag{5.8}$$

where

$$D_R(u) = \sum_{v_i \in \mathcal{S}} \alpha_{|\mathcal{S}|-i+1} \, C(u, v_i). \tag{5.9}$$

The function $C(i, j) = C(j, i)$ denotes the number of occurrences (measured in the initial index image) corresponding to pixels with index i that are spatially adjacent to pixels with index j. The α_k are weights controlling the impact of the $C(u, v_k)$ on $D_L(u)$ and $D_R(u)$ and, originally [84], were proposed to be given by

$$\alpha_k = \log_2 \left(1 + \frac{1}{k}\right).$$

The new set is given by $\{u_L, v_1, \ldots, v_{|\mathcal{S}|}\}$, if $D_L(u_L) > D_R(u_R)$, or by $\{v_1, \ldots, v_{|\mathcal{S}|}, u_R\}$, otherwise. This iterative process continues until assigning all indexes. Finally, the re-indexed

image is constructed by applying the mapping $v_i \mapsto (i-1)$ to all image pixels, and changing the color-map accordingly.

5.2.4 Fojtik's method

Fojtik *et al.* [14] developed a palette reordering technique specially designed for coding methods working on a bitplane basis. In fact, their aim was to find a method to permute the indexes of the palette such that the resulting binary images representing the individual bitplanes would be as smooth as possible. Most general purpose lossless image coding methods, such as JPEG-LS or lossless JPEG2000, do not rely on intensity domain bitplane coding, and, therefore, do not satisfy the main assumption used in the design of this reordering technique. Nevertheless, this preprocessing technique can be used with success with bitplane based compression methods, such as, JBIG.

The method proposed by Fojtik performs a statistical analysis of the adjacency of the intensity values in the neighborhood of each pixel and minimizes the first order entropy in the current bitplane, rearranging bitplanes below it. The algorithm starts from the most significant bitplane and proceeds to the bitplanes below it. The lower bitplanes can be modified, but the already rearranged bitplanes above it should remain intact. The aim is to permute indexes of the palette in such manner that the resulting binary planes contains a small number of large regions.

5.2.5 Battiato's method

Battiato *et al.* [6] formulated the reindexing problem as that of finding the Hamiltonian path of maximum weight in a non-directed weighted graph $G = (V, E, w)$, where the vertices $V = \{v_1, v_2, \ldots, v_{|V|}\}$ ($|V| = M$) represent the palette colors, and $w(v_i, v_j) = C(i,j) + C(j,i), (v_i, v_j) \in E$ denotes the weight associated to the edge connecting vertices v_i and v_j. Function $C(i,j)$ has the same meaning as that used by Memon *et al.*, i.e., denotes the number of times index i is used as the predicted value for a pixel with index j. Then, the problem is to find a permutation $\sigma : \{1, 2, \ldots, |V|\} \rightarrow V$ satisfying

$$\sigma = \arg\max_{\sigma_n} \sum_{i=1}^{|V|-1} w\big(\sigma_n(i), \sigma_n(i+1)\big), \tag{5.10}$$

which is similar of finding a solution to the TSP. This formulation resembles that of (5.1), the only difference being the meaning assigned to the $w(v_i, v_j)$ weights: in (5.1) they represent

distances in color space, whereas in (5.10) they denote the number of occurrences of pairs of pixels having a certain color pair.

In order to find a solution to this problem, Battiato *et al.* [6] proposed a greedy strategy based on sequentially selecting the best edge still not processed (i.e., the one with the largest weight). The complexity of the proposed algorithm is bounded by the edge sorting operation and is, therefore, of order $\mathcal{O}(M^2 \log M)$.

5.3 Experimental comparison of the methods

In this section, we present experimental compression results, based on three sets of images, concerning the efficiency of six palette reordering methods described in previous sections: Luminance, Hadenfeldt and Spira, regarding color-based methods, and Memon, Zeng and Battiato, regarding index-based methods; for comparison, results using the unordered index images are also presented.

The first set of images presented in Tables 5.1 and 5.2 is composed of 18 computer-generated images having different numbers of colors and geometries. Color quantization was applied only to the images originally with a number of colors greater than 256 ("clegg", "cwheel", "frymire", "house" and "serrano"). The set "natural1", also known as the "kodak" set, is composed of 23 768×512 natural images. The set "natural2" contains the following twelve popular natural images: "airplane", "lena", "peppers", "girl", "baboon" and "boat" (512×512), "monarch" and "tulips" (768×512), "anemone" (722×471), "arial" (735×493), "bike3" (781×919) and "house" (256×256).

Color quantization was applied to the images in sets "natural1" and "natural2", both with and without Floyd-Steinberg color dithering, originating images with 256, 128 and 64 colors. Version 1.2.3 of the *"Gimp"* program was used in order to generate the color-quantized images.

Software implementations provided by the respective authors have been used for Spira, Zeng and Battiato's methods. All other methods have been implemented by us. In what follows, when we mention Memon's method we refer to the pairwise merge technique, and when we mention Hadenfeldt's method we refer to the greedy approach, not to the one based on simulated annealing.

Table 5.1 shows JPEG-LS and Table 5.2 shows JPEG2000 lossless compression results, in bits per pixel, of the reordered index images (the size of the corresponding color-maps are included

in the presented values). The rows in Tables 5.1 and 5.2 corresponding to the "natural1" and "natural2" sets display average compression results calculated over the particular instance of the image set (i.e., combination of number of colors and dithering approach). The rows labeled "Average" provide overall results.

The results presented in Tables 5.1 and 5.2 show that Memon's method provided the highest average compression improvement amongst all tested methods, both for JPEG-LS and for lossless JPEG2000, and for all classes of images addressed in the experiments. The second best technique was Zeng's method for the first set and "natural1" set without dithering. The luminance based method was the second best for the remaining sets. Interesting to note is the fact that luminance based reordering, the fastest amongst all methods, provided competitive results for natural images, especially for those with more colors and quantized using dithering.

Images	colors	Unordered	Color-based methods			Index-based methods		
			Luminance	Hadenfeldt	Spira	Memon	Zeng	Battiato
clegg	256	6.333	5.330	6.579	5.925	**5.220**	5.863	6.075
cwheel	256	4.134	4.087	3.686	3.254	**2.724**	3.058	3.374
fractal	256	7.036	6.086	6.229	6.347	**5.763**	6.193	7.234
frymire	256	4.083	3.759	3.966	3.680	**3.259**	3.619	3.946
ghouse	256	6.399	4.890	5.846	5.204	**4.229**	4.841	5.418
serrano	256	4.087	3.583	3.805	3.461	**3.001**	3.393	3.779
yahoo	229	2.701	2.787	2.481	2.759	**1.743**	1.798	2.008
sunset	204	2.854	2.328	2.704	3.553	2.407	2.570	2.647
descent	122	3.762	3.485	3.994	4.769	**2.817**	2.943	3.531
gate	84	3.490	2.930	2.903	3.770	**2.548**	2.587	3.116
benjerry	48	1.379	1.423	1.349	1.954	**1.133**	1.154	1.186
sea_dusk	46	0.194	0.191	0.192	0.257	0.197	**0.189**	**0.189**
netscape	32	2.040	1.918	1.945	1.915	**1.745**	1.791	1.907
party8	12	0.346	0.367	0.360	0.355	0.321	**0.318**	0.319
winaw	10	0.552	0.546	0.539	0.644	**0.450**	**0.450**	0.464
music	8	1.171	1.143	1.287	1.296	**1.051**	1.060	1.071
books	7	2.046	1.884	1.592	2.212	**1.453**	1.469	**1.453**
pc	6	0.812	0.768	0.796	0.849	0.748	**0.743**	**0.743**
Average	–	2.854	2.526	2.728	2.579	**2.243**	2.452	2.650
natural1	256	6.764	5.289	6.200	5.532	**4.870**	5.595	6.232
with	128	5.621	4.398	5.076	4.494	**4.112**	4.537	5.146
dithering	64	4.531	3.555	3.910	3.639	**3.341**	3.590	3.915
Average	–	5.639	4.414	5.062	4.555	**4.108**	4.574	5.098
natural1	256	6.077	4.897	5.528	5.070	**4.203**	4.907	5.530
without	128	4.838	3.926	4.292	3.960	**3.441**	3.844	4.287
dithering	64	3.661	3.002	3.076	2.953	**2.641**	2.804	3.144
Average	–	4.859	3.942	4.299	3.994	**3.428**	3.852	4.320
natural2	256	6.923	5.363	6.319	5.643	**5.063**	6.006	5.455
with	128	5.844	4.495	5.285	4.755	**4.205**	4.766	5.314
dithering	64	4.712	3.643	4.105	3.883	**3.417**	3.712	4.127
Average	–	5.826	4.500	5.236	4.760	**4.228**	4.828	4.965
natural2	256	6.433	5.008	5.804	5.163	**4.607**	5.491	5.983
without	128	5.318	4.117	4.629	4.378	**3.771**	4.339	4.729
dithering	64	3.967	3.069	3.300	3.263	**2.793**	3.060	3.281
Average	–	5.239	4.065	4.578	4.268	**3.724**	4.297	4.664

Table 5.1: Lossless compression results, in bits per pixel, obtained with JPEG-LS applied to the index images after using the palette reordering methods presented in Sections 5.1 and 5.2.

Images	colors	Unordered	Color-based methods			Index-based methods		
			Luminance	Hadenfeldt	Spira	Memon	Zeng	Battiato
clegg	256	6.961	6.184	7.341	6.751	**5.951**	6.497	7.166
cwheel	256	4.876	4.686	4.108	3.861	**3.068**	3.398	3.661
fractal	256	7.244	6.229	6.429	6.555	**5.974**	6.406	7.522
frymire	256	5.176	5.377	5.214	4.843	**4.093**	4.369	4.721
ghouse	256	7.050	5.488	6.519	5.835	**4.760**	5.402	6.083
serrano	256	5.508	5.148	5.183	4.935	**4.038**	4.416	4.901
yahoo	229	3.673	4.378	4.042	3.951	**2.220**	2.229	2.500
sunset	204	4.193	3.328	4.090	5.282	3.424	3.579	3.859
descent	122	4.715	4.536	4.945	5.486	3.499	3.504	4.510
gate	84	4.152	3.460	3.684	4.399	**2.987**	3.088	3.538
benjerry	48	2.416	2.825	2.407	3.118	**1.745**	1.761	2.008
sea_dusk	46	0.159	0.314	0.225	0.335	0.255	0.129	**0.120**
netscape	32	2.836	2.653	2.776	2.598	**2.382**	2.420	2.611
party8	12	0.586	0.970	0.746	0.833	**0.411**	0.414	0.432
winaw	10	0.860	0.919	0.852	1.373	**0.611**	**0.611**	0.634
music	8	1.708	2.142	1.794	1.696	**1.374**	1.388	1.428
books	7	2.239	2.169	1.763	2.433	**1.598**	1.601	1.604
pc	6	1.082	0.953	1.055	1.158	0.748	**0.736**	**0.736**
Average	–	3.438	3.141	3.330	3.178	**2.611**	2.794	3.078
natural1	256	7.034	5.537	6.545	5.800	**5.148**	5.835	6.536
with	128	5.902	4.650	5.416	4.759	**4.354**	4.754	5.403
dithering	64	4.845	3.815	4.245	3.909	**3.590**	3.807	4.189
Average	–	5.927	4.667	5.402	4.823	**4.364**	4.799	5.376
natural1	256	6.574	5.274	6.033	5.492	**4.575**	5.281	5.938
without	128	5.358	4.308	4.802	4.366	**3.792**	4.203	4.717
dithering	64	4.208	3.399	3.580	3.355	**2.981**	3.146	3.539
Average	–	5.380	4.327	4.805	4.404	**3.783**	4.210	4.731
natural2	256	7.234	5.651	6.741	6.054	**5.408**	6.289	6.867
with	128	6.164	4.754	5.662	5.015	**4.482**	5.049	5.606
dithering	64	5.037	3.908	4.534	4.107	**3.694**	3.999	4.447
Average	–	6.145	4.771	5.646	5.059	**4.528**	5.112	5.640
natural2	256	6.959	5.429	6.364	5.680	**5.062**	5.904	6.532
without	128	5.903	4.552	5.208	4.833	**4.203**	4.769	5.201
dithering	64	4.596	3.523	3.950	3.717	**3.213**	3.493	3.814
Average	–	5.819	4.501	5.174	4.743	**4.159**	4.722	5.182

Table 5.2: Lossless compression results, in bits per pixel, obtained with JPEG2000 applied to the index images after using the palette reordering methods presented in Sections 5.1 and 5.2.

5.4 New proposed approaches

5.4.1 Modified Zeng's method

In this section, we address Zeng's approach for color re-indexing of palette-based images [84]. We provide a theoretical analysis of the technique, leading to a set of parameters that differs from the one originally suggested in [84]. In our approach [57], we prove that the optimal weights used in the process of choosing the next symbol are given by $\alpha_k = 1$ if an exponential distribution is assumed for the differences between the neighboring pixels (this is a widely accepted model for the prediction residuals of continuous-tone images [80]). We also show that the process for determining the correct side of the list for attaching new symbols requires different weights (β_k)that decrease linearly with the value of k.

In the analysis that follows, we consider the entropy of the absolute differences between the neighboring pixels as an indicator of the degree of compressibility of an image. This seems to be a reasonable assumption, specially if prediction-based compression methods are intended to be used after the re-indexing.

According to the greedy strategy of Zeng's algorithm, the next index, $\bar{u}$, that should integrate $\mathcal{S}$ is the one that implies the largest increase in code length if its choice is postponed to the next iteration. It is well-known that, for a memoryless source, the number of bits required to represent a given symbol s is given by $-\log_2 P(s)$, where $P(s)$ denotes the probability of occurrence of s.

We start by defining the estimated code length implied by placing index u on the left side of $\mathcal{S}$

$$l_L(u) = -\sum_{v_i \in \mathcal{S}} C(u, v_i)\ \log_2 P(i), \tag{5.11}$$

by placing it one position farther away

$$l_L^+(u) = -\sum_{v_i \in \mathcal{S}} C(u, v_i)\ \log_2 P(i+1), \tag{5.12}$$

by placing it on the right side of $\mathcal{S}$

$$l_R(u) = -\sum_{v_i \in \mathcal{S}} C(u, v_i)\ \log_2 P(|\mathcal{S}| - i + 1), \tag{5.13}$$

and by placing it one position farther away from the right side

$$l_R^+(u) = -\sum_{v_i \in \mathcal{S}} C(u, v_i)\ \log_2 P(|\mathcal{S}| - i + 2). \tag{5.14}$$

The new index, $\bar{u}$, should satisfy

$$\bar{u} = \arg\max_{u \notin \mathcal{S}} \Delta l(u), \tag{5.15}$$

with

$$\Delta l(u) = \begin{cases} l_L^+(u) - l_L(u), & \text{if} \quad l_R(u) - l_L(u) > 0 \\ l_R^+(u) - l_R(u), & \text{otherwise.} \end{cases} \tag{5.16}$$

In words, for each candidate index, u, its best position (left or right) is chosen, i.e., the one that minimizes the code length. Then, among all those indexes, we pick the one producing the largest increase in code length if its choice is postponed to the next iteration.

Now, we can write

$$l_L^+(u) - l_L(u) = \sum_{v_i \in \mathcal{S}} \log_2 \frac{P(i)}{P(i+1)}\, C(u, v_i) = \sum_{v_i \in \mathcal{S}} \alpha_i\, C(u, v_i), \tag{5.17}$$

if the best position for index u is the left end side, or

$$l_R^+(u) - l_R(u) = \sum_{v_i \in \mathcal{S}} \log_2 \frac{P(|\mathcal{S}| - i + 1)}{P(|\mathcal{S}| - i + 2)}\, C(u, v_i) = \sum_{v_i \in \mathcal{S}} \alpha_{|\mathcal{S}|-i+1}\, C(u, v_i), \tag{5.18}$$

if the best position is the right end side, where

$$\alpha_k = \log_2 \frac{P(k)}{P(k+1)}, \tag{5.19}$$

and where $P(k)$ denotes the probability of occurrence of a difference of k units between two neighboring pixels.

Moreover, we can also write

$$l_R(u) - l_L(u) = \sum_{v_i \in \mathcal{S}} \big(\log_2 P(i) - \log_2 P(|\mathcal{S}| - i + 1) \big)\, C(u, v_i) = \sum_{v_i \in \mathcal{S}} \beta_i\, C(u, v_i), \tag{5.20}$$

where

$$\beta_k = \log_2 \frac{P(k)}{P(|\mathcal{S}| - k + 1)}. \tag{5.21}$$

For exponentially distributed residuals, i.e., considering

$$P(k) = A\theta^k, \quad 0 < \theta < 1, \quad 0 \leq k < M \tag{5.22}$$

Eq. (5.19) reduces to

$$\alpha_k = \log_2 \frac{A\theta^k}{A\theta^{(k+1)}} = \big(k - (k+1)\big) \log_2 \theta = -\log_2 \theta, \tag{5.23}$$

and (5.21) to

$$\beta_k = \log_2 \frac{A\theta^k}{A\theta^{(|\mathcal{S}|-k+1)}} = \big(k - (|\mathcal{S}| - k + 1)\big) \log_2 \theta = (2k - |\mathcal{S}| - 1) \log_2 \theta. \tag{5.24}$$

Finally, we note that the $\log_2 \theta$ term can be eliminated, since it is a constant factor. Moreover, β_k decreases linearly with k (notice that $\log_2 \theta < 0$).

5.4.2 Generalized Zeng's method

In Section 5.4.1, a modification of Zeng's algorithm was presented, relying on an exponential model for the distribution of first order prediction residuals, and on the assumption that the entropy of the absolute differences between neighboring pixels is a good indicator of the degree of compressibility of an image. In what follows, we extend the work reported in the Section 5.4.1 in order to accommodate an exponential power distribution model of the prediction residuals [58].

For exponentially power distributed residuals, i.e., considering

$$P(k) = A\theta^{k^\gamma}, \quad 0 < \theta < 1, \quad 0 \leq k < M, \quad \gamma > 0 \tag{5.25}$$

Eq. (5.19) reduces to

$$\alpha_k = \log_2 \frac{A\theta^{k^\gamma}}{A\theta^{(k+1)^\gamma}} = \big(k^\gamma - (k+1)^\gamma\big) \log_2 \theta, \tag{5.26}$$

and (5.21) to

$$\beta_k = \log_2 \frac{A\theta^{k^\gamma}}{A\theta^{(|\mathcal{S}|-k+1)^\gamma}} = \big(k^\gamma - (|\mathcal{S}| - k + 1)^\gamma\big) \log_2 \theta. \tag{5.27}$$

An noted previously, the $\log_2 \theta$ term can be eliminated, since it is a constant factor, although bearing in mind that it is always negative.

In the Section 5.5 we present experimental results showing the compression gain that can be obtained if an exponential power model is used in comparison to the exponential model (i.e., for $\gamma = 1.0$).

5.4.3 Block based palette reordering

Generally, images are not stationary. Therefore a global reordered palette could not be optimum for an entire image. Block based palette reordering consists on applying a palette reordering algorithm to each block of a given image partition (see Fig. 5.2). The basic idea is

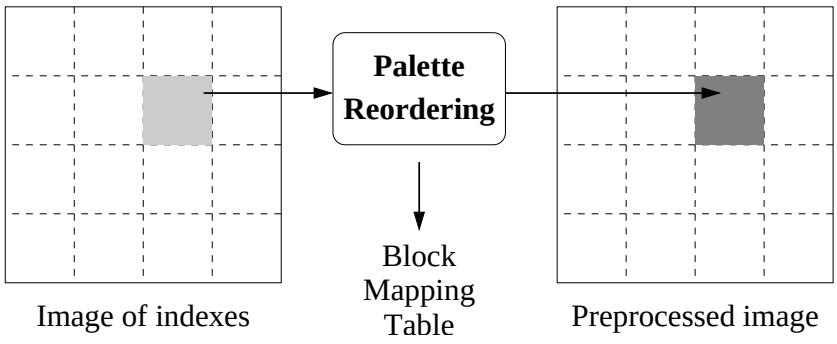


Figure 5.2: Schematic overview of block-based palette reordering method.

to address the palette reordering problem from a local point of view, i.e., on image regions, instead of addressing it globally.

In this case, we have to store several mapping tables, one for each image region. In the work reported in [44], fixed-shape and fixed-size regions (block of 128×128 pixels) have been used.

Figure 5.3: An example of a reordered image using the block-based approach.

Figure 5.3 shows the 512×512 "Lena" image reordered using the block-based approach. This reordered index image requires 139 586 bytes for JPEG-LS encoding (149 584 if lossless JPEG2000 is used), including the size needed to store all the mapping tables necessary for

reconstruction. The same image unordered requires 234 992 bytes for JPEG-LS encoding (241 116 if lossless JPEG2000 is used), whereas the global reordered version needs 169 158 bytes (174 385 for lossless JPEG2000). Detailed experimental results are presented in Section 5.5.

5.4.4 Bitplane based palette reordering

The bitplane based reordering algorithm that we propose in this section has been inspired by the work of Fojtik *et al.* [14]. The aim of this method is to permute indexes in such way that the resulting binary images of each bitplane contain less and larger regions, improving the compression [45]. To better understand the idea, we present an example in Fig. 5.4, where the most significant bitplane (MSB) of the image "peppers" is shown before and after the reordering procedure. As can be observed, the preprocessed bitplane contains less and larger regions, a characteristic highly desired by most image coders.

Figure 5.4: The most significant bitplane of the image "peppers" before and after the reordering procedure.

The method starts by performing the analysis of the adjacency of the intensity values in the neighborhood of each pixel. This is given by the function $w(i, j)$, which is responsible for conveying the information of how frequently the pairs of neighboring pixels occur in the image.

The image is processed in a bitplane basis, starting from the most significant bitplane and

proceeding to bitplanes below it. The operation of the algorithm is such that modifications performed in lower bitplanes do not affect already processed (upper) bitplanes.

Let us assume that there is some initial division of indexes into two groups, G_1 and G_2. For each group, we choose the index that has more relations with the other group than with its own group. These two indexes are swapped between groups. The process is repeated until the swapping procedure does not reduce the number of relations between the two groups. Since the algorithm proceeds from the most to the least significant bitplanes, the task now is to split G_1 and G_2 into four groups in the bitplane below (G_{11}, G_{12}, G_{21} and G_{22}). The swapping approach is similar to that described above, but, to maintain the already processed bitplanes intact, the swapping procedure is only allowed between some groups: Elements of G_{11} can be swapped with elements of G_{12} and elements of G_{21} with elements of G_{22}. Figure 5.5 provides a schematic view of the procedure.

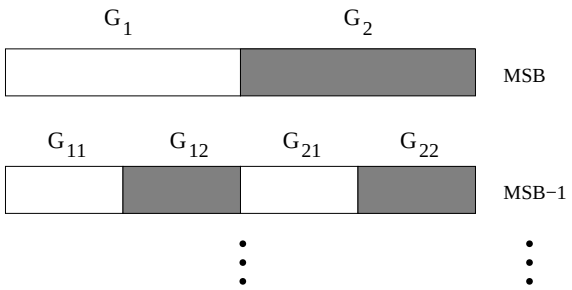


Figure 5.5: The division of the indexes begins in the most significant bitplane (MSB). Swapping is only allowed between groups G_1 and G_2 in the MSB and between groups $G_{11} - G_{12}$ and $G_{21} - G_{22}$ in the bitplane bellow it. In the remaining bitplanes, the procedure is identical.

5.5 Experimental results

5.5.1 Modified Zeng's and generalized Zeng's methods

Table 5.3 shows the practical appropriateness of the theoretical analysis presented in Section 5.4.1 and 5.4.2. We collected a number of color-indexed images of various sizes and number of colors, both from synthetic and natural origins . These images have been re-indexed using Zeng's original method [84], the modification proposed in Section 5.4.1 ("mZeng" column) and the generalized approach described in Section 5.4.2 ("gZeng" column), being af-

terward compressed using a JPEG-LS codec. Table 5.3 presents the compression results that have been obtained (in terms of bits per pixel), which include the size of the color-maps.

Image	Colors	Zeng	mZeng		gZeng		
		bpp	bpp	Gain	bpp	γ	Gain
pc	6	0.743	0.745	-0.2	0.745	1.0	0.0
books	7	1.469	1.469	0.0	1.453	1.3	1.1
music	8	1.060	1.051	0.9	1.051	1.0	0.0
winaw	10	0.450	0.450	0.0	0.450	1.0	0.0
party8	12	0.318	0.318	-0.1	0.316	2.3	0.6
netscape	32	1.791	1.752	2.2	1.741	0.8	0.6
sea_dusk	46	0.189	0.189	0.0	0.189	1.0	0.0
benjerry	48	1.154	1.137	1.5	1.137	1.1	0.1
gate	84	2.587	2.566	0.8	2.559	0.9	0.3
descent	122	2.943	2.854	3.0	2.843	0.7	0.4
sunset	204	2.570	2.307	10.2	2.294	1.2	0.6
yahoo	229	1.798	1.789	0.5	1.767	0.5	1.3
airplane	256	5.056	4.445	12.1	4.362	1.8	1.9
anemone	256	5.806	4.966	14.5	4.801	1.5	3.3
arial	256	6.871	6.183	10.0	6.183	1.0	0.0
baboon	256	7.097	6.496	8.5	6.442	1.5	0.8
bike3	256	4.915	4.154	15.5	4.154	1.0	0.0
boat	256	6.048	5.823	3.7	5.623	2.0	3.4
clegg	256	5.863	5.456	6.9	5.388	1.3	1.2
cwheel	256	3.058	2.878	5.9	2.854	0.8	0.8
fractal	256	6.193	5.828	5.9	5.653	1.6	3.0
frymire	256	3.619	3.376	6.7	3.329	0.5	1.4
ghouse	256	4.841	4.541	6.2	4.494	1.3	1.0
girl	256	5.727	5.255	8.2	5.196	1.1	1.1
house	256	5.180	4.854	6.3	4.815	1.4	0.8
lena	256	5.710	5.049	11.6	4.871	1.9	3.5
monarch	256	4.325	3.917	9.4	3.917	1.0	0.0
peppers	256	5.544	5.019	9.5	4.732	1.6	5.7
serrano	256	3.393	3.273	3.5	3.173	0.7	3.1
tulips	256	4.724	4.032	14.6	4.032	1.0	0.0
Average	—	**3.402**	**3.122**	**8.2**	**3.077**	—	**1.4**

Table 5.3: Compression results, in bits per pixel (bpp), using the JPEG-LS codec, of Zeng's method, the modified version ("mZeng") and the generalized case ("gZeng"). For "mZeng" columns, "Gain" indicates the percentage of compression of the modified version in relation to Zeng's approach. For "gZeng" columns, "Gain" indicates the percentage of compression of the generalized case in relation to the modified version.

Regarding the modifications made in Zeng's method, analyzing the results presented in Table 5.3, we conclude that they are effective. In fact, for all but two of the test images ("pc" and "party8"), compression improvements have occurred (seven of them with a gain of 10% or more). Moreover, the reduction in compression rate verified in the "pc" and "party8" images is negligible (-0.2% and -0.1%, respectively). We note that, although appropriate for modeling the prediction residuals of continuous-tone images, the exponential distribution might not be appropriate for some images. In fact, the differences in the compression gains among the images and, particularly, the reduction in compression rate that was verified in images "pc" and "party8", might be directly related to the level of matching attained between the model and the particular image.

The experimental results presented also suggest that, in general, the larger the number of colors in the image, the larger seems to be the compression improvement. This behavior may indicate that the re-indexing technique is less affected by sub-optimal parameters when the number of colors to re-index is small.

In relation to the generalized approach, from the results presented in Table 5.3, we observe that, in fact, the exponential model seems to be a reasonable choice for most of the images. 17 of the 30 test images included in Table 5.3, had compression improvements of less than one percent. However, for some of the images the lossless compression gain was over 3%.

Moreover, although currently it is not very practical to use the exponential power model for palette reordering (due to the need of searching for the best γ for each image), it may be so if a low complexity way of guessing it from the image is found.

5.5.2 Block based palette reordering

Table 5.4 presents experimental results regarding the use of block-based palette reordering. The compression results that are presented include, besides the size of the encoded index image, the (uncompressed) size of the color table (6 144 bits) and, for the block-based palette reordering approach, the overhead needed for storing the (uncompressed) permutation of the colors for each block (2 048 bits per block, a total of 49 152 bits for 128×128 blocks). The lossless compression of the index images was performed using JPEG-LS after palette reordering using Memon's and mZeng techniques. "Block Reordering" and "Block Reordering (Best)" indicate, respectively, the results of applying the proposed approach on square blocks of 128×128 pixels and of using square blocks with size indicated in column "N".

Image	Block Reordering				Block Reordering (Best)					
	mZeng		Memon		mZeng			Memon		
	bpp	%	bpp	%	bpp	%	N	bpp	%	N
01	5.388	2.1	5.170	3.6	5.353	2.8	224	5.153	4.0	144
02	4.786	1.9	4.666	1.5	4.786	2.0	128	4.634	2.2	160
03	2.283	12.7	2.166	10.9	2.283	12.8	128	2.149	11.7	224
04	3.641	12.8	3.419	10.2	3.632	13.0	96	3.410	10.5	112
05	4.699	5.1	4.530	6.8	4.694	5.3	112	4.530	6.8	128
06	4.341	9.9	4.220	9.4	4.296	10.9	160	4.161	10.7	160
07	3.323	4.8	3.103	4.9	3.270	6.4	192	3.072	5.9	176
08	5.163	8.7	4.979	5.6	5.101	9.9	176	4.955	6.1	176
09	3.719	5.0	3.606	3.7	3.645	6.9	176	3.571	4.7	320
10	3.969	15.6	3.764	14.1	3.954	15.9	112	3.764	14.2	128
11	4.110	8.2	3.960	5.8	4.083	8.8	144	3.960	5.8	128
12	3.269	15.0	3.144	13.3	3.262	15.2	112	3.144	13.4	128
13	5.648	10.5	5.569	7.3	5.648	10.6	128	5.569	7.4	128
14	4.206	8.2	4.032	3.4	4.206	8.3	128	4.013	3.9	176
15	3.128	11.5	2.983	8.6	3.107	12.2	144	2.960	9.4	176
16	4.021	11.9	3.847	7.2	4.021	11.9	128	3.847	7.3	128
17	4.261	8.7	3.989	3.3	4.250	9.0	112	3.976	3.7	240
18	4.764	8.8	4.522	10.3	4.764	8.7	128	4.522	10.4	128
19	4.164	13.5	3.974	6.9	4.164	13.6	128	3.973	7.0	144
20	3.178	7.4	3.063	2.1	3.167	7.8	192	3.011	3.8	192
21	4.289	8.4	4.092	6.9	4.269	8.9	112	4.092	7.0	128
22	4.393	8.3	4.254	9.9	4.365	8.9	112	4.254	10.0	128
23	2.565	14.2	2.442	11.1	2.560	14.4	144	2.424	11.8	144
Average	**4.057**	**9.1**	**3.891**	**7.2**	**4.038**	**9.5**	—	**3.876**	**7.5**	—

Table 5.4: Compression results, in bits per pixel (bpp), of the color-indexed "Kodak" images regarding the use of block-based palette reordering. Percentages were obtained comparing these results with the corresponding results presented in Table 5.1.

As can be observed in Table 5.4 , an increase in compression efficiency was obtained for all images, in comparison with the globally palette reordered approach. The columns under the label "Block Reordered (Best)" in Table 5.4 contain the compression values corresponding to the size of the blocks that provide the highest compression for each image.

5.5.3 Bitplane based palette reordering

In the Table 5.5 we present experimental results to show the performance of the bitplane based palette reordering method described in Section 5.4.4. Compression results are given

for JPEG-LS, lossless JPEG2000 and JBIG. The compression results obtained with JBIG after palette reordering with Memon's and the modified Zeng's methods include a Gray code conversion. This is the default mode provided by the JBIG codec, and is an effective procedure for encoding natural images on a bitplane basis [1]. Images reordered according to the bitplane approach are encoded without this code conversion (`-b` flag of the JBIG codec).

Image	JPEG-LS			JPEG2000			JBIG		
	Memon	mZeng	Bitp	Memon	mZeng	Bitp	Memon	mZeng	Bitp
pc	0.748	0.745	0.812	0.748	0.749	1.082	0.264	0.261	0.331
books	1.453	1.469	1.911	1.598	1.601	1.862	1.246	1.220	1.298
music	1.051	1.051	1.171	1.374	1.374	1.708	0.665	0.665	0.770
winaw	0.450	0.450	0.536	0.611	0.611	0.812	0.370	0.370	0.407
party8	0.321	0.318	0.336	0.411	0.413	0.571	0.162	0.167	0.180
netscape	1.745	1.752	2.113	2.382	2.390	2.900	1.752	1.792	1.873
sea_dusk	0.197	0.189	0.196	0.255	0.129	0.201	0.076	0.065	0.076
benjerry	1.133	1.137	1.323	1.745	1.769	2.207	1.108	1.169	1.086
gate	2.548	2.566	3.337	2.987	3.037	3.955	2.565	2.492	2.692
descent	2.817	2.854	3.603	3.499	3.424	4.530	2.705	2.561	2.595
sunset	2.407	2.307	2.735	3.424	3.263	4.048	2.060	2.058	2.425
yahoo	1.743	1.789	2.195	2.220	2.253	2.890	1.755	1.710	1.852
airplane	4.202	4.445	5.866	4.506	4.748	6.302	4.383	4.507	4.950
anemone	4.678	4.966	6.223	5.340	5.678	6.999	4.612	4.807	4.663
arial	5.890	6.183	6.905	6.144	6.512	7.273	5.705	5.857	5.580
baboon	6.272	6.496	7.029	6.498	6.697	7.221	6.287	6.442	6.037
bike3	4.034	4.154	5.072	4.720	4.847	5.888	3.724	3.894	4.047
boat	5.236	5.823	6.627	5.538	6.077	6.851	5.341	5.717	5.296
clegg	5.220	5.456	6.072	5.951	6.126	6.730	4.585	4.750	4.843
cwheel	2.724	2.878	4.284	3.068	3.243	5.025	2.099	2.129	2.805
fractal	5.763	5.828	6.612	5.974	6.038	6.872	5.681	5.700	5.425
frymire	3.259	3.376	3.916	4.093	4.217	4.994	2.403	2.523	2.649
ghouse	4.229	4.541	6.201	4.760	5.091	6.867	3.960	4.113	4.632
girl	5.107	5.255	6.392	5.357	5.545	6.644	4.916	5.019	4.878
house	4.467	4.854	5.390	4.820	5.139	5.658	4.640	4.917	4.793
lena	4.519	5.049	6.305	4.834	5.435	6.672	4.478	4.806	4.450
monarch	3.715	3.917	5.086	4.344	4.520	6.002	3.404	3.551	4.122
peppers	4.740	5.019	5.889	5.081	5.398	6.265	4.443	4.542	4.296
serrano	3.001	3.273	3.830	4.038	4.323	5.257	2.308	2.468	2.890
tulips	3.703	4.032	5.352	4.252	4.656	6.175	3.589	3.767	4.303
Average	2.982	3.122	3.744	3.377	3.520	4.321	2.600	2.691	2.803

Table 5.5: Lossless compression results, in bits per pixel, obtained with JPEG-LS, JBIG and JPEG2000, after reordering the palette of the color-indexed images using Memon, mZeng and the bit-plane based palette reordering methods.

The results presented in Table 5.5 show that Memon's method provides the highest average compression improvement for the three coding standards addressed in this table. The bitplane

reordering, the fastest amongst the three palette reordering methods, attains better results when used with JBIG and, for some images, provides the best compression results. This shows the effectiveness of the method when used with a bitplane based encoder.

To perform the reordering of all images presented in Table 5.5, our implementation of Memon's method required 140.6 seconds, whereas mZeng required 3.8 seconds. The bitplane reordering needed only 1.2 seconds. These results were obtained on a Pentium IV Mobile 2.0 GHz with 512 MB of memory. We only took into account the time spent on parts of the code directly involved with the reordering operation. These time measures allow us to conclude that bitplane reordering is, in fact, the fastest method and, on the opposite side, we have Memon's method.

5.6 Final remarks

Palette reordering is a very effective approach for improving the compression of color-indexed images. In this chapter, we described several approaches that have been proposed during the last decade and presented new approaches.

Regarding the modified version of Zeng's method, we conclude that the modifications proposed are indeed effective. In fact for almost all of the test images, compression improvements have occurred. We note that although appropriate for modeling the prediction residuals of continuous-tone images, the exponential distribution might not be appropriate for some digital images. This assumption is comproved by the experimental results obtained using the generalized approach.

A detailed comparison between Memon's method and the modified version of Zeng's method was presented in [62]. The main conclusion of this study was that Memon's method can be viewed as an extension of the modified Zeng's method, the latter being included into the former. In our opinion, this is a quite interesting finding, due to the fact that they have been developed independently and formulated according to different approaches (graph theory versus information theory).

The experimental results regarding the block-based approach show that the lossless compression of color-indexed images can be improved if we explore the local information of the images.

Finally, we conclude that Memon's method is the best one in terms of average compression performance, although it is also the slowest one. The bitplane based approach and modified Zeng's method provide competitive compression results and are faster than Memon's method. The bitplane based method gives better results when used in association with compression methods that are bitplane based, as is the case of JBIG. In fact, we conclude that this standard method is the best to compress color-indexed images.

Chapter 6

Lossless compression of microarray images

The DNA microarray technology has become an important tool in the study of gene function, regulation, and interaction across large numbers of genes, and even entire genomes. It allows the analysis of thousands of genes in a single experience [40, 19].

The raw data of a microarray experiment consist of a pair of 16 bits per pixel grayscale images (an example is presented in Fig. 6.1). These images are analyzed using a variety of software tools which extract relevant information, such as the intensity of the spots and the background level. This information is then used to evaluate the expression level of individual genes [40, 19]. Depending on the size of the array and the resolution of the scanner, these images may require several tens of megabytes in order to be stored or transmitted.

Microarrays have been the focus of significant research. Most of this effort has been directed to the analysis of the data resulting from such experiments, whereas problems such as the efficient representation of the microarray images have received relatively less attention. However, giving the massive amount of data currently produced and the need of long-term storage and efficient transmission, the development of efficient compression methods is an important challenge.

The common approach towards the compression of microarray images has been based on image analysis for spot finding (griding followed by segmentation) with the aim of separating the microarray image data into channels based on pixel similarities [32, 28, 21, 29, 20, 13, 12, 36, 86]. Once separated, the channels are compressed individually, together with the

segmentation information. Although appealing, image dependent compression methods may potentially run into trouble if the assumptions in which they are based change in the future. One such assumption might be, for example, the rectangular organization of the spots. In fact, although initially this was the organization used for spot placement in the microarrays, non-rectangular packings have also been used recently.

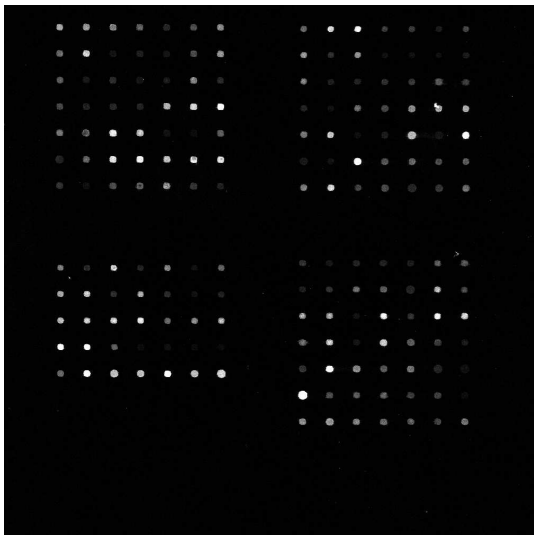


Figure 6.1: An example of a microarray image.

In this chapter, we present a comprehensive study of the image compression standards applied to the compression of microarray images, addressing the effect of noise in their compression performance. We also propose a new algorithm for lossless compression of microarray images. The method is based on arithmetic coding driven by a 3D finite-context model [47, 46]. Basically, the image is compressed on a bitplane basis, going from the most significant bitplane to the least significant bitplane. Encoding is stopped if an average of more than one bit per pixel is obtained after encoding a given bitplane. In this case, the remainder bitplanes are sent uncompressed. The finite-context model used by the arithmetic encoder uses (causal) pixels from the bitplane under compression and also pixels from the bitplanes already encoded.

6.1 Specialized image compression techniques for microarray images

In this section, we present the most important methods for compression of microarray images, namely, the works of Jörnsten *et al.* [29], Hua *et al.* [21], Faramarzpour *et al.* [13], Lonardi *et al.* [36] and Zhang *et al.* [86].

Although all the methods presented in this section address the microarray compression problem using different approaches, some of the processing steps are common, i.e., they try to decompose the problem in very similar ways as depicted in Fig. 6.2. All the methods start by segmenting the microarray images. Through segmentation, each spot is isolated in *regions of interest* (ROI's), with the spot and some surrounding background. Some methods go even further, separating the spot area from the background. However, the segmentation algorithm used in each method is different.

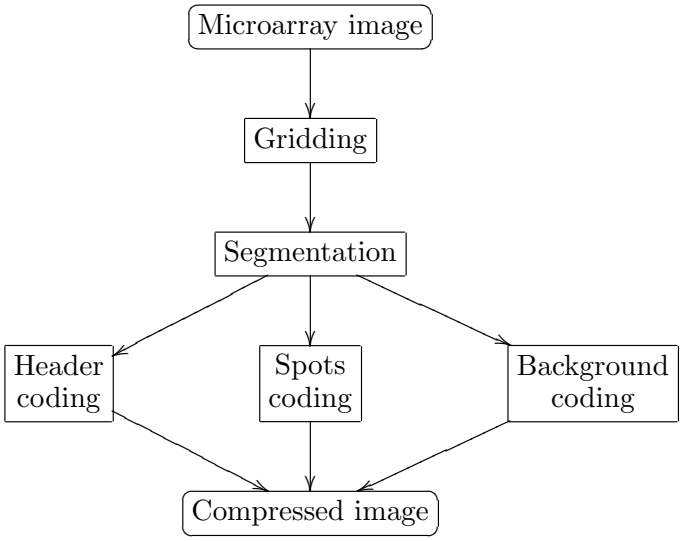


Figure 6.2: The common processing steps of the compression methods presented in this section.

Through segmentation, it is possible to code separately the spots and their background. This is especially notorious in the work of Jörsten *et al.* [29, 28, 33, 32, 30, 31], Hua *et al.* [21, 20] and Lonardi *et al.* [36]. In the work of Faramarzpour *et al.* [12, 13], this is not so perceptible. This is due to the fact that the image is segmented in ROIs, but there is not an

explicit separation between the spot area and background until the point where the sequence is entropy coded.

Almost all available methods also have a lossy compression version. These methods remove what is considered to be noise, or redundant. Although this step sounds obvious, the question is "What should be considered noise or redundant?" Note that background, unlikely what could be thought at first glance, is very important to noise estimation. Through noise estimation, the bias due to noise can be estimated and removed in the calculation of the gene expression level of each spot.

6.1.1 Segmented LOCO (SLOCO)

To our knowledge, Jörnsten *et al.* presented in [30] the first work in compression of microarray images. Although presenting good results for lossless compression, the algorithm presented in [30], which also could have a lossy mode, encodes a *conservative* version of the ROIs, i.e., it encodes the spots and only a small portion of the background around them. This was considered to be enough to the calculations of the gene expression level, and, therefore, the terms lossless and lossy are only applicable to what was considered a ROI.

Later, a more structured method was presented in [29, 28, 33, 32, 31], named segmented LOCO (SLOCO). Note that in [32] a slightly different algorithm is presented, but under the same base.

The method begins by estimating the spots grid, in order to determine their approximate center. After that, a seeded region growing algorithm is used for initial spot segmentation, and then a two-component gaussian mixture is fitted to further refine the boundaries. The image is then divided in ROIs with spot and background surrounding it.

Intrinsic to the method is also a step for genetic information extraction. The idea is to provide a progressive transmission scheme. Initially, only the header information is supplied, which allows to choose the interesting image subsets, and then only the intended subsets are transferred. The genetic information extracted is the differential expression level, given as the log ratio of the spot intensities, spot variances and shapes and product intensities.

The information from segmentation and genetic extraction is transmitted first. The spot and background means are encoded using adaptive Lempel-Ziv, and the segmentation map using a chain code.

The image encoding is done using a modified LOCO-I (LOw COmplexity LOssless COm-

pression of Images) algorithm. LOCO-I is the algorithm behind JPEG-LS. The algorithm differs in three main aspects. First, the spots and background are encoded separately. Second, a UQ-adjust (Uniform Quantizer) quantizer is used instead of the UQ quantizer. Third, varying pixel error (δ) is allowed.

The algorithm proceeds as follows. Primarily, the genetic information is used to determine the SNR to each spot. Then, to allow subset reconstruction, the image is divided into sub-blocks and the modified LOCO is separately applied to the spots and background for each image sub-block, with a locally determined error bound according to the determined SNR. Hence, a residual image remains from the previous compression step. The final lossy-to-lossless behavior is determined by this image coding.

After encoding a lossy version, the residual image can be progressively bitplane coded, from the most significant to the least significant bitplane, refining the lossy version of the microarray image (and the residual image). Controlling how much is coded from the residual image defines the quality of the microarray image lossy reconstruction. Lossless compression is, hence, a limit case of the lossy compression. The difference in [32] is that the residual image is vector quantized, which does not allow to have an error bound to the final reconstruction.

6.1.2 Hua's method

Hua *et al.* presented in [21] a transform based microarray compression algorithm. Later, the method (with a segmentation preprocessing step) was used in a microarray compression and analysis framework named BASICA [20].

Initially, segmentation is done using a modified Mann-Whitney algorithm, presented in the same paper. The Mann-Whitney algorithm is an iterative threshold algorithm. Accordingly to Hua *et al.* , this modified version has increased speed over the previous version, since the iterative process converges much faster. The algorithm begins with a predefined threshold mask and iteratively adjusts the threshold while the Mann-Whitney test holds. Due to some irregularities usually seen in the spots edges, which negatively affect the coding performance, a post-processing is applied in [20]. This processing estimates the probability of the neighboring edge pixels being affected with noise and corrects the segmentation accordingly. The segmentation information is coded separately with a chain code, resulting in a cost around 0.05 bits/pixel.

After segmentation, a modified version of EBCOT (Embedded Block Coding with Optimized

Truncation) [76], also presented in [21], is used. The original version of EBCOT is used in JPEG2000. The modified EBCOT is an object-based wavelet transform coding scheme. Unlike the original method, the scheme introduced by Hua *et al.* has support for coding arbitrarily shaped objects (i.e, *object-based* coding support) which allows it to code spots and background separately.

Furthermore, the lossy version of the algorithm uses the object-based coding support of the modified EBCOT to lossless compress the spots and lossy-to-lossless compress the background, although lossy-to-lossless coding of the spots can also be used.

6.1.3 Faramarzpour's method

The compression method proposed by Faramarzpour *et al.* [13] starts by locating and extracting the microarray spots, isolating each spot into an individual ROI. To do so, the image is integrated separately along its rows and columns. Two signals are calculated, $Int_x(i)$ and $Int_y(j)$, in which each element is the average of a row or column. Considering an $M \times N$ image, and where $I(i,j)$ is the pixel intensity at the position (i,j), we get

$$Int_x(i) = \sum_{j=1}^{N} I(i,j)$$

$$Int_y(j) = \sum_{i=1}^{M} I(i,j)$$

The two resulting signals [12, Fig.3(a), Fig.3(b)] have period approximately equal to the spots grid spacing, having maxima on spot locations and minima on background locations. Then, the Direct Fourier Transform of these signals is calculated. The local minima of Int_x and Int_y are extracted to form two vectors. These vector entries are the coordinates of rectangular regions where the spots are located.

After spot extraction, a spiral path is applied to the ROI. The purpose of the path is to transform the ROI into 1-dimensional data with minimal number of transitions. The path is initially centered in the spot "mass center", and optimized to minimize the first order entropy of the associated signal. This minimization implies the reduction of the transitions that occur when the path reaches the spot border (see [12, Fig.1]). Since the spots have a circular (or almost circular) shape this should give very few transitions. If the spot shape deviates considerably, or the path center is not carefully chosen, an *edge effect* occurs, with

the signal having several transitions near the spot edge, significantly reducing the coding efficiency.

A linear predictor is then applied to the pixel intensities while moving along the path previously determined. The prediction is performed using already coded pixels *spatially* close. These pixels are not necessarily from the immediately previous pixels of the path.

Finally, the residual sequence is entropy coded. The residual sequences reveal different statistical properties between the spot and background regions. Therefore, to improve the coding efficiency, the residual sequence is divided and coded separately. The separation criterion is the difference in statistical behaviors, and hence, the minimization of the residuals entropy. The two resulting residual sequences are then adaptive Huffman coded.

The lossy compression algorithm is different than the lossless one, not just an adaptation. In fact, as we will see, among the presented methods this is the only case where the lossy compression has an algorithm of its own, rather than a lossy-to-lossless version of the lossless algorithm. Nevertheless, the basic ideas of this lossy algorithm are the same, and even share some steps.

Here is how the lossy algorithm works. The image is segmented into ROI's and a circle is assigned to each spot, with the circle center and radius being optimized to best fit the spot. After that, the circles are circle-to-square transformed. Then, the image (composed of all squares) is divided into 8×8 blocks, DCT transformed and quantized. At last, the DCT coefficients are entropy coded. Hence, the information loss occurs in two phases: first, in the circle assignment to each spot, which simply removes the background, second, in the quantization of the DCT coefficients.

6.1.4 MicroZip

Lonardi *et al.* [36] proposed lossless and lossy compression algorithms for microarray images (MicroZip). Initially, the microarray grid is determined in two steps. First, the griding algorithm calculates the row-by-row and column-by-column average of the whole image (these are the same as $Int_x(i)$ and $Int_y(j)$ presented by Faramarzpour). The resultant signals are then smoothed with a rectangular 25 sample low-pass filter. Due to the greater background space between subgrids, the smoothed signal should have minima in those regions, which are used to create a first estimate of the subgrids structure. This estimative is then further refined based on the high regularity of the microarray structure. After that, the same procedure (but

with a 4 sample filter) is applied to each subgrid to isolate each spot. Using the same signals, a value B is calculated as the average of the auxiliary signals minima. This value is then used as a threshold to classify the pixels as foreground or background. Note that this griding algorithm works only on images in which the grids are perfectly aligned with the image border. The final segmentation step is to adaptively circle segment the spots. With the circle center chosen as the average of the foreground pixels coordinates for the spot, the radius is chosen such that the average intensity of the pixels inside the circle is slightly bellow B.

After segmentation, each background and foreground channels is divided into the eight most significant bits (MSByte) and the eight least significant bits (LSByte) subchannels. These four channels are then entropy coded using the Burrows-Wheeler transform and arithmetic coding. The final compressed file (see Fig. 6.3) will have, therefore, some header information with griding and segmentation information, followed by the entropy coded image channels.

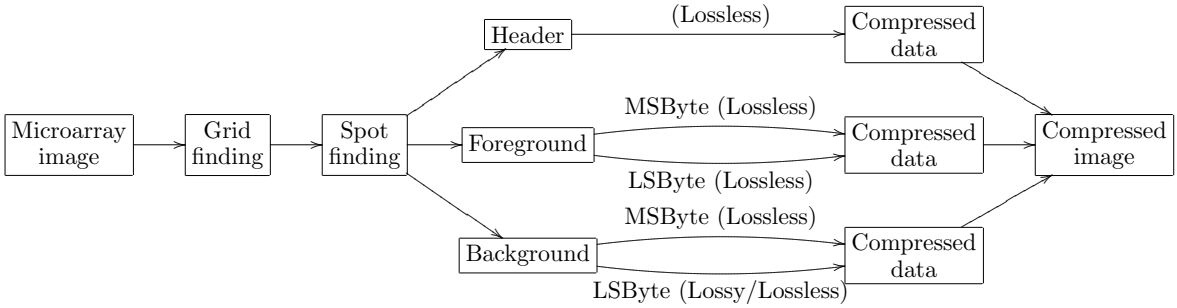


Figure 6.3: Flow chart of MicroZip

With the image information split into several channels, it is very easy to proceed to a lossy compression scheme of the image. In MicroZip, lossy compression is obtained with lossy coding of the background LSByte channel. The lossy compression is done by SPIHT (Set Partitioning in Hierarchical Trees) [69] on an image constructed from one-dimensional data stream, which is a square, as far as possible, to more effectively apply the wavelets decomposition, and likewise, to improve the compression method efficiency.

6.1.5 Zhang's method

The method proposed by Zhang *et al.* [86] is based on PPAM (Prediction by Partial Approximate Matching). PPAM is an image compression algorithm which extends the PPM text compression algorithm, considering the special characteristics of natural images [85]. Initially the microarray image is separated into its different components: background and foreground,

i.e., the microarray spots. For each component, the pixel representation is separated into its most significant and least significant parts. Then, to compress the data, the most significant part is first processed by an error prediction scheme and then residuals are encoded by the PPAM context model and encoder. The least significant part is encoded directly by the PPAM encoder. The segmentation information is saved without compression.

6.2 The use of standard image compression methods

In this section, we present a set of experiments that have been performed with the aim of providing a reference regarding the performance of standard image coding techniques, namely, lossless JPEG2000 [25, 76], JBIG [23] and JPEG-LS [24, 76], when applied to the lossless compression of microarray images. We are also interested in the study of relevant features for the microarray image compression problem, such as lossy-to-lossless reconstruction and the effect of noise, characteristic of this type of images, in the compression performance of standards.

6.2.1 Experimental results

The compression results presented in this section were obtained using microarray images that have been collected from three different publicly available sources: (1) 32 images that we refer to as the Apo AI set and which have been collected from `http://www.stat.berkeley.edu/users/terry/zarray/Html/index.html` (this set was previously used by Jörnsten *et al.* [28, 29]); (2) 14 images forming the ISREC set which have been collected from `http://www.isrec.isb-sib.ch/DEA/module8/P5_chip_image/images/`; (3) three images previously used to test MicroZip [36] and Zhang's method [86], which were collected from `http://www.cs.ucr.edu/~yuluo/MicroZip/`. These three sets have also been used in the experiments reported in [63].

Image size ranges from 1000×1000 to 5496×1956 pixels, i.e., from uncompressed sizes of about 2 MB to more than 20 MB (all images have 16 bits per pixel). The average results presented take into account the different sizes of the images, i.e., they correspond to the total number of bits divided by the total number of image pixels.

From the point of view of compression efficiency, and taking into account the results presented in Table 6.1, JPEG-LS is the overall best lossless compression method, followed by JBIG and lossless JPEG2000. The difference between JPEG-LS and lossless JPEG2000 is

about 4.1% and between JPEG-LS and JBIG is 1.7%. However, the better compression performance provided by JPEG-LS might be somewhat overshadowed by a potentially important functionality provided by the other two standards, which is progressive, lossy-to-lossless, decoding. It is interesting to note that the set for which JBIG gave the best results is also the one requiring more bits per pixel for encoding.

Image set	Gzip	JPEG2000	JBIG	JPEG-LS
APO_AI	12.711	11.063	10.851	10.608
ISREC	12.464	11.366	10.925	11.145
YuLou	11.434	9.515	9.297	8.974
Total average	**12.273**	**10.653**	**10.393**	**10.218**

Table 6.1: Compression results, in bits per pixel (bpp), using lossless JPEG2000, JBIG and JPEG-LS. For reference, results are also given for the popular compression tool GZIP. The average results presented take into account the different sizes of the images, i.e., they correspond to the total number of bits divided by the total number of image pixels.

In the case of JPEG2000, this functionality results both from the multi-resolution wavelet technology used in its encoding engine and from a strategy of information encoding based on layers [76]. In the case of JBIG, this property comes from two different sources. On one hand, images with more than one bitplane are encoded using a bitplane by bitplane coding approach. This provides a kind of progressive decoding, from most to least significant bitplanes, where the precision of the pixels is improved for each added bitplane and the L_∞ error is reduced by a factor of two. On the other hand, JBIG permits the progressive decoding of each bitplane by progressively increasing its spatial resolution [23]. However, the compression results that we present in Table 6.1, do not take into account the additional overhead implied by this encoding mode of JBIG (we used the -q flag of the encoder, which disables this mode).

6.2.2 Lossy-to-lossless compression

In Fig. 6.4, we present rate-distortion curves for image "1230c1G" from APO_AI set, obtained with the JPEG2000 and JBIG coding standards, and according to two error metrics: norm L_2 (root mean squared error) and norm L_∞ (maximum absolute error). Regarding norm L_2, we observe that JPEG2000 provides slightly better rate-distortion results for bitrates less than 8 bpp. For higher bitrates, this codec exhibits a sudden degradation of the rate-distortion.

We believe that this phenomenon is related to the default parameters used in the encoder, which might not be well suited for images having 16 bpp. Moreover, we think that a careful setting of these parameters may lead to improvements in the rate-distortion of JPEG2000 for bitrates higher than 8 bpp.

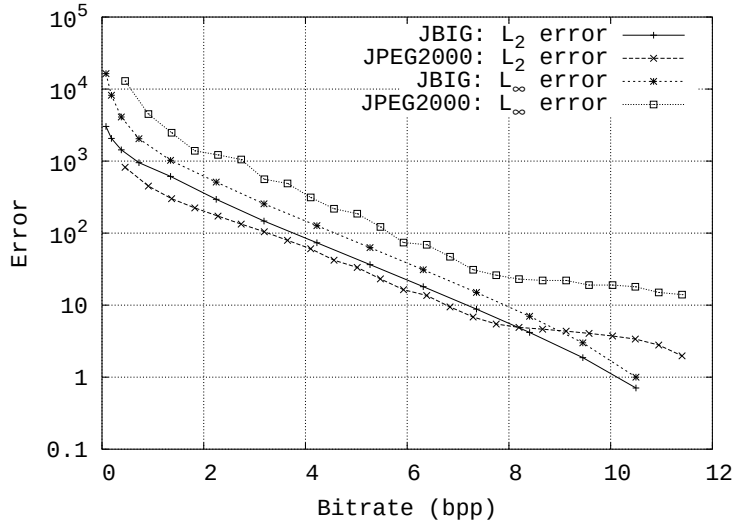


Figure 6.4: Rate distortion curves (image "1230c1G") showing the performance of JPEG2000 and JBIG in a lossy-to-lossless mode of operation. Results are given both for the L_2 (root mean squared error) and L_∞ (maximum absolute error) norms.

With respect to norm L_∞, we observe that JBIG is the one with the best rate-distortion performance. In fact, due to its bitplane by bitplane approach, it guarantees an exponential and upper bounded decrease of the maximum absolute error. The upper bound of the error is given by $2^{(16-p)} - 1$, where p is the number of bitplanes already decoded. Contrarily, JPEG2000 cannot guarantee such bound, which may be a major drawback in some cases. Finally, we note that the sudden deviation of the JPEG2000 curves around bitrates of 8 bpp is probably related to the same problem pointed out earlier for the case of the L_2 norm.

6.2.3 The effect of noise

It has been noted by Jörnsten *et al.* that, in general, the eight least significant bitplanes of cDNA microarray images are close to random and, therefore, incompressible [29]. Since this fact may result in some degradation in the compression performance of the encoders, we

decided to address this problem and to study the effect of noisy bitplanes in the compression performance of the standards.

To perform this evaluation, we separated the images into a number p of most significant bitplanes and $16-p$ least significant bitplanes. Whereas the p most significant bitplanes have been sent to the encoder, the $16-p$ least significant bitplanes have been left uncompressed. This means that the bitrate of a given image results from the sum of the bitrate generated by encoding the p most significant bitplanes plus the $16-p$ bits concerning the bitplanes that have been left uncompressed.

Table 6.2 compares average results for the three set of images regarding two situations: (1) the image is divided into the eight most significant bitplanes (which are encoded) and the eight least significant bitplanes (which are left uncompressed); (2) the optimum value of p is determined for each image. From this table, and comparing with results of Table 6.1, we can see that, in fact, this splitting operation can provide some additional compression gains. The best results attained provided improvements of 3.1%, 2.6% and 1.9%, respectively for JBIG, lossless JPEG2000 and JPEG-LS.

Image set	JPEG2000		JBIG		JPEG-LS	
	8 planes	Best	8 planes	Best	8 planes	Best
Apo_AI	10.940	10.790	10.510	10.507	10.523	10.433
ISREC	11.100	10.954	10.607	10.583	10.838	10.713
YuLou	9.918	9.321	9.506	9.030	9.588	8.912
Total average	**10.661**	**10.376**	**10.224**	**10.073**	**10.302**	**10.026**

Table 6.2: Average compression results, in bits per pixel (bpp), when a number of bitplanes is left uncompressed. Columns labeled "8 planes" provide results for the case where only the 8 most significant bitplanes have been encoded and the 8 least significant bitplanes have been left uncompressed. The column named "Best" contains the results for the case where the separation of most and least significant bitplanes has been optimally found.

However, finding the right value for p may require as many as 16 iterations of the compression phase, in order to find it. Moreover, from the results shown in Table 6.2, we can see that a simple separation of the bitplanes in an upper and lower half may improve the compression in some cases (Apo_AI and ISREC image sets), but may also produce the opposite result (YuLou image set).

6.3 Proposed method

6.3.1 Description

Although initially most of the specialized techniques for microarray image compression considered the lossy approach as a reasonable possibility [32, 28, 21, 29, 20, 12], the most recent methods focus the attention mainly on reversible techniques [13, 36, 86].

In fact, the analytic methods that are used for extracting information from the images are continuously being developed [71, 35, 34]. Keeping the original images allows future reanalysis by possibly better algorithms. Moreover, as with other biomedical related data, legal issues may also be a decisive point when choosing between maintaining or deleting the original data.

In Section 6.2 we addressed three image coding standards: JPEG2000, JBIG and JPEG-LS. Since they rely on three different coding technologies, we were able not only to evaluate the performance of each of these standards, but also to collect hints regarding what might be the best coding technology regarding microarray image compression. In that study, we concluded that from the three technologies evaluated (predictive coding in the case of JPEG-LS, transform coding in the case of JPEG2000 and context-based arithmetic coding in the case of JBIG), the technology behind JBIG seemed to be the most promising. In fact, JPEG-LS provided the highest compression, closely followed by JBIG. However, unlike JPEG2000 and JBIG, it does not provide lossy-to-lossless capabilities, a characteristic that might be of high interest, specially in the case where remote databases have to be accessed using transmission channels of reduced bandwidth. Moreover, with JBIG the image bitplanes are compressed independently, suggesting that there is some room for improvement.

Motivated by these observations, we have developed a compression method for microarray images which is based on the same technology as JBIG. However, unlike JBIG, it exploits inter-bitplane dependencies, providing coding gains in relation to JBIG.

In this section, we present a lossless compression method for microarray images based on a 3D finite-context model followed by arithmetic coding. This method was inspired by EIDAC [82], a compression method that has been used with success for coding images with a reduced number of intensities (see Chapter 3). Here, we show that similar ideas can also be used for developing an efficient compressing method for microarray images, despite the fact of these images having a large number of different intensities.

The images are compressed on a bitplane basis, starting from the most significant bitplane (MSB) and stopping at the least significant bitplane (LSB) or whenever a bitplane requires more than one bit per pixel for encoding. In this case, the rest of the bitplanes are left uncoded. As a bitplane encoder, the proposed method generates a fully embedded bitstream that enables progressive transmission and reconstruction.

The causal finite-context model that drives the arithmetic encoder uses pixels both from the bitplane currently being encoded and from the bitplanes already encoded (see Fig. 6.5). During the encoding of the image, the number of pixels in each plane used to construct the context are changing, in order to maintain the 21 bits limit of context size. This limitation is imposed in order to avoid, on one hand, the increase of memory usage and, on the other hand, the effect of context dilution. Since the decoder is able to select the same context configuration as the encoder, there is no need for side information.

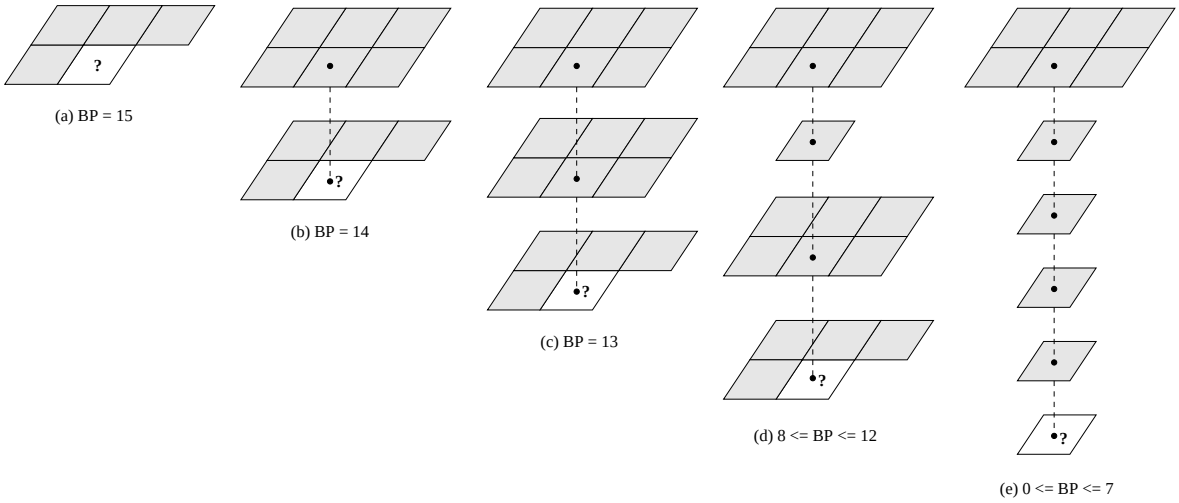


Figure 6.5: Context configurations used by the presented method at five different compression stages.

In Fig. 6.5 we show the context configurations used by the presented method at five different compression stages:

a) When encoding the most significant bitplane (four pixels of context);

b) When encoding the second most significant bitplane (ten pixels of context);

c) When encoding the third most significant bitplane (16 pixels of context);

d) From the fourth until the eighth most significant bitplanes (17–21 pixels of context);

e) The eight least significant bitplanes (13–20 pixels of context).

When encoding the eight least significant bitplanes, the finite-context model is only formed with pixels from the upper bitplanes. This procedure avoids the degradation in compression rate which occurs because, in general, the eight least significant bitplanes are close to random and, therefore, they are almost incompressible [29]. Moreover, we have introduced another mechanism that helps overcoming this problem. As the method proceeds encoding the image, the average bit-rate obtained after encoding each bitplane is monitored. If, for some bitplane, the average bit-rate exceeds one bit per pixel, then we stop the encoding process and the remaining bitplanes are saved without compression. The encoding procedure is presented in Fig. 6.6.

6.3.2 Experimental results

In this section, we present a set of experiments that have been performed to show the efficiency of the proposed method. We compare the results obtained by the proposed method with three standard image coding techniques, namely, lossless JPEG2000, JBIG and JPEG-LS. We also compare the results obtained with two specialized methods, namely MicroZip and Zhang's method.

Tables 6.3, 6.4 and 6.5 show that, for all the images used in the test, the proposed method is the best among all tested methods. Table 6.4 confirms the performance of our method relatively to the two newest specialized methods for compression of microarray images. Our method provides compression gains of 7.3% relatively to MicroZip and 4.4% relatively to Zhang's method.

Table 6.6 shows the average compression results, in bits per pixel, for the three sets of images used in this chapter regarding the use of JBIG and the proposed method. In this table, we also show the effect of using Gray codes instead of the natural binary codes for representing the pixel intensities. Gray coding has been proposed as a mean of improving the compression attained by JBIG in graylevel images [1].

Compared to JBIG, the method that we present in this section provides an overall compression gain of about 5.6%. Moreover, as can be seen in Table 6.6, the compression results provided by JBIG improved, on average, about 3.6% when using Gray codes. However, for the ISREC image set, it resulted in a small loss of performance. On the other hand, the presented

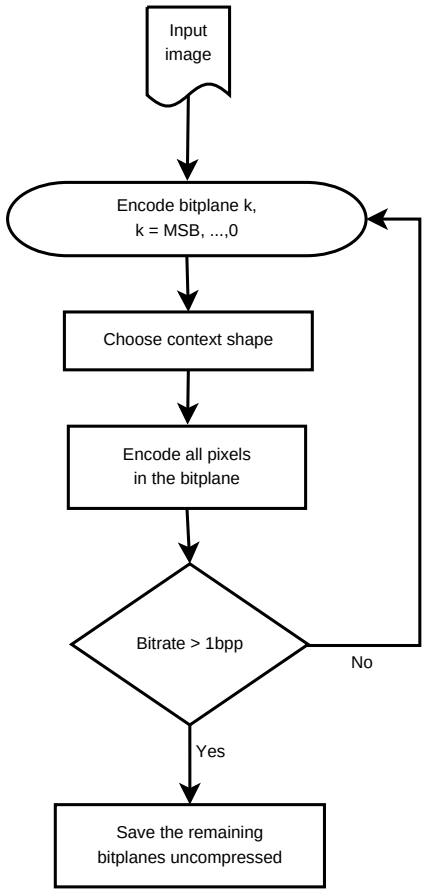


Figure 6.6: Encoding procedure of the proposed method. The choice of the context shape is based on Fig. 6.5. Being a bitplane based encoder, it is possible monitoring the bitrate used to encode each bitplane.

method was quite insensible to the use of Gray codes. On average, it improved less than 0.2%, which can be justified by its exploitation of inter-bitplane dependencies. Unlike JBIG, this (small) improvement was consistent: none of the image sets suffered a loss of compression performance when using Gray codes.

Figure 6.7 shows, for three different images, the average number of bits per pixel that are needed for representing each bitplane. As expected, this value generally increases when going from most significant bitplanes to least significant bitplanes. For the case of images "`Def661Cy3`" and "`1230c1G`" it can be seen that the average number of bits per pixel required by the eight least significant bitplanes is close to one, as pointed out in [29]. However, image

Image	Gzip	JPEG2000	JBIG	JPEG-LS	Proposed
Def661Cy3	12.658	11.914	11.218	11.713	10.406
Def661Cy5	11.418	9.714	9.451	9.392	8.874
Def662Cy3	11.636	10.881	10.007	10.575	9.161
Def662Cy5	12.722	11.369	11.251	11.156	10.555
Def663Cy3	12.437	11.903	11.023	11.665	10.121
Def663Cy5	11.961	10.405	10.124	10.151	9.528
Def664Cy3	12.322	11.592	10.813	11.384	10.001
Def664Cy5	13.142	11.768	11.755	11.632	11.138
Def665Cy3	13.363	12.462	12.111	12.289	11.436
Def665Cy5	14.451	13.590	13.429	13.557	12.663
Def666Cy3	11.768	10.946	10.132	10.659	9.305
Def666Cy5	13.116	11.727	11.748	11.572	11.043
Def667Cy3	11.690	10.540	9.923	10.248	9.218
Def667Cy5	11.807	10.304	9.951	10.033	9.331
Average	**12.464**	**11.366**	**10.925**	**11.145**	**10.199**

Table 6.3: Compression results, in bits per pixel (bpp), using lossless JPEG2000, JBIG, JPEG-LS and the proposed method in the ISREC set. For reference, results are also given for the popular compression tool GZIP.

Image	Gzip	JPEG2000	JBIG	JPEG-LS	MicroZip	Zhang	Proposed
array1	13.385	12.027	11.819	11.590	11.490	11.380	11.105
array2	11.470	9.272	9.071	8.737	9.570	9.260	8.628
array3	10.375	8.599	8.351	7.996	8.470	8.120	7.962
Average	**11.434**	**9.515**	**9.297**	**8.974**	**9.532**	**9.243**	**8.826**

Table 6.4: Compression results, in bits per pixel (bpp), using lossless JPEG2000, JBIG and JPEG-LS, MicroZip, Zhang's method and the proposed method in the YuLou set. For reference, results are also given for the popular compression tool GZIP.

"`array3`" shows a different behavior. Because this image is less noisy, the compression algorithm is able to exploit redundancies even in lower bitplanes. This is done without compromising the compression efficiency of noisy images due to the mechanism that monitors the average number of bits per pixel required for encoding each bitplane (compression switches off when it exceeds one bit per pixel).

Image	Gzip	JPEG2000	JBIG	JPEG-LS	Proposed
1230c1G	13.263	11.864	11.544	11.408	10.896
1230c1R	13.181	11.488	11.226	11.002	10.642
1230c2G	13.198	11.805	11.630	11.463	11.011
1230c2R	13.097	11.424	11.343	11.052	10.800
1230c3G	12.729	11.190	10.879	10.715	10.288
1230c3R	12.483	10.618	10.461	10.143	9.940
1230c4G	12.849	11.272	11.122	10.876	10.469
1230c4R	12.803	10.936	10.854	10.528	10.341
1230c5G	12.531	10.958	10.633	10.452	10.002
1230c5R	12.371	10.488	10.307	9.975	9.756
1230c6G	12.691	11.268	10.962	10.792	10.309
1230c6R	12.721	11.102	10.982	10.696	10.474
1230c7G	12.777	11.130	10.818	10.652	10.203
1230c7R	12.449	10.451	10.316	9.982	9.793
1230c8G	12.874	11.332	11.094	10.884	10.439
1230c8R	12.966	11.204	11.076	10.785	10.540
1230ko1G	12.410	10.766	10.369	10.206	9.978
1230ko1R	12.695	10.979	10.606	10.422	10.272
1230ko2G	12.465	10.852	10.618	10.410	10.044
1230ko2R	12.528	10.768	10.631	10.324	10.135
1230ko3G	12.822	11.309	11.013	10.833	10.409
1230ko3R	12.674	10.925	10.761	10.477	10.196
1230ko4G	12.510	10.976	10.697	10.516	10.077
1230ko4R	12.609	10.887	10.730	10.409	10.215
1230ko5G	12.795	11.286	11.100	10.881	10.427
1230ko5R	12.589	10.874	10.704	10.409	10.149
1230ko6G	12.594	11.086	10.917	10.679	10.232
1230ko6R	12.459	10.659	10.546	10.208	10.040
1230ko7G	12.752	11.278	10.929	10.785	10.298
1230ko7R	12.554	10.772	10.613	10.295	10.036
1230ko8G	12.669	11.173	10.965	10.737	10.275
1230ko8R	12.644	10.889	10.785	10.448	10.247
Average	**12.711**	**11.063**	**10.851**	**10.608**	**10.280**

Table 6.5: Compression results, in bits per pixel (bpp), using lossless JPEG2000, JBIG, JPEG-LS and the proposed method in the APO_AI set. For reference, results are also given for the popular compression tool GZIP.

Image Set	JBIG		Proposed	
	Bin Code	Gray Code	Bin Code	Gray Code
Apo_AI	11.367	10.851	10.280	10.258
ISREC	10.849	10.925	10.199	10.194
YuLou	9.788	9.297	8.840	8.826
Total average	**10.783**	**10.393**	**9.826**	**9.809**

Table 6.6: Average compression results, in bits per pixel, using JBIG and the presented method, combined with Gray codes and the natural binary codes.

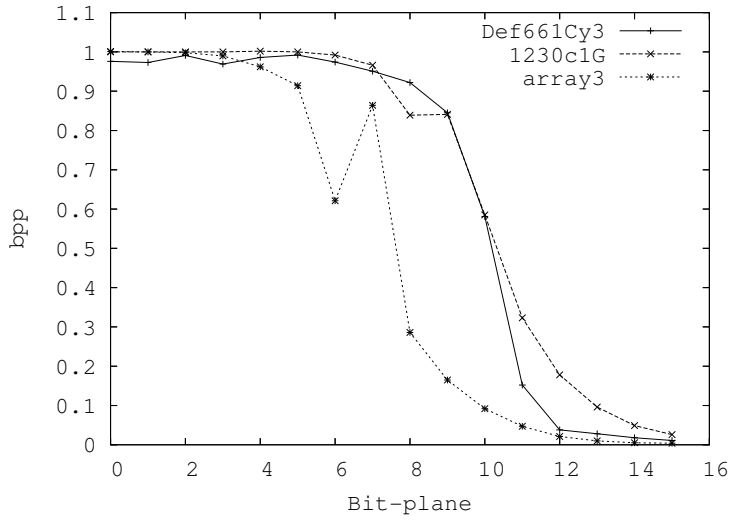


Figure 6.7: Average number of bits per pixel required for encoding each bitplane of three different microarray images (one from each test set).

6.3.3 Complexity

As can be seen in Fig. 6.5, the number of pixels composing the finite-context varies depending on the bitplane that is being encoded, ranging from a minimum of four to a maximum of 21. Since the coding alphabet is binary, this implies a maximum of $2 \times 2^{21} = 4\,194\,304$ counters that can be stored in approximately 16 MB of computer memory. In a Pentium 4 computer at 2 GHz with 512 MB of memory, the YuLou test set (three images totaling approximately 21 million pixels) required approximately four minutes to compress. Decoding time is similar (the algorithm is symmetrical). It is important to note that this compression time is only indicative, because the code has not been optimized for speed. Regarding the standard image coding methods, JBIG requires approximately one minute to compress the YuLou test set, JPEG-LS requires approximately one minute and fifteen seconds and JPEG2000 requires approximately two minutes.

6.4 Final remarks

From our study regarding the use of image compression standards, we concluded that JPEG-LS gives the best lossless compression performance. Moreover, according to the implementations used (not necessarily optimized for speed) it is about four times faster than the other two. However, it lacks lossy-to-lossless capability, which might be a decisive functionality if remote transmission over slow links is a requirement. Regarding the rate-distortion performance, JPEG2000 was the best algorithm according to the L_2 error metric, whereas JBIG was the most efficient considering the L_∞ norm. Regarding lossless compression performance, JBIG was consistently better than JPEG2000.

The method that gained most from a correct separation of most significant bitplanes (that are encoded) and least significant bitplanes (that are left uncompressed) was JBIG. It is, simultaneously, the encoding technique that, due to the bitplane by bitplane coding, can search for the optimum point of separation more easily. In fact, this can be done by monitoring the bitrate resulting from the compression of each bitplane, and stopping compressing when this value is over 1 bpp. It also worths mentioning that since JBIG was designed for bi-level images, the bitplanes are compressed independently. Therefore, techniques based on the same technology, but exploiting inter-bitplane dependencies, most probably could do better.

Based on these observations, we presented an efficient method for lossless compression of microarray images, allowing progressive, lossy-to-lossless decoding. This method is based on

bitplane compression using finite-context models and arithmetic coding. It does not require griding and/or segmentation as most of the specialized methods that have been proposed do. This may be an advantage if only compression is sought, since it reduces the computational complexity of the method. Moreover, since it does not depend on image content, it is robust, for example, against layout changes in spot placement.

The results obtained were better than those provided by the image coding standards and by two recent specialized methods (MicroZip and Zhang's method). Compared to the specialized methods, the proposed method was the best for all images in the YuLou test set, showing lossless compression gains of more than 4.4%.

Finally, we should mention that the compression method that we present in this chapter still has room for improvements. For example, we believe that changing the predefined context layouts (shown in Fig. 6.5) by some adaptive scheme that takes into consideration the evolution of the curves depicted in Fig. 6.7 could avoid some abrupt variations in bit-rate, such as the one that can be found in the curve of image "`array3`" (bitplane number seven). This and other aspects will be addressed in future work.

Chapter 7

Conclusions and future work

This thesis presents efficient preprocessing and complete methods for lossless compression of images with specific characteristics. We studied problems associated with the compression of simple images, color-quantized images and microarray images.

Simple images are usually characterized by having a small number of intensities, at least locally, and by having a sparse histogram of image intensities. In Chapter 3, we considered several preprocessing methods, based on histogram packing, that could be used together with standard image compression methods for improving their performance. These preprocessing methods consist on suitable image intensity mappings with the aim of creating a smooth image. We provide experimental results showing the effectiveness of these methods. The proposed histogram packing with a limited number of symbols provides globally the best results.

In Chapter 4, we discussed the specialized image compression techniques for lossless compression of color-indexed images. These images are represented by a matrix of indexes (the index image) and by a color-map or palette. The indexes in the matrix point to positions in the color-map and, therefore, establish the colors of the corresponding pixels. We studied the most successful methods designed for this type of images and we proposed a modified version of the method developed by Chen *et al.* , where a new context adaptation model was developed. The experimental results show the effectiveness of the model, which is currently state-of-the-art for this type of images.

Color-indexed images are characterized by having a locally sparse histogram. Based on this observation, we developed a preprocessing method, denoted region histogram packing, that

improves the lossless compression of this type of images when a standard image compression method is used. This method applies an off-line packing procedure on consecutive image regions. These regions might have fixed size and shape or an adaptive shape and size, accordingly to the local image characteristics.

In Chapter 5, we studied the most important palette reordering methods and we developed new algorithms and new approaches. In fact, for a particular color-indexed image, the mapping between index values and colors is not unique — it can be arbitrarily permuted, under the condition that the corresponding index image is changed accordingly. However, for most continuous-tone image coding techniques, these alternative representations are far from being equivalent. Palette reordering is a class of preprocessing methods aiming at finding a permutation of the color palette such that the resulting image of indexes is more amenable for compression. These preprocessing techniques have the advantage of not requiring post-processing and of being costless in terms of side information. The modifications of the method developed by Zeng *et al.* , based on a theoretical analysis, gave a significant improvement of the compression results.

The use of block-based palette reordering, an original approach, tries to explore the local characteristics of the images. This method gives a significant improvement when compared with a global reordering approach, as shown by the experimental results obtained.

Bitplane reordering has the aim of permuting the indexes of a color-indexed image in such way that the resulting binary images of each bitplane contains less and larger regions, improving the compression when used in association with compression methods that are bitplane based, as is the case of JBIG. This work allow us to conclude that this standard method is the best to compress color-indexed images. Moreover, we conclude that the bitplane-based approach and modified Zeng's method provide competitive compression results and are faster than Memon's method, the best palette reordering method, "being the best choice" to compress this type of images.

The DNA microarray technology has become an important tool in the study of gene function, regulation, and interaction across a large number of genes, and even entire genomes. The raw data of a microarray experiment consist of a pair of 16 bits per pixel grayscale images. These images are analyzed using a variety of software tools which extract relevant information, such as the intensity of the spots and the background level. This information is then used to evaluate the expression level of individual genes. In Chapter 6, we presented an efficient method for lossless compression of microarray images, allowing progressive, lossy-to-lossless

decoding. This method is based on bitplane compression using finite-context models and arithmetic coding. It does not require griding and/or segmentation as most of the specialized methods that have been proposed do. At the moment, our method is the state-of-the-art for this type of images, as shown in the experimental results presented.

7.1 Future work

At the end of this thesis, we observe that more work could be done to improve the methods that we have studied, as well as in the development of new algorithms. A brief discussion about possible directions for future research are presented next.

- A new algorithm for the calculation of the best number of symbols to use in the histogram packing algorithm, based on the characteristics of the image.
- To explore and improve the other parts of the algorithm presented in Section 4.2 and try to adapt these method to encode other type of images (preliminary results have been presented in [61]).
- To study an efficient way to construct arbitrary regions to be used with the region-based histogram packing. Moreover, to develop an efficient compression method to represent these regions.
- The development of new palette reordering methods based on the knowledge acquired in this work.
- The study of a new compression method for color-quantized images using bitplane decomposition, based on the results obtained with JBIG and with our bitplane-based palette reordering method.
- The improvement of the method studied for the compression of microarray images, in particular the development of an adaptive scheme to change the context layout accordingly to the characteristics of the image.
- The development of a new compression method that could use efficiently the relation between the two microarray images generated in each microarray experiment.

Appendix A

Image test sets

This Appendix provides the images used to evaluate the performance of the compression methods presented in this thesis.

A.1 Simple images

Set 1

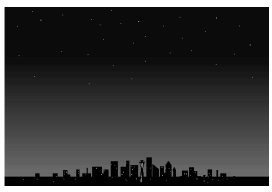
This set (corresponding to the first group of images in Tables 3.1, 3.2, and 3.3) is a gray-scale-converted version of a set used by Ausbeck in its PWC coder [4], a compression method designed for compressing palette images.

benjerry

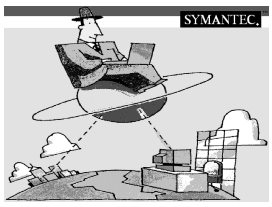
gate

netscape

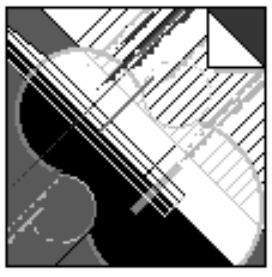
yahoo


sea_dusk


sunset



winaw


music

books

cmpndd

cmpndn

Set 2

This set (second group in Tables 3.1, 3.2, and 3.3) comprises several natural images and has the objective of testing the compression performance of the methods in images that are not "simple" (this set was also used in [82]).

lena

woman

bikeh

goldhill

aerial2

cafe

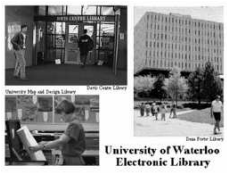
bike

Set 3

This set (last group of images in Tables 3.1, 3.2, and 3.3) is composed of five images taken from the BragZone archive.

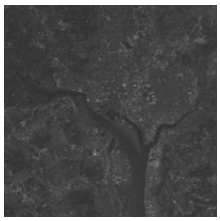
france


mountain



library


frog

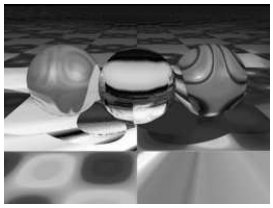

washsat

A.2 Color-indexed images

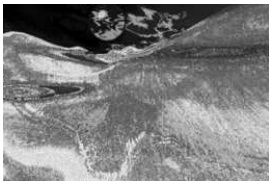
Set 1

This set of images, used in Tables 5.1, 5.2, is composed of 18 computer-generated images having different numbers of colors and geometries. Color quantization was applied only to the images originally with a number of colors greater than 256 ("clegg", "cwheel", "frymire", "house" and "serrano"). Here we present the index images of the color-indexed images.

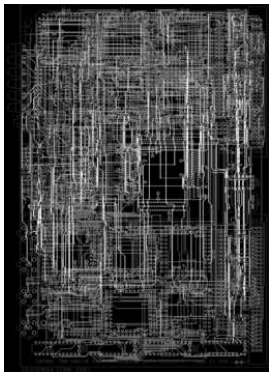

party8


cwheel

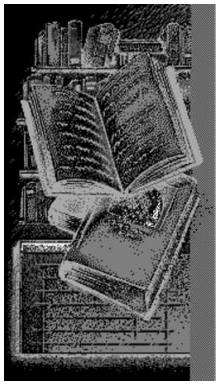

descent


fractal


clegg

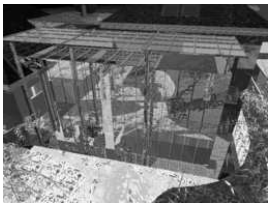

pc

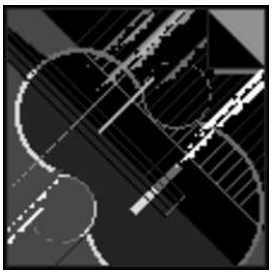

serrano


books


frymire


sea_dusk


ghouse


music

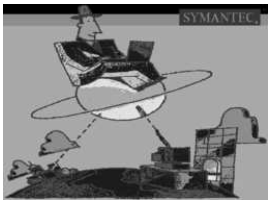
gate

benjerry

netscape


yahoo


sunset

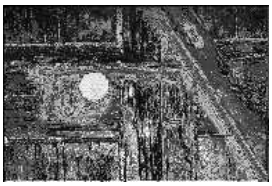


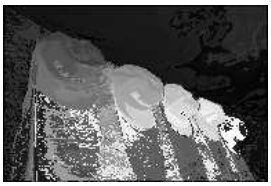
winaw

Natural1 set

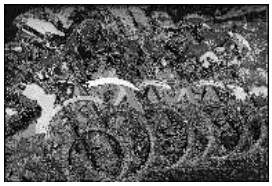
The set "natural1", also known as the "kodak" set, is composed of 23 768×512 natural images. Here we present the index image of the color-indexed images.


01


02


03


04


05


06


07


08


09


10


11

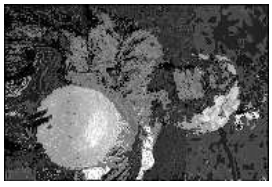

12


13


14

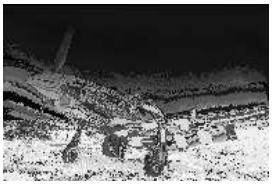

15


16


17


18


19


20


21


22


23

Natural2 set

The set "natural2" contains twelve popular natural images. Color quantization was applied to the images in this set originating images with 256 colors. Here we present the index image of the color-indexed images.

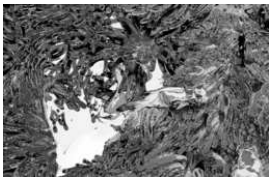

airplane


peppers


boat


baboon


arial


anemone


monarch


tulips


bike3


house


girl

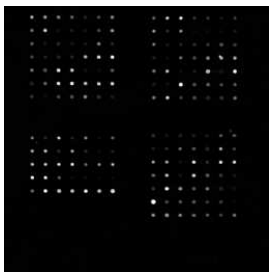

lena

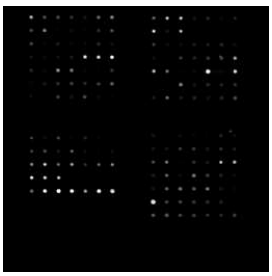
A.3 Microarray images

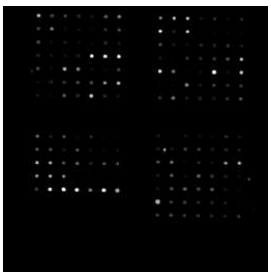
The compression results presented in Chapter 6 were obtained using microarray images that have been collected from three different publicly available sources. Image size ranges from 1000×1000 to 5496×1956 pixels, i.e., from uncompressed sizes of about 2 MB to more than 20 MB (all images have 16 bits per pixel).

ISREC set

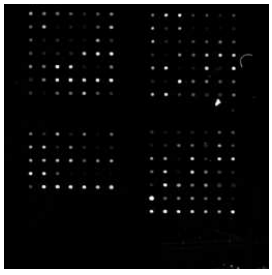
The 14 images forming the ISREC set have been collected from `http://www.isrec.isb-sib.ch/DEA/module8/P5_chip_image/images/`.

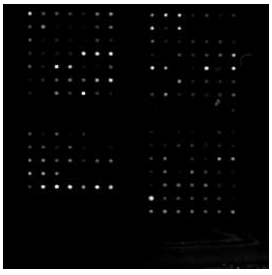

Def661Cy3


Def661Cy5


Def662Cy3


Def662Cy5

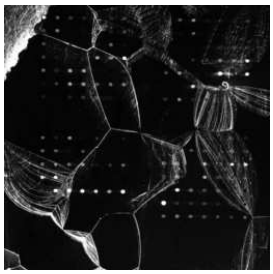

Def663Cy3

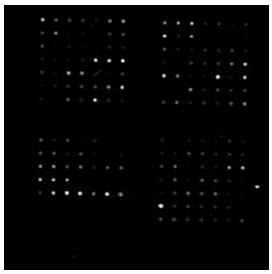

Def663Cy5


Def664Cy3

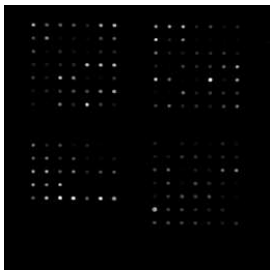

Def664Cy5

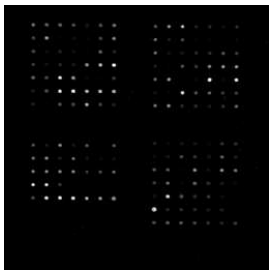

Def665Cy3


Def665Cy5


Def666Cy3

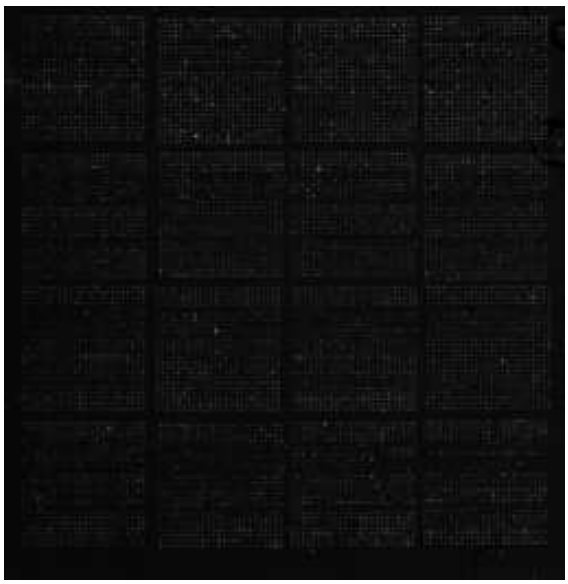

Def666Cy5

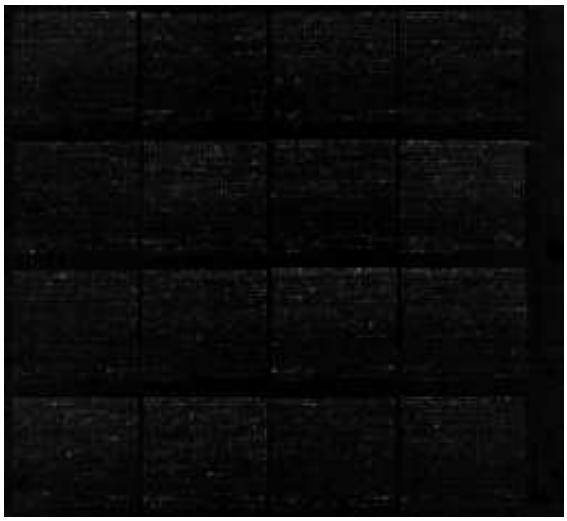

Def667Cy3

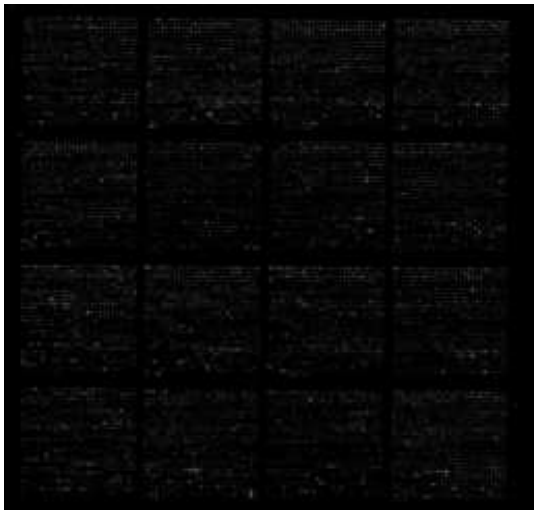

Def667Cy5

YuLou set

The three images previously used to test MicroZip[36] and Zhang's method[86] were collected from `http://www.cs.ucr.edu/~yuluo/MicroZip/`. We only present a fraction of images "array2" and "array3" due to their high resolution.


array1

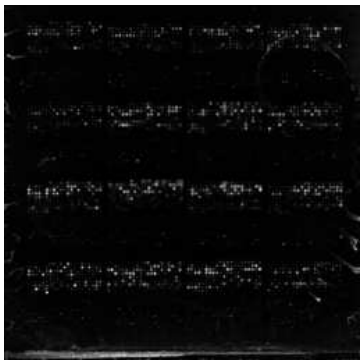

array2



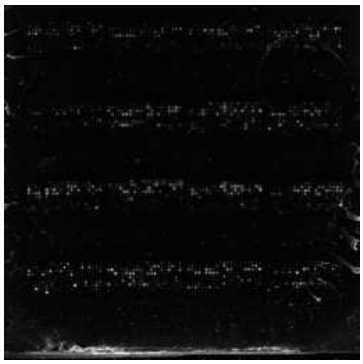
array3

APO_AI set

The 32 images that we refer to as the Apo AI set were collected from `http://www.stat.berkeley.edu/users/terry/zarray/Html/index.html` (this set was previously used by Jörnsten *et al.* [28, 29]).



1230c1G

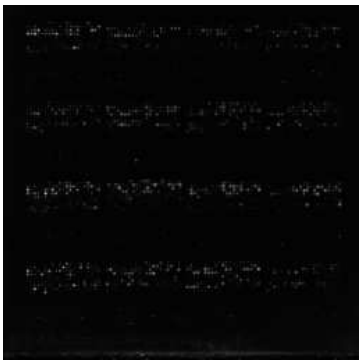


1230c1R

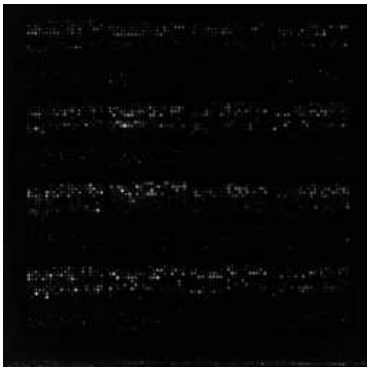
1230c2R

1230c2G

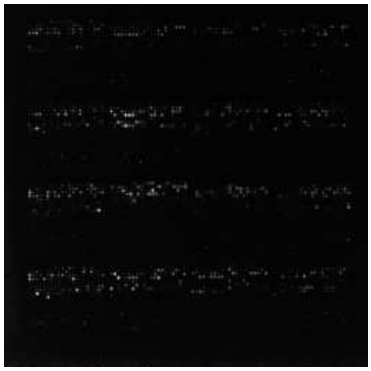
1230c3G



1230c3R



1230c4G

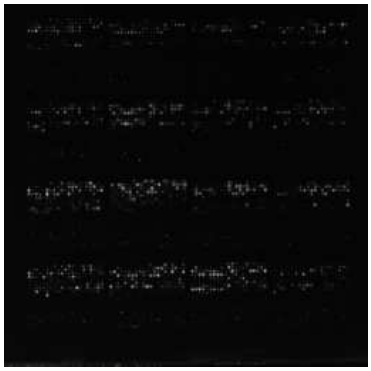


1230c4R

1230c5G

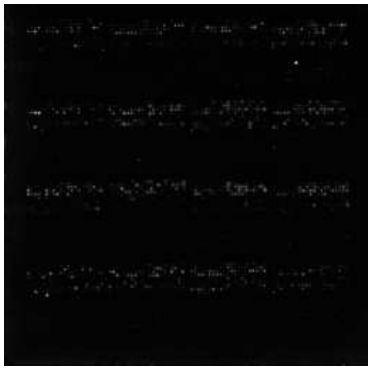
1230c5R

1230c6G

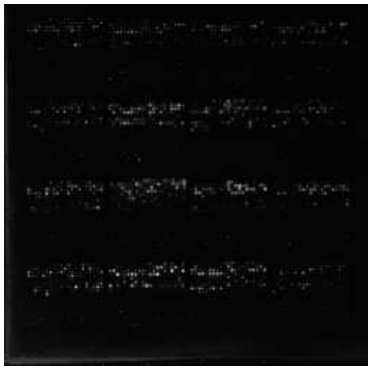


1230c6R

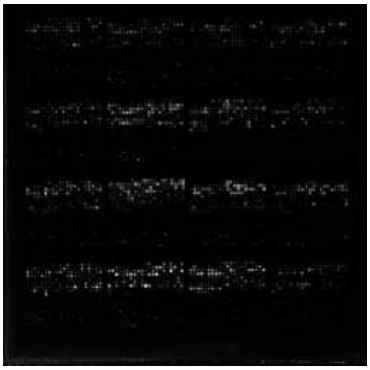
1230c7G



1230c7R



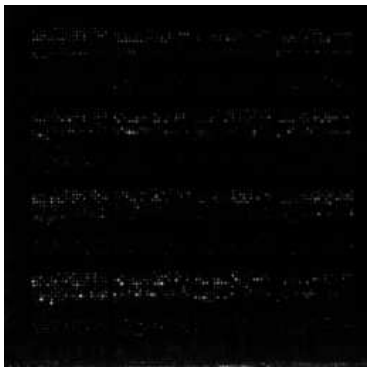
1230c8R



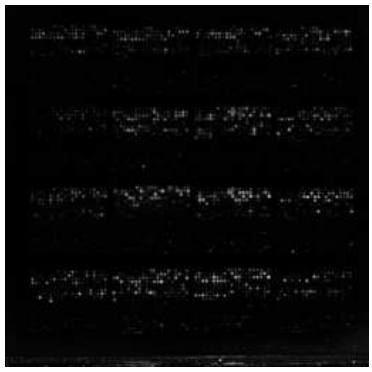
1230c8G

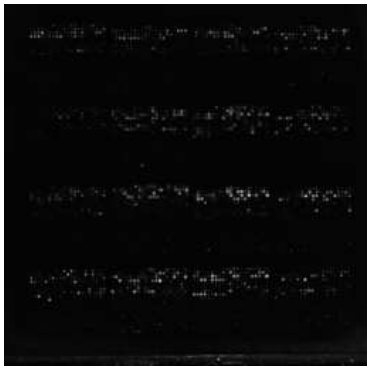
1230ko1G

1230ko1R

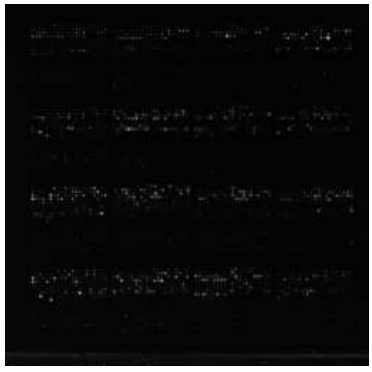

1230ko2G


1230ko2R

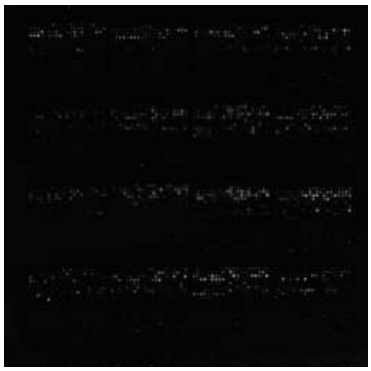

1230ko3G

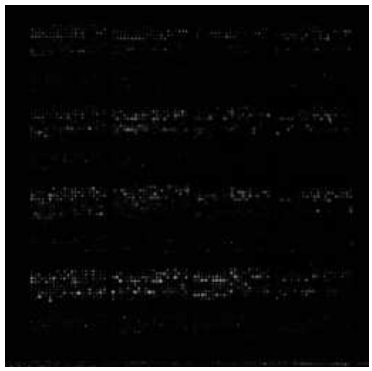

1230ko3R

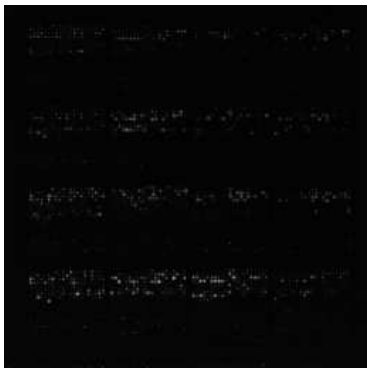

1230ko4G


1230ko4R

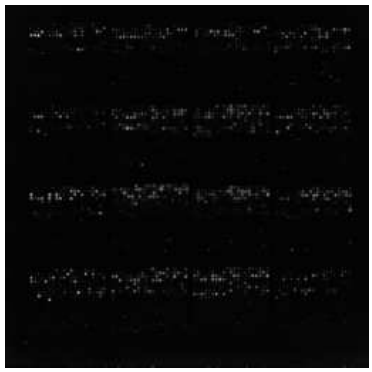

1230ko5G

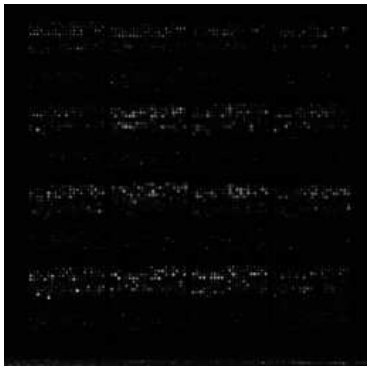

1230ko5R


1230ko6G

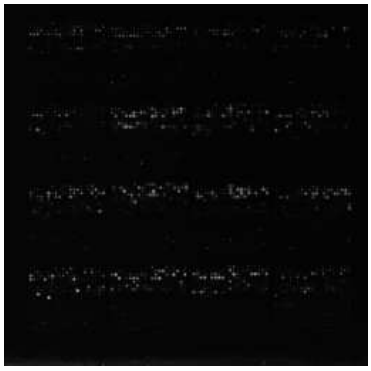

1230ko6R


1230ko7G


1230ko7R



1230ko8G



1230ko8R

Bibliography

[1] M. Abdat and M. G. Bellanger. Combining Gray coding and JBIG for lossless image compression. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-94*, volume III, pages 851–855, Austin, TX, November 1994.

[2] P. J. Ausbeck Jr. Context models for palette images. In *Proc. of the Data Compression Conf., DCC-98*, pages 309–318, Snowbird, Utah, April 1998.

[3] P. J. Ausbeck Jr. A streaming piecewise-constant model. In *Proc. of the Data Compression Conf., DCC-99*, pages 208–217, Snowbird, Utah, March 1999.

[4] P. J. Ausbeck Jr. The piecewise-constant image model. *Proceedings of the IEEE*, 88(11):1779–1789, November 2000.

[5] R. Barequet and M. Feder. SICLIC: A simple inter-color lossless image coder. In *Proc. of the Data Compression Conf., DCC-99*, pages 501–510, Snowbird, Utah, March 1999.

[6] S. Battiato, G. Gallo, G. Impoco, and F. Stanco. A color reindexing algorithm for lossless compression of digital images. In *Proc. of the IEEE Spring Conf. on Computer Graphics*, pages 104–108, Budmerice, Slovakia, April 2001.

[7] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text compression.* Prentice Hall, 1990.

[8] B. Carpentieri, M. J. Weinberger, and G. Seroussi. Lossless compression of continuous-tone images. *Proceedings of the IEEE*, 88(11):1797–1809, November 2000.

[9] X. Chen, J. f. Feng, and S. Kwong. Lossy and lossless compression for color-quantized images. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2001*, pages 870–873, Thessaloniki, Greece, 2001.

[10] X. Chen, S. Kwong, and J.-F. Feng. A new compression scheme for color-quantized images. *IEEE Trans. on Circuits and Systems for Video Technology*, 12(10):904–908, October 2002.

[11] C. Christopoulos, A. Skodras, and T Ebrahimi. The JPEG2000 still image coding system: an overview. *IEEE Trans. on Consumer Electronics*, 46(4):1103–1127, November 2000.

[12] N. Faramarzpour and S. Shirani. Lossless and lossy compression of DNA microarray images. In *Proc. of the Data Compression Conf., DCC-2004*, page 538, Snowbird, Utah, March 2004.

[13] N. Faramarzpour, S. Shirani, and J. Bondy. Lossless DNA microarray image compression. In *Proc. of the 37th Asilomar Conf. on Signals, Systems, and Computers, 2003*, volume 2, pages 1501–1504, November 2003.

[14] J. Fojtík and V. Hlaváč. Invisible modification of the palette color image enhancing lossless compression. In *Electronic Imaging: Processing, Printing, and Publishing in Color — Proc. of the SPIE*, volume 3409, pages 242–252, Zurich, Switzerland, May 1998.

[15] R. Gallager and D. V. Voorhis. Optimal source codes for geometrically distributed integer alphabets. *IEEE Transactions on Information Theory*, IT-21:228–230, Mar. 1975.

[16] S. W. Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, IT-12:399–401, July 1996.

[17] A. C. Hadenfeldt and K. Sayood. Compression of color-mapped images. *IEEE Trans. on Geoscience and Remote Sensing*, 32(3):534–541, May 1994.

[18] H. Hampel, R. B. Arps, C. Chamzas, D. Dellert, D. L. Duttweiler, T. Endoh, W. Equitz, F. Ono, R. Pasco, I. Sebestyen, C. J. Starkey, S. J. Urban, Y. Yamazaki, and T. Yoshida. Technical features of the JBIG standard for progressive bi-level image compression. *Signal Processing: Image Communication*, 4(2):103–111, April 1992.

[19] P. Hegde, R. Qi, K. Abernathy, C. Gay, S. Dharap, R. Gaspard, J. Earle-Hughes, E. Snesrud, N. Lee, and John Q. A concise guide to cDNA microarray analysis. *Biotechniques*, 29(3):548–562, September 2000.

[20] J. Hua, Z. Liu, Z. Xiong, Q. Wu, and K. Castleman. Microarray BASICA: background adjustment, segmentation, image compression and analysis of microarray images. In

Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2003, volume 1, pages 585–588, Barcelona, Spain, September 2003.

[21] J. Hua, Z. Xiong, Q. Wu, and K. Castleman. Fast segmentation and lossy-to-lossless compression of DNA microarray images. In *Proc. of the Workshop on Genomic Signal Processing and Statistics, GENSIPS*, Raleigh, NC, October 2002.

[22] International Standard ISO/IEC 15948:2004. *Information technology - Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification*, 2004.

[23] ISO/IEC. *Information technology - Coded representation of picture and audio information - progressive bi-level image compression.* International Standard ISO/IEC 11544 and ITU-T Recommendation T.82, March 1993.

[24] ISO/IEC. *Information technology - Lossless and near-lossless compression of continuous-tone still images.* ISO/IEC 14495–1 and ITU Recommendation T.87, 1999.

[25] ISO/IEC. *Information technology - JPEG 2000 image coding system.* ISO/IEC International Standard 15444–1, ITU-T Recommendation T.800, 2000.

[26] ISO/IEC. *Information technology - Lossless and near-lossless compression of continuous-tone still images: extensions.* ISO/IEC 14495–2, 2000.

[27] ISO/IEC. *JBIG2 bi-level image compression standard.* International Standard ISO/IEC 14492 and ITU-T Recommendation T.88, 2000.

[28] R. Jörnsten, Y. Vardi, and C.-H. Zhang. On the bitplane compression of microarray images. In Y. Dodge, editor, *Proc. of the 4th Int. L1-norm Conf.*, 2002.

[29] R. Jörnsten, W. Wang, B. Yu, and K. Ramchandran. Microarray image compression: SLOCO and the effect of information loss. *Signal Processing*, 83:859–869, 2003.

[30] R. Jörnsten and B. Yu. Comprestimation: microarray images in abundance. In *Proc. of the Conf. on Information Sciences*, Princeton, NJ, March 2000.

[31] R. Jörnsten and B. Yu. Compression of cDNA microarray images. In *Proc. of the IEEE Int. Symp. on Biomedical Imaging, ISBI-2002*, pages 38–41, Washington, DC, July 2002.

[32] R. Jörnsten, B. Yu, W. Wang, and K. Ramchandran. Compression of cDNA and inkjet microarray images. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2002*, volume 3, pages 961–964, Rochester, NY, September 2002.

[33] R. Jörnsten, B. Yu, W. Wang, and K. Ramchandran. Microarray image compression and the effect of compression loss. In *Proc. of the Workshop on Genomic Signal Processing and Statistics, GENSIPS*, Raleigh, NC, October 2002.

[34] R. Kothapalli, S. J. Yoder, S. Mane, and T. P. Loughran Jr. Microarray results: how accurate are they? *BMC Bioinformatics*, 3, 2002.

[35] Y. F. Leung and D. Cavalieri. Fundamentals of cDNA microarray data analysis. *Trends on Genetics*, 19(11):649–659, November 2003.

[36] S. Lonardi and Y. Luo. Gridding and compression of microarray images. In *Proc. of the IEEE Computational Systems Bioinformatics Conference, CSB-2004*, Stanford, CA, August 2004.

[37] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of JPEG-2000. In *Proc. of the Data Compression Conf., DCC-2000*, pages 523–541, Snowbird, Utah, March 2000.

[38] N. D. Memon and A. Venkateswaran. On ordering color maps for lossless predictive coding. *IEEE Trans. on Image Processing*, 5(5):1522–1527, November 1996.

[39] A. Mojsilović and E. Soljanin. Color quantization and processing by Fibonacci lattices. *IEEE Trans. on Image Processing*, 10(10):1712–1725, November 2001.

[40] S. K. Moore. Making chips to probe genes. *IEEE Spectrum*, 38(3):54–60, March 2001.

[41] A. N. Netravali and B. G. Haskell. *Digital pictures: representation, compression and standards*. Plenum, New York, 2nd edition, 1995.

[42] Network Working Group. *RFC 1951: DEFLATE compressed data format Specification*, 2006.

[43] A. J. R. Neves and A. J. Pinho. Improving the JPEG-LS compression of images with locally sparse histograms. In *Proc. of the 12th Portuguese Conf. on Pattern Recognition, Recpad-2002*, Aveiro, Portugal, June 2002.

[44] A. J. R. Neves and A. J. Pinho. Lossless compression of color-quantized images using block-based palette reordering. In *Proc. of the Int. Conf. on Image Analysis and Recognition, ICIAR-2004*, volume 1, pages 277–284, Porto, Portugal, September 2004.

[45] A. J. R. Neves and A. J. Pinho. A bit-plane approach for lossless compression of color-quantized images. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2006*, volume II, pages 429–432, Toulouse, France, May 2006.

[46] A. J. R. Neves and A. J. Pinho. Lossless compression of microarray images. In *Proc. of the IEEE International Conference on Image Processing, ICIP 2006*, pages 2505–2508, Atlanta, GA, October 2006.

[47] A. J. R. Neves, A. J. Pinho, and A. R. C. Paiva. Lossless bit-plane compression of microarray images using 3D context models. In *Proc. of the 5th IASTED Int. Conf. on Visualization, Imaging, and Image Processing, VIIP-2005*, Benidorm, Spain, September 2005.

[48] M. T. Orchard and C. A. Bouman. Color quantization of images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[49] M. T. Orchard and C. A. Bouman. Color quantization of images. *IEEE Trans. on Signal Processing*, 39(12):2677–2690, December 1991.

[50] A. W. Paeth. *Image file compression made easy.* Academic Press Inc., 1991.

[51] N. Papamarkos, A. E. Atsalakis, and C. P. Strouthopoulos. Adaptive color reduction. *IEEE Trans. on Systems, Man, and Cybernetics — Part B: Cybernetics*, 32(1):44–56, February 2002.

[52] A. J. Pinho. An online preprocessing technique for improving the lossless compression of images with sparse histograms. *IEEE Signal Processing Letters*, 9(1):5–7, January 2002.

[53] A. J. Pinho and A. J. R. Neves. Improvement of the lossless compression of images with quasi-sparse histograms. In *Signal Processing XI — Theories and Applications, Proc. of the 11th European Signal Processing Conf., EUSIPCO-2002*, volume II, pages 467–470, Toulouse, France, September 2002.

[54] A. J. Pinho and A. J. R. Neves. Block-based histogram packing of color-quantized images. In *Proc. of the IEEE Int. Conf. on Multimedia and Expo, ICME-2003*, volume 1, pages 341–344, Baltimore, MD, July 2003.

[55] A. J. Pinho and A. J. R. Neves. JPEG 2000 coding of color-quantized images. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2003*, volume 2, pages 181–184, Barcelona, Spain, September 2003.

[56] A. J. Pinho and A. J. R. Neves. On the efficiency of luminance-based palette reordering of color-quantized images. In *Proc. of the Iberian Conf. on Pattern Recognition and Image Analysis, IbPRIA-2003*, pages 766–772, Puerto de Andratx, Spain, June 2003.

[57] A. J. Pinho and A. J. R. Neves. A note on Zeng's technique for color re-indexing of palette-based images. *IEEE Signal Processing Letters*, 11(2):232–234, February 2004.

[58] A. J. Pinho and A. J. R. Neves. Palette reordering under an exponential power distribution model of prediction residuals. In *IEEE Int. Conf. on Image Processing, ICIP-2004*, pages 501–504, Singapore, October 2004.

[59] A. J. Pinho and A. J. R. Neves. A survey on palette reordering methods for improving the compression of color-indexed images. *IEEE Trans. on Image Processing*, 13(11):1411–1418, January 2004.

[60] A. J. Pinho and A. J. R. Neves. A context adaptation model for the compression of images with a reduced number of colors. In *Proc. of the IEEE International Conference on Image Processing, ICIP 2005*, volume II, pages 738–741, Genoa, Italy, September 2005.

[61] A. J. Pinho and A. J. R. Neves. Lossy-to-lossless compression of images based on binary tree decomposition. In *Proc. of the IEEE International Conference on Image Processing, ICIP 2006*, pages 2257–2260, Atlanta, GA, October 2006.

[62] A. J. Pinho and A. J. R. Neves. On the relation between Memon's and the modified Zeng's palette reordering methods. *Elsevier Image and Vision Computing*, (24):534–540, 2006.

[63] A. J. Pinho, A. R. C. Paiva, and A. J. R. Neves. On the use of standards for microarray lossless image compression. *IEEE Trans. on Biomedical Engineering*, 53(3):563–566, March 2006.

[64] L. M. Po and W. T. Tan. Block address predictive colour quantisation image compression. *Electronics Letters*, 30(2):120–121, January 1994.

[65] J. Puzicha, M. Held, J. Ketterer, J. M. Buhmann, and D. W. Fellner. On spatial quantization of color images. *IEEE Trans. on Image Processing*, 9(4):666–682, April 2000.

[66] V. Ratnakar. RAPP: Lossless image compression with runs of adaptive pixel patterns. In *Proc. of the 32nd Asilomar Conf. on Signals, Systems, and Computers, 1998*, volume 2, pages 1251–1255, 1998.

[67] J. Rissanen. A universal data compression system. *IEEE Trans. on Information Theory*, 29(5):656–664, September 1983.

[68] J. Rissanen and G. G. Langdon, Jr. Universal modeling and coding. *IEEE Trans. on Information Theory*, 27(1):12–23, January 1981.

[69] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[70] D. Salomon. *Data compression - The complete reference.* Springer, 2nd edition, 2000.

[71] R. Sasik, C. H. Woelk, and J. Corbeil. Microarray truths and consequences. *Journal of Molecular Endocrinology*, 33(1):1–9, August 2004.

[72] K. Sayood. *Introduction to data compression.* Morgan Kaufmann, 2nd edition, 2000.

[73] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, September 2001.

[74] A. Spira and D. Malah. Improved lossless compression of color-mapped images by an approximate solution of the traveling salesman problem. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-2001*, volume III, pages 1797–1800, Salt Lake City, UT, May 2001.

[75] D. Taubman, E. Ordentlich, M. Weinberger, G. Seroussi, I. Ueno, and F. Ono. Embedded block coding in JPEG 2000. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2000*, volume II, pages 33–36, Vancouver, Canada, September 2000.

[76] D. S. Taubman and M. W. Marcellin. *JPEG 2000: image compression fundamentals, standards and practice*. Kluwer Academic Publishers, 2002.

[77] David Taubman. High performance scalable image compression. *IEEE Trans. on Image Processing*, 9(7):1158–1170, July 2000.

[78] P. Waldemar and T. A. Ramstad. Subband coding of color images with limited palette size. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP-94*, volume V, pages 353–356, Adelaide, Australia, April 1994.

[79] M. J. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: A low complexity, context-based, lossless image compression algorithm. In *Proc. of the Data Compression Conf., DCC-96*, pages 140–149, Snowbird, Utah, March 1996.

[80] M. J. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *IEEE Trans. on Image Processing*, 9(8):1309–1324, August 2000.

[81] Y. Yoo, Y. G. Kwon, and A. Ortega. Embedded image-domain adaptive compression of simple images. In *Proc. of the 32nd Asilomar Conf. on Signals, Systems, and Computers*, 1998.

[82] Y. Yoo, Y. G. Kwon, and A. Ortega. Embedded image-domain compression using context models. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-99*, volume I, pages 477–481, Kobe, Japan, October 1999.

[83] A. Zaccarin and B. Liu. A novel approach for coding color quantized images. *IEEE Trans. on Image Processing*, 2(4):442–453, October 1993.

[84] W. Zeng, J. Li, and S. Lei. An efficient color re-indexing scheme for palette-based compression. In *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2000*, volume III, pages 476–479, Vancouver, Canada, September 2000.

[85] Y. Zhang and D. Adjeroh. Prediction by partial approximate matching for lossless image compression. In *Proc. of the Data Compression Conf., DCC-2005*, page 494, Snowbird, Utah, 2005.

[86] Yong Zhang, Rahul Parthe, and Don Adjeroh. Lossless compression of DNA microarray images. In *Proc. of the IEEE Computational Systems Bioinformatics Conference, CSB-2005*, Stanford, CA, August 2005.

[87] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. on Information Theory*, 23:337–343, 1977.