

Lossless and nearly-lossless image compression based on combinatorial transforms

Elfitrin Syahrul

▶ **To cite this version:**

Elfitrin Syahrul. Lossless and nearly-lossless image compression based on combinatorial transforms. Other [cs.OH]. Université de Bourgogne, 2011. English. NNT : 2011DIJOS088 . tel-00750879

HAL Id: tel-00750879
https://theses.hal.science/tel-00750879

Submitted on 12 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE BOURGOGNE

ECOLE DOCTORALE ENVIRONNEMENT - SANTÉ / STIC (E2S)

THÈSE

Pour obtenir le grade de

Docteur de l'université de Bourgogne

dicipline : Instrumentation et Informatique de l'Image

par

Elfitrin SYAHRUL

le 29 Juin 2011

Lossless and Nearly-Lossless Image Compression Based on Combinatorial Transforms

---

Directeur de thése : Vincent VAJNOVSZKI

Co-encadrant de thése : Julien DUBOIS

JURY

Abderrafiâa KOUKAM	Professeur, UTBM - Belfort	Rapporteur
Sarifudin MADENDA	Professeur, Université de Gunadarma, Indonésie	Rapporteur
Vlady RAVELOMANANA	Professeur, LIAFA	Examinateur
Michel PAINDAVOINE	Professeur, Université de Bourgogne	Examinateur
Vincent VAJNOVSZKI	Professeur, Université de Bourgogne	Directeur de thèse
Julien DUBOIS	Maître de Conférences, Université de Bourgogne	Co-encadrant

ACKNOWLEDGMENTS

In preparing this thesis, I am highly indebted to pass my heartfelt thanks many people who helped me in one way or another.

Above all, I would like to appreciate Laboratoire Electronique, Informatique et Image (LE2I) - Université de Bourgogne for valuable facility and support during my research. I also gratefully acknowledge financial support from the French Embassy in Jakarta, Universitas Gunadarma and Indonesian Government.

My deepest gratitude is to my advisor, Prof. Vincent Vajnovszki and co-advisor, Julien Dubois. Without their insights and guidance, this thesis would not have been possible. During thesis period, they played a crucial role. Their insights, support and patience allowed me to finish this dissertation.

I am blessed with wonderful friends that I could not stated one by one here in many ways, my successes are theirs, too. Most importantly, SYAHRUL family, my parents, dr. Syahrul Zainuddin and Netty Syahrul, my beloved sister Imera Syahrul and my only brother Annilka Syahrul, with their unconditional support, none of this would have been possible.

RÉSUMÉ

Les méthodes classiques de compression d'image sont communément basées sur des transformées fréquentielles telles que la transformée en Cosinus Discret (DCT) ou encore la transformée discrète en ondelettes. Nous présentons dans ce document une méthode originale basée sur une transformée combinatoire celle de Burrows-Wheeler (BWT). Cette transformée est à la base d'un réagencement des données du fichier servant d'entrée au codeur à proprement parler. Ainsi après utilisation de cette méthode sur l'image originale, les probabilités pour que des caractères identiques initialement éloignés les uns des autres se retrouvent côte à côte sont alors augmentées. Cette technique est utilisée pour la compression de texte, comme le format `BZIP2` qui est actuellement l'un des formats offrant un des meilleurs taux de compression.

La chaîne originale de compression basée sur la transformée de Burrows-Wheeler est composé de 3 étapes. La première étape est la transformée de Burrows-Wheeler elle même qui réorganise les données de façon à regrouper certains échantillons de valeurs identiques. Burrows et Wheeler conseillent d'utiliser un codage Move-To-Front (MTF) qui va maximiser le nombre de caractères identiques et donc permettre un codage entropique (EC) (principalement Huffman ou un codeur arithmétique). Ces deux codages représentent les deux dernières étapes de la chaîne de compression.

Nous avons étudié l'état de l'art et fait des études empiriques de chaînes de compression basées sur la transformée BWT pour la compression d'images sans perte. Les données empiriques et les analyses approfondies se rapportant aux plusieurs variantes de MTF et EC. En plus, contrairement à son utilisation pour la compression de texte, et en raison de la nature 2D de l'image, la lecture des données apparaît importante. Ainsi un prétraitement est utilisé lors de la lecture des données et améliore le taux de compression. Nous avons comparé nos résultats avec les méthodes de compression standards et en particulier `JPEG 2000` et `JPEG-LS`. En moyenne le taux de com-

pression obtenu avec la méthode proposée est supérieur à celui obtenu avec la norme JPEG 2000 ou JPEG-LS.

Mots-clés : Transformé de Burrows-Wheeler, Compression sans perte et quasi sans perte d'images.

xvi + 107

Réf. : 81 ; 1948-2011

ABSTRACT

Common image compression standards are usually based on frequency transform such as Discrete Cosine Transform or Wavelets. We present a different approach for lossless image compression, it is based on combinatorial transform. The main transform is Burrows Wheeler Transform (BWT) which tends to reorder symbols according to their following context. It becomes a promising compression approach based on context modelling. BWT was initially applied for text compression software such as `BZIP2`; nevertheless it has been recently applied to the image compression field. Compression scheme based on Burrows Wheeler Transform is usually lossless; therefore we implement this algorithm in medical imaging in order to reconstruct every bit. Many variants of the three stages which form the original BWT-based compression scheme can be found in the literature. We propose an analysis of the more recent methods and the impact of their association. Then, we present several compression schemes based on this transform which significantly improve the current standards such as `JPEG2000` and `JPEG-LS`. In the final part, we present some open problems which are also further research directions.

Keywords: Burrows-Wheeler Transform (BWT), lossless (nearly lossless) image compression

xvi + 107

Ref. : 81 ; 1948-2011

Contents

Acknowledgment ii

Résumé iii

Abstract v

Table of Contents vi

List of Figures x

List of Tables xiv

1 Introduction **1**

1.1 Background 2

1.2 Scope and Objective 3

1.3 Research Overview 6

1.4 Contribution 10

1.5 Thesis Organization 11

2 Lossless Image Compression **13**

2.1 Introduction 14

2.2 Standards 15

2.2.1 Joint Photographic Experts Group (JPEG) 15

2.2.2 Joint Bi-level Image Experts Group (JBIG) 21

2.2.3 JPEG2000 22

2.2.4 Graphics Interchange Format (GIF) 28

2.2.5 Portable Network Graphics (PNG) 28
2.3 Miscellaneous Techniques Based on predictive methods 29
2.3.1 Simple fast and Adaptive Lossless Image Compression Algorithm (SFALIC) 29
2.3.2 Context based Adaptive Lossless Image Codec (CALIC) . . . 30
2.3.3 Fast, Efficient, Lossless Image Compression System (FELICS) 31
2.3.4 TMW 32
2.3.5 Lossy to Lossless Image Compression based on Reversible Integer DCT 34
2.3.6 Specifics lossless image compression methods 34
2.4 Compression performances 36
2.5 Summary 37

3 Compression Scheme based on Burrows-Wheeler Transform 39
3.1 Introduction 40
3.2 BWT: What? Why? Where? 40
3.3 Burrows Wheeler Transform 42
3.4 Global Structure Transform 46
3.4.1 Move-To-Front (MTF) and its variants 47
3.4.2 Frequency Counting Methods 49
3.5 Data Coding 52
3.5.1 Run-level coding 52
3.5.2 Statistical Coding 54
3.6 BWT success : Text Compression 58
3.6.1 `BZIP2` 58
3.7 Innovative Works: Image Compression 59
3.7.1 Lossy 59

3.7.1.1 BWT based JPEG 60
3.7.2 Lossless 62
3.7.2.1 DWT and BWT 62
3.7.2.2 A Study of Scanning Paths for BWT Based Image Compression 62
3.7.2.3 The Impact of Lossless Image Compression to Radiographs. 63
3.7.3 Nearly Lossless 63
3.7.3.1 Segmentation-Based Multilayer Diagnosis Lossless Medical Image Compression 63
3.8 Summary 64

4 Improvements on classical BWCA **65**
4.1 Introduction 66
4.2 Corpus 66
4.3 Classic BWCA Versus Text and Image Compression Standards . . . 68
4.4 Some of Pre-processing Impacts in BWCA 72
4.4.1 Image Block 72
4.4.2 Reordering 75
4.5 GST Impact 77
4.6 Data Coding Impact 80
4.6.1 Run Length Encoding Impact 80
4.6.2 Statistical Coding 83
4.7 Nearly Lossless BWCA Compression 91
4.8 Bit Decomposition in BWCA 94

5 Conclusion and Perspectives **99**

List of Figures

1.1 Data compression model [5]. 4

1.2 The benefit of prediction. 9

2.1 Basic model for lossless image compression [54, 51]. 14

2.2 DCT-based encoder simplified diagram [37, 74]. 15

2.3 JPEG image division [67]. 16

2.4 Example of JPEG encoding [67]. 17

2.5 Lossless encoder simplified diagram [37]. 17

2.6 Locations relative to the current pixel, X. 18

2.7 Probability distribution function of the prediction errors. 19

2.8 Block diagram of JPEG-LS [76]. 20

2.9 JBIG sequential model template. 22

2.10 The JPEG2000 encoding and decoding process [66]. 23

2.11 Tiling, DC-level shifting, color transformation (optional) and DWT for each image component [66]. 24

2.12 Dyadic decomposition. 25

2.13 Example of Dyadic decomposition. 25

2.14 The DWT structure. 27

2.15 Example of FELICS prediction process [35]. 32

2.16 Schematic probability distribution of intensity values for a given context $\Delta = H - L$ [35]. 32
2.17 TMW block diagram [48]. 33

3.1 Original scheme of BWCA method. 40
3.2 Transformed data of the input string `abracadabraabracadabra` by the different stages [4]. 41
3.3 The BWT forward. 43
3.4 Relationship between BWT and suffix arrays. 44
3.5 Different sorting algorithms used for BWT. 45
3.6 The BWT inverse transform. 46
3.7 Forward of Move-To-Front. 47
3.8 Forward of Move One From Front. 48
3.9 Forward of Time Stamp or Best 2 of 3 Algorithm. 49
3.10 Example of Huffman coding. (a) Histogram. (b) Data list. 54
3.11 Sample of Huffman tree. 54
3.12 Creating an interval using given model. 56
3.13 BZIP2 principle method. 59
3.14 The first proposed methods of Baiks [14]. 61
3.15 The second proposed methods of Baik [34]. 61
3.16 Wisemann proposed method [78]. 61
3.17 Guo proposed scheme [33]. 62
3.18 Lehmann proposed methods [41]. 63
3.19 Bai et al. proposed methods [13]. 64

4.1 Example of tested images. Upper row: directly digital, lower row: secondarily captured. From left to right: Hand; Head; Pelvis; Chest, Frontal; Chest, Lateral. 67

4.2 The first tested classic chain of BWCA. 68

4.3 Block size impact to compression efficiency in images [78]. Tests are obtained with three JPEG parameters, which are 100, 95, and 90. . . . 73

4.4 Example of an image divided into a small blocks. 74

4.5 Block of quantised coefficients [57]. 75

4.6 6 examples of data reordering. 76

4.7 A few of reordering methods. 76

4.8 Method 1 of BWCA. 76

4.9 The "Best x of $2x-1$" analysis for x=3,4,5,...,16. 79

4.10 Example of Run Length Encoding two symbols (RLE2S) 80

4.11 Second proposed method of BWCA (M2). 81

4.12 The third proposed method of BWCA (M3). 81

4.13 The fourth proposed method of BWCA (M4). 84

4.14 Gain percentage of M4-Bx11 and JPEG-LS. 89

4.15 Gain percentage of M4-Bx11 and JPEG2000. 90

4.16 Our proposed method of JPEG-LS and M4-Bx11 combination. . . . 90

4.17 The background segmentation process. 92

4.18 Morphological closing operation: the downward outliers are eliminated as a result [17]. 92

4.19 Morphological opening operation: the upward outliers are eliminated as a result [17]. 93

4.20 Method T1 with bit decomposition - MSB LSB processed separately. . 95

4.21 Average CR for different of AC parameters. 95

4.22 Four proposed methods by arranging MSB and LSB blocks. (a) 4 bits decomposition by rearrange decomposition bit. (b) T2: each LSB group is followed by its corresponding MSB group. (c) T3: each MSB group is followed by its corresponding LSB group. (d) T4: alternating LSB and MSB blocks, starting with LSB block. (e) T5: alternating MSB and LSB blocks, starting with MSB block. 96
4.23 T6: 4 bits image decomposition by separating MSB and LSB consecutively. The two neighbor BWT output are combined into a byte. . . . 97
4.24 T7: 4 bits image decomposition by separating MSB and LSB consecutively. The two neighbor of GST output are combined into a byte. . . 98

List of Tables

1.1 Current image compression standards [32]. 5

2.1 Predictor used in Lossless JPEG. 19

2.2 Example of Lossless JPEG for the pixel sequence (10, 12, 10, 7, 8, 8, 12) when using the prediction mode 1 (i.e. the predictor is the previous pixel value). The predictor for the first pixel is zero. 19

2.3 Example of Golomb-Rice codes for three values of k. Note, bits constituting the unary section of the code are shown in bold. 21

2.4 Comparison results of a few lossless image compression techniques. . 27

2.5 Functionality matrix. A "+" indicates that it is supported, the more "+" the more efficiently or better it is supported. A "-" indicates that it is not supported [60]. 36

2.6 Compression performances of natural and medical images for different compression techniques [68]. 37

3.1 Percentage share of the index frequencies of MTF output. 48

3.2 Inversion Frequencies on a sample sequence. 50

3.3 Example of Wheeler's RLE [28]. 53

3.4 Probability table. 57

3.5 Low and high value. 58

4.1 The Calgary Corpus. 66

4.2 The Canterbury Corpus [12]. 67
4.3 IRMA Corpus. 67
4.4 Comparison results of Canterbury Corpus using our BWCA classic chain and BZIP2. 69
4.5 Compression performances of Calgary Corpus using classical BWCA, BW94, and BZIP2. 69
4.6 Experimental results on Calgary Corpus using different BWCA chain. 69
4.7 Compression results using BWCA Classic chain for Lukas corpus[72]. 71
4.8 CR comparable results between classical BWCA and existing image standards for IRMA corpus. 72
4.9 The effect of varying block size (N) on compression of book1 from the Calgary Corpus [18]. 73
4.10 Results of image blocks in Figure 4.4. 74
4.11 Image compression ratios for different type of reordering methods [70, 71]. 77
4.12 Comparative results of MTF and its variants using up-down reordering [70]. 78
4.13 Comparative results of MTF and its variants using spiral reordering[70]. 78
4.14 M1-M4 results for IRMA images using Spiral reordering. 82
4.15 M1-M4 results for IRMA images using Ud-down reordering. 82
4.16 Results comparison of M4-Bx11, JPEG-LS and JPEG2000. 85
4.17 Detail results of M4-Bx11. 85
4.18 Results summary of M4-Bx11. 87
4.19 Compression ratios of nearly-lossless using M4-Bx11. 93

Chapter 1

Introduction

Contents

1.1 Background 2
1.2 Scope and Objective 3
1.3 Research Overview 6
1.4 Contribution 10
1.5 Thesis Organization 11

1.1 Background

Data compression is an old and eternally new research field. Maybe the first example of data compression is stenography (or shorthand writing) which is a method that increases speed and brevity of writing. Until recently, stenography was considered an essential part of secretarial training as well as being useful for journalists. The earliest known indication of shorthand system is from Ancient Greece, from mid-4^th century BC. More recently, example of data compression method is the Morse code invented in 1838 for the use in telegraphy. It uses shorted codewords for more frequent letters: for example, the letter E is coded by "." and T by "- ". Generally, data compression is used in order to reduce expensive resources, such as storage memory or transmission bandwidth. The *Information Theory* founded by Clause Shannon in 1949 gives very elegant issues in data compression, but it does not give us any answer knowing how much a file can be compressed by any method that is, what is its entropy. Its value depends on the information source – more specifically, the statistical nature of the source. It is possible to compress the source, in a lossless manner, with compression rate close to the entropy rate. He has shown that it is mathematically impossible to do better than entropy rate by using appropriate coding method [23, 64].

The data compression field remains forever as an open research domain: indeed, given a particular input, it is computationally undecidable if a compression algorithm is the best one for this particular input. Researchers are searching many algorithms and data types. The best solution is to classify files and match the data type to the correct algorithm.

Data compression domain is still an interesting topic also because of the number of files/data keep growing exponentially. There are many examples of fast growing digital data. The first example is in radiography and medical imaging. Hospitals and clinical environments are rapidly moving toward computerization that means digitization, processing, storage, and transmission of medical image. And there are two reason why these data keep growing exponentially. Firstly, because of the growth of the patients that need to be scanned. Secondly, is the conversion of archived film medical images. The second example of the growing digital data is in the oil and gas industry that has been developing what's known as the "digital oilfield", where sensors monitoring activity at the point of exploration and the wellhead connect to information system at headquarters and drive operation and exploration decisions in real time, that means more data produce each day. And there are more applications that produce data exponentially such as broadcast, media, entertainment, etc [30].

Those digitized data have heightened the challenge for ways to manage and organize the data (compression, storage, and transmission). This purpose is emphasized, for example, by non compression of raw image with size of 512×512 pixels, each pixel is represented by 8 bits, contains 262 KB of data. Moreover, image size will be tripled if the image is represented in color. Furthermore, if the image is composed for a video that needs generally 25 frames per second for just a one second of color film, requires approximately 19 megabytes of memory. So, a memory with 540 MB can store only about 30 seconds of film. Thus, data compression process is really obvious to represent data into small size possible, therefore less storage capacity will be needed and data transmission will be faster than uncompressed data.

Modern data compression began in the late 1940s with the development of information theory. Compression efficiency is the principal parameter of a compression technique, but it is not sufficient by itself. It is simple to design a compression algorithm that achieves a low bit rate, but the challenge is how to preserve the quality of the reconstructed information at the same time. The two main criteria of measuring the performance of data compression algorithm are compression efficiency and distortion caused by the compression algorithm. The standard way to measure them is to fix a certain bit rate and then compare the distortion caused by different methods.

The third feature of importance is the speed of the compression and decompression process. In online applications the waiting times of the user are often critical factors. In the extreme case, a compression algorithm is useless if its processing time causes an intolerable delay in the image processing application. In an image archiving system one can tolerate longer compression times if the compression can be done as a background task. However, fast decompression is usually desired.

1.2 Scope and Objective

The general approach to data compression is the representation of the source in digital form with as few bits as possible. Source can be data, still images, speech, audio, video or whatever signal needs to be stored and transmitted. In general, data compression model can be divided in three phases: removal or reduction in data redundancy, reduction in entropy and entropy coding as seen in Figure 1.1.

Removal or reduction in data redundancy is typically achieved by transforming the original data from one form or representation to another. Some of popular transformation techniques are Discrete Cosine Transform (DCT) and Discrete Wavelet Transfor-

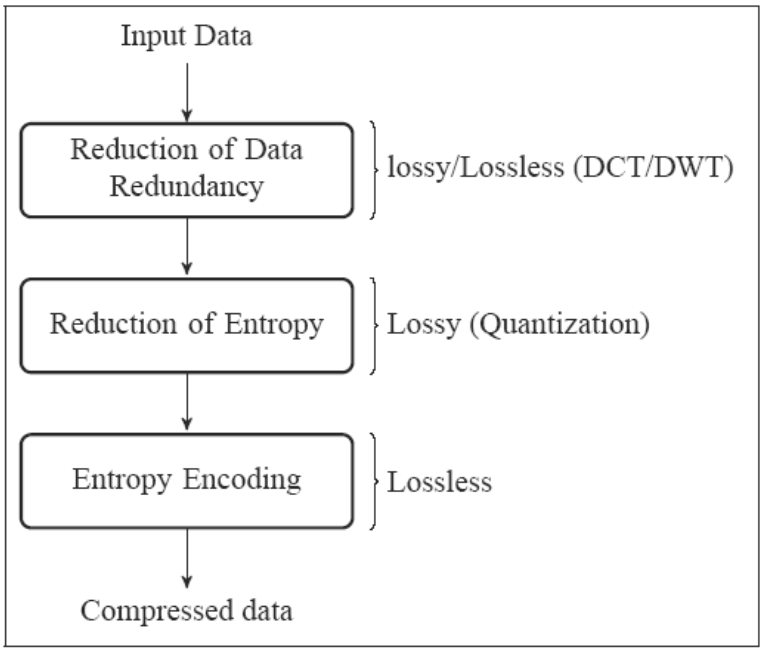


Figure 1.1: Data compression model [5].

mation (DWT). This step leads to the reduction of entropy.

Reduction in entropy phase is a non-reversible process. It is achieved by dropping insignificant information in the transformed data by some quantization techniques. Amount of quantization dictate the quality of the reconstructed data. Entropy of the quantized data is less compared to the original one, hence more compression.

And the last phase is Entropy Coding (EC) which compress data efficiently. There are some well-known of EC, such as Huffman, Arithmetic Coding (AC), etc.

Data compression can be classified in two category based on their controllability which are lossless and lossy (see Figure 1.1). It concern the quality of a data after decompression. Lossy compression (or irreversible) reconstructs a data in an approximation of the original one. The information preserving property is usually required, but not always compulsory for certain applications.

Lossless compression (or information preserving, or reversible) decompresses a data which is identical with its original. There are some applications that need to recover the essential or crucial information in it, such An expert in these fields needs to process and analyze all of the information, for example in medical imaging, scientific imaging, museums/art, machine vision, remote sensing (that is satellite imaging), etc, therefore lossless method is needed to accomplish these applications.

Figure 1.1 and Table 1.1 shows that there are a few controllability of compressed

Table 1.1: Current image compression standards [32].

Compression Method	Type of Image	Compression Ratio	Controllability
G3 Fax	Binary	1/2 - 10:1	Lossless Only
JBIG	Binary	2-40:1	Lossless Only
JPEG[a] (Baseline)	8 Bit Grayscale and Color	5-25:1	lossy
Lossless JPEG	4-12 Bit Image Components	2-10:1	Lossless Only
JBIG-2	Binary	2-40:1	Lossy/Lossless
JPEG-LS	4-12 Bit Image Components	2-10:1	Lossless/Near Lossless
JPEG2000	ANY	2-200:1	Lossy/Lossless

a. JPEG defines four algorithms: sequential, progressive, hierarchical, and lossless, sequential and progressive are commonly implemented, files compressed in the other algorithms, or with the QM-coder will not be decodeable by 99% of JPEG applications.

image that concern of image quality after decompression. In general, controllability can be classified in two; lossless and lossy. Lossy compression (or irreversible) reconstructs an image in an approximation of the original one. The information preserving property is usually required, but not always compulsory for certain applications. For near-lossless of JPEG-LS is done by controlling the maximal error in pixel value, thereby limiting the maximal elevation error in the reconstructed image. Sometimes this mode is called controlled lossy mode. Meanwhile, The controllability of JPEG-baseline is defined by four algorithm that based on its quantization table (Q-table), and the mode of this method is always lossy. Lossless compression (or information preserving, or reversible) decompresses an image which is identical with its original. There are some applications that need to recover the essential or crucial information in an image. An expert in these fields needs to process and analyze all of the information, for example in medical imaging, scientific imaging, museums/art, machine vision, remote sensing (that is satellite imaging), etc, therefore lossless method is needed to accomplish these applications.

Prior data compression standards serve only a limited applications. Table 1.1 shows some of current image compression standards for some type of images. For example for binary images that less than 6 bit per pixel gets better compression performances by JBIG or JBIG-2. This table also shows the range of compression ratios that obviously depend on image controllability (lossless, lossy or near-lossless) and image nature. Each compression method has the advantages and disadvantages. There does not exist the best compression method that provides the best performance for any kind of images. Table 1.1 shows that JPEG2000 can be implemented for any kind of images, but it does not mean it obtains the best performance. Some image compression methods were developed to fulfilled some requirements of certain applications and certain types of images. Moreover, people keeps searching a suitable compression

method for specific application or data type, such as medical imaging. The researches are more focused to certain type of medical images. Since there are various modalities of these images, such as computed radiography (CR), computed tomography (CT), magnetic resonance imaging (MRI)s, etc, to provide the better performances. Thus, a new image compression standard is useful to serve applications that have not provided yet for by current standards, to cover some current needs with one common system, and to provide full employment for image compression researchers.

This thesis analyzed other universal lossless data compression based on Burrows Wheeler Transform (BWT). This transform was created by Burrows and Wheeler [18] for text compression, such as `BZIP2`. Here, we analyze the implementation of this transform in text compression and also in lossless and near-lossless image compression.

1.3 Research Overview

The traditional lossless image compression methods uses spatial correlations to model image data. It can be done for gray scale images, moreover some methods have been developed and they are more complex to improve compression performances. One of them is done by converting the spatial domain of an image to frequency domain. It encodes the frequency coefficients to reduce image size, such `JPEG2000` with Discrete Wavelet Transform (DWT) that convert an image in small DWT coefficients. Meanwhile, target of spatial domain is pixels values its self. They work under principle of pixel redundancy. These techniques are taken advantage of repeated values in consecutive pixel positions. For instance, the Ziv-Lempel Algorithm which is used in Portable Network Graphics (`PNG`).

Almost all of spatial methods are lossless. They have less computational cost and easy to implement than frequency domain. For example Ziv-Lempel algorithms are simple and have some nice asymptotic optimality properties. However, frequency domain usually have better compression rate than spatial domain since most of them are used in lossy compression techniques.

The foundation of image compression is information theory, as laid down by the likes of Shannon in the late 1940s [63]. The information theory states that information of an event is:

$$I(e) = \log_2 \frac{1}{p(e)} (\text{bits}) \tag{1.1}$$

where $p(e)$ is the probability of the event occurring. Therefore, information content of event is directly proportional to our surprise at the event happening. A very unlikely event carries a lot of information, while an event that is very probable carries little information. Encoding an image can be thought of as recording a sequence of events, where each event is the occurrence of a pixel value. If we have no model for an image, we might assume that all pixel values are equally likely. Thus, for a greyscale image with 256 grey levels, we would assume $p(e) = 1/256$ for all possible pixel values. The apparent information required to record each pixel value is then $\log_2 256 = 8$. Clearly, this is no better than the raw data representation mentioned above. However, due to the spatial smoothness common in images, we expect a given pixel to be much like the one before it. If the given pixel value conforms to our expectation of being close to the previous value, then little information is gained by the event of learning the current pixel's value. Consequentially, only a little information need be recorded, so that the decoding process can reconstruct the pixel value.

This idea of using previous pixel values to lower the information content of the current pixel's encoding has gone under several names: Differential Pulse Code Modulation (DPCM), difference mapping and more generally predictive coding. From early work in the 50s and 60s on television signal coding, to modern lossless image compression schemes, predictive coding has been widely used. The common theme has always been to use previous data to predict the current pixel and then only the prediction error (or prediction residual) need be encoded. Predictive coding requires the notion of a current pixel and past pixels and this implies a one-dimensional (1D) sequence of pixels. However, image data is two-dimensional. To correct for this mismatch a 1D path is needed that visits every pixel in the 2D image. By far the most common path is raster-scan ordering, which starts at the top left of an image and works left to right, top to bottom, over the whole image.

From the practical point of view the last but often not the least feature is complexity of the algorithm itself, i.e. the ease of implementation. Reliability of the software often highly depends on the complexity of the algorithm. Let us next examine how these criteria can be measured.

Entropy and Symbol Coding

One way to get a quantitative measure of the benefits of prediction is to use entropy. This commonly used measure of information content, again provided by Shannon, says that for a collection of independent, identically distributed (i.i.d.) symbols $x_0, x_1..., x_i$,

each with a value in the range $0 \leq j \leq (N-1)$, the average information content per symbol is:

$$\sum_{j=0}^{N-1} p(x_i = j) \log_2 \frac{1}{p(x_i = j)} (\text{ bits}) \tag{1.2}$$

where $p(x_i = j)$ is the probability of a given symbol having value j. The i.i.d. assumption implies a $memoryless\ source$ model for the data. That is, it does not use information based on preceding symbols (memory) to model the value of the current symbol. Note that this assumption is almost never true at any stage in image coding, however it is a useful simplifying model at this stage. Equation 1.2 shows that the distribution of pixel values, the $p(x_i = j)$, that is important. It can be inferred that distributions that are nearly at (all symbols nearly equiprobable) have average information measures approaching $\log_2 N$, whereas sharply peaked distributions have much lower entropies.

Figure 1.2 gives an example of this and shows that simple predictive coding produces a residual image with a sharply peaked distribution. The entropy of pixel values of Figure 1.2(a) is close to the raw data size of 8 bits per pixel, otherwise in Figure 1.2(c), using the previous pixel value as predictor, the prediction errors (adjusted for display) have a much lower entropy.

Having changed the distribution of symbols so that their entropy is reduced, it remains to store them efficiently. An effective way to do this involves Variable Length Codes (VLCs), where short codes are given to frequent symbols and longer codes are given to infrequent symbols. In 1952 Huffman [36] described his algorithm for producing optimal codes of integer length. However, in the late 1970s, researchers at IBM succeeded in implementing Arithmetic Coding, which by removing the integer length constraint allows more efficient symbol coding. Image compression schemes tend to be innovative in the stages leading up to symbol coding. The final symbol coding stage is generally implemented using traditional algorithms, such as Huffman Coding.

Compression efficiency

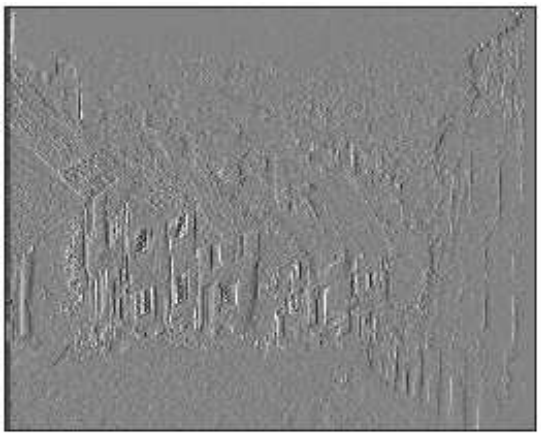
The most obvious measure of the compression efficiency is the bit rate, which gives the average number of bits per stored pixel of the image:



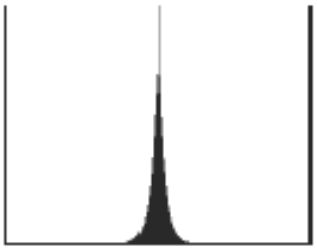
(a) Without Prediction



(b) Pixel Distribution (7.53 bits per pixel)



(c) With Prediction



(d) Pixel Distribution (5.10 bits per pixel)

Figure 1.2: The benefit of prediction.

$$\text{Bit Rate (BR)} = \frac{\text{size of compressed file}}{\text{size of uncompressed file}} k \tag{1.3}$$

where k is the number of bits per pixel in the original image. If the bit rate is very low, compression ratio might be a more practical measure:

$$\text{Compression Ratio (CR)} = \frac{\text{size of uncompressed file}}{\text{size of compressed file}} \tag{1.4}$$

The overhead information (header) of the files is ignored here. Sometimes the space savings is given instead, which is defined as the reduction in size relative to the original data size:

$$\text{space savings} = \left(1 - \frac{\text{size of compressed file}}{\text{size of uncompressed file}} \right) \times 100\% \qquad (1.5)$$

thus, data that compresses a 10 mega Bytes file into 2 mega bytes would yield a space savings 0.8 that usually notated as a percentage, 80%.

1.4 Contribution

The major result of the thesis is the development of universal data compression using BWT that is commonly used as the main transform in text compression (such as `BZIP2`). This transform is also working well in image compression, especially in medical images.

The first part of this research is mainly concerned with lossless and nearly-lossless medical image compression.

Many algorithms have been developed in the last decades. Different methods have been defined to fulfilled the lossless image applications. Obviously, none of these techniques suits all kind of images; image compression is still an open problem. Most of the techniques are based on prediction, domain transform, and entropy coding. We propose a new compression scheme based on combinatorial transform, the BWT. This original approach does not propose any domain change (as with DCT or DWT), but a modification of data order to improve coding results. This method enables to compete with usual standards even more to improve their performances. We have applied with success on medical images. A nearly-lossless version is also defined in this thesis.

The research of this thesis focuses on the following key point:

- BWT impact as an image compression transform.
- Impact of other coding to support BWT as its main algorithm in lossless image compression
- Detail analysis of this method to image compression since it is usually used in text compression

The compression results are compared with the existing image compression standard, such as `JPEG`, `JPEG-LS` and `JPEG2000`.

1.5 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 presents image lossless image compression that have been developed and used for lossless image compression applications, such as `Lossless JPEG`, `JPEG-LS`, `JPEG2000` and other lossless compression techniques.

Chapter 3 proposes a state of the art for some algorithms that form Burrows Wheeler Compression Algorithm and some compression algorithms that have been used as intermediate stages of Burrows Wheeler Compression Algorithm (BWCA) chain.

Chapter 4 details simulations and results of some of BWCA chains, and also the proposed methods for lossless and nearly-lossless medical image compression using BWT as its main transform. The compression results are interesting since BWCA proposed methods provide better CR than existing image compression standards.

Chapter 5 presents the conclusions of the thesis and also some future works. A summary of the research objective, thesis contributions are presented. Future research issues regarding further development of the framework are discussed.

Chapter 2

Lossless Image Compression

Contents

2.1 Introduction 14

2.2 Standards 15

2.2.1 Joint Photographic Experts Group (JPEG) 15

2.2.2 Joint Bi-level Image Experts Group (JBIG) 21

2.2.3 JPEG2000 22

2.2.4 Graphics Interchange Format (GIF) 28

2.2.5 Portable Network Graphics (PNG) 28

2.3 Miscellaneous Techniques Based on predictive methods 29

2.3.1 Simple fast and Adaptive Lossless Image Compression Algorithm (SFALIC) 29

2.3.2 Context based Adaptive Lossless Image Codec (CALIC) . . 30

2.3.3 Fast, Efficient, Lossless Image Compression System (FELICS) 31

2.3.4 TMW 32

2.3.5 Lossy to Lossless Image Compression based on Reversible Integer DCT 34

2.3.6 Specifics lossless image compression methods 34

2.4 Compression performances 36

2.5 Summary 37

2.1 Introduction

Some of references divided lossless image compression method in two parts: Modelling and Coding [76, 65, 73]. Penrose et al. [54, 51] add mapping stage before modelling stage as seen in Figure 2.1. This stage is less correlated with the original data. It can be a simple process such as replacing each pixel with the difference between the current and previous pixel (difference mapping), although more complex mapping often yield better results. It changes the statistical properties of the pixel data. This makes data to be encoded closer to being independent, identically distributed (i.i.d.) and therefore closes to the kind of data that can be efficiently coded by traditional symbol coding techniques.

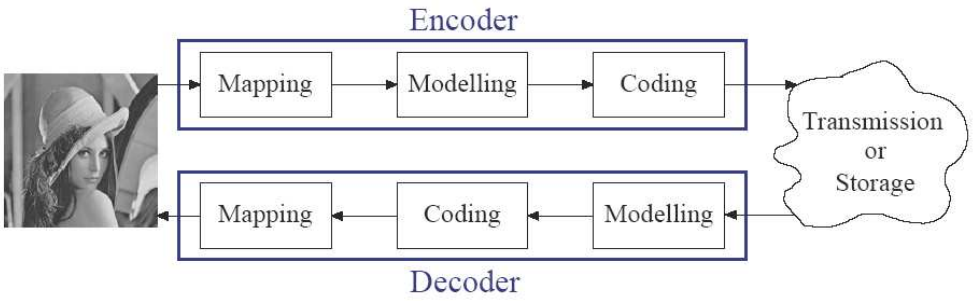


Figure 2.1: Basic model for lossless image compression [54, 51].

The modelling stage attempts to characterize the statistical properties of the mapped image data. It attempts to provide accurate probability estimates to the coding stage, and may even slightly alter the mapped data. By being mindful of higher order correlations, the modelling stage can go beyond the memoryless source model and can provide better compression than would be apparent from measuring the entropy of the mapped image data.

The symbol coding stage aims to store the mapped pixel efficiently, making use of probability estimates from the modelling stage. Symbol coders are also sometimes called statistical coders (because they use source statistics for efficient representation) and entropy coders (because they aim to represent data using no more storage than allowed by the entropy of the data). Coding schemes take a sequence of symbols from an input alphabet and produce codewords using an output alphabet. They aim to produce a code with the minimum average codeword length. The decoding process can be asymmetric with the encoding process, caused by the coding stage's dependence on the modelling stage.

This chapter presents a survey of the literature relating to lossless greyscale image compression. The literature in this field is quite extensive and it is impossible to cover exhaustively due to space constraints. However, to ensure a reasonably thorough coverage, what are considered to be good examples of all the major techniques are discussed. In particular both the old and new JPEG standards for lossless image compression are covered in some details.

2.2 Standards

2.2.1 Joint Photographic Experts Group (JPEG)

Joint photographic experts group (JPEG) is a standardization organization and also named for its image compression standards for gray scale and color images. It is a very well known standard, especially for lossy compression using Discrete Cosine Transform (DCT), but its performance decreases in the lossless case. The technique used is completely different between lossy and lossless. Lossless method uses predictor to decrease data entropy (see Figure 2.2 and Figure 2.5).

JPEG standard includes four modes of operation; Sequential coding, Progressive coding, Hierarchical coding and Lossless coding [37]. Sequential and progressive coding are generally used for lossy method.

Lossy baseline JPEG

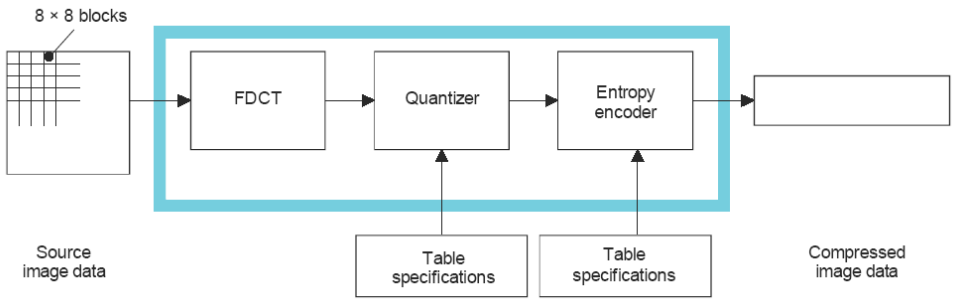


Figure 2.2: DCT-based encoder simplified diagram [37, 74].

Figure 2.2 shows an simplified of JPEG encoding of lossy method. The lossy baseline JPEG (sequential coding mode) is based on Discrete Cosine Transform (DCT) (or fast DCT /FDCT). Firstly, the image is divided into small blocks of size 8×8, each

containing 64 pixels. Three of these 8×8 groups are enlarged in this figure, showing the values of the individual pixels, a single byte value between 0 and 255 as seen in Figure 2.3. Then, each block is transformed by DCT or FDCT into a set of 64 values referred to as DCT coefficients.

Figure 2.4 shows an example of JPEG encoding for one block. DCT-based encoders are always lossy, because roundoff errors are inevitable in DCT steps. The deliberate suppression of some information produces irreversible loss and so the exact original image can not be obtained. The left column shows three 8×8 pixel groups, the same ones shown in Figure 2.3 for an eye part. The center column shows the DCT spectra of these three groups. The third column shows the error in the uncompressed pixel values resulting from using a finite number of bits to represent the spectrum.

The suppression deliberates information loss in the down sampling and quantization steps, so it is not possible to obtain an original bits.

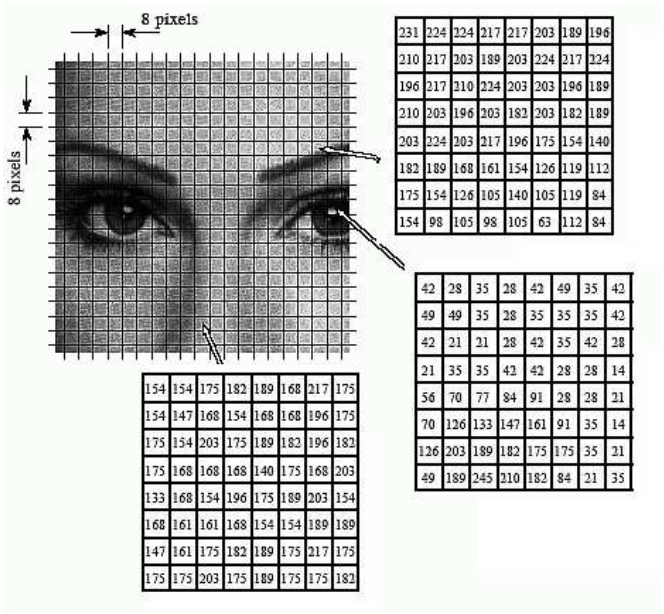


Figure 2.3: JPEG image division [67].

Entropy encoder such as Huffman coding or Arithmetic Coding is used as the last part of this method to compressed the data efficiently. For details of this transformation, the reader is referred to Recommendation T.81 of ITU [37, 74].

Lossless baseline JPEG

The lossless coding mode is completely different form the lossy one. It uses predictor method as seen in Figure 2.5 [43, 47, 50]. Figure 1.2 have shown the benefit of pre-

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

70	24	-28	-4	-2	-10	-1	0
-53	-35	43	13	7	13	1	3
23	9	-10	-8	-7	-6	5	-3
6	2	-2	8	2	-1	0	-1
-10	-2	-1	-12	2	1	-1	4
3	0	0	11	-4	-1	5	6
-3	-5	-5	-4	3	2	-3	5
3	0	4	5	1	2	1	0

0	-3	-1	-1	1	0	0	-1
1	0	-1	-1	0	0	0	-1
-1	-2	1	0	-2	0	-2	-2
-1	-2	-1	2	0	2	0	1
0	-2	1	0	0	1	0	0
0	-4	-1	0	1	0	0	0
0	-2	0	1	-1	-1	1	-1
-1	-3	1	1	1	-3	-2	-1

Figure 2.4: Example of JPEG encoding [67].

dictor that can decreased the image entropy. Decoder of lossless compression scheme must be able to produce the original image from compressed data by the encoder. To ensure this, the encoder must only make predictions on the basis of pixels whose value the decoder will already know. Therefore, if all past pixels have been losslessly decoded, the decoder's next prediction will the same as that made by the encoder. Also of importance, is that the encoder and decoder agree to the nature of the variable length coding scheme to be used. This is easy when a fixed coding scheme is used, but if an adaptive scheme is used, where the meaning of codes change over time, then the decoder must make the same adaptations. This can be achieved either by the encoder making a decision based on future pixels and transmitted that change to the decoder (forward adaptation) or by the encoder and decoder both making changes, in a predefined way, based on the values of previous pixels (backward adaptation).

The predictor stages as seen in Figure 2.5 can be divided in three elements, which are predictor itself, calculating predictor errors and modelling error distribution. There are two JPEG lossless standards, which are Lossless JPEG and JPEG−LS.

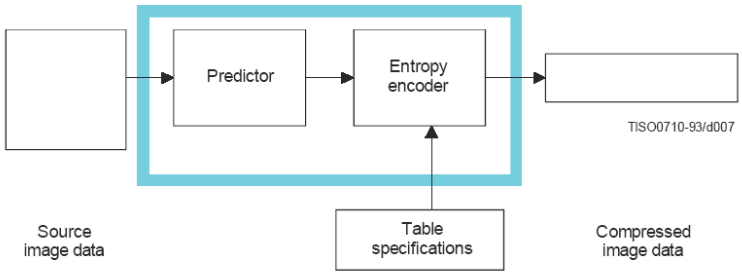


Figure 2.5: Lossless encoder simplified diagram [37].

The first Lossless JPEG uses Predictive Lossless Coding of a 2D Differential Pulse Code Modulation (DPCM) scheme. The basic premise is that the value of a pixel is combined with the values of up to three neighboring pixels to form a predictor value.

The predictor value is then subtracted from the original pixel value. When the entire bitmap has been processed, the resulting predictors are compressed using either the Huffman or the binary arithmetic entropy encoding [40].

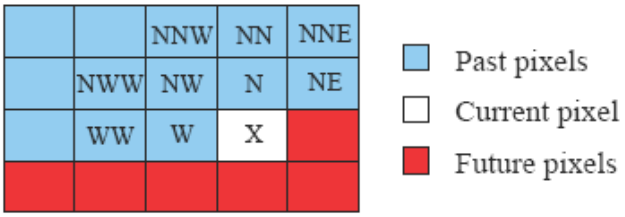


Figure 2.6: Locations relative to the current pixel, X.

Lossless JPEG proceeds the image pixel by pixel in row-major order. The value of the current pixel is predicted on the basis of the neighboring pixels that have already been coded (see Figure 2.6). There are three components of predictive modelling which are prediction of current pixel value, calculating the prediction error and modelling error distribution.

The prediction value of DPCM method is based on two of its neighboring pixels. Current pixel value (x) is predicted on the basis of the pixels that have already been coded (thus seen by the decoder too). Refer the neighboring pixels as in Figure 2.6, thus a possible predictor could be:

$$\bar{x} = \frac{(X_W) + (X_N)}{2} \tag{2.1}$$

The prediction error is the difference between the original and the predicted pixel values:

$$e = x - \bar{x} \tag{2.2}$$

The prediction error is then coded instead of the original pixel value. The probability distribution of the prediction errors are concentrated around zero while very large positive, and very small negative errors are rare to appear, thus the distribution resembles Gaussian normal distribution function where the only parameter is the variance of the distribution, see Figure 2.7.

The prediction functions available in JPEG are given in Table 2.1. The prediction errors are coded either by Huffman coding or arithmetic coding. The category of prediction value is coded first, then followed by the binary representation of the value within the corresponding category. Table 2.2 gives an simple example of seven input

pixels.

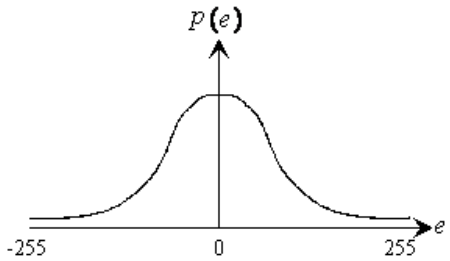


Figure 2.7: Probability distribution function of the prediction errors.

Table 2.1: Predictor used in Lossless JPEG.

Mode	Predictor	Mode	Predictor
0	Null	4	*N + W - NW*
1	*W*	5	*W + (N - NW)/2*
2	*N*	6	*N + (W - NW)/2*
3	*NW*	7	(N + W)/2

Table 2.2: Example of Lossless JPEG for the pixel sequence (10, 12, 10, 7, 8, 8, 12) when using the prediction mode 1 (i.e. the predictor is the previous pixel value). The predictor for the first pixel is zero.

Pixel	10	12	10	7	8	8	12
Prediction error	10	2	-2	-3	1	0	4
Category	4	2	2	2	1	0	3
Bit sequence	1011010	1110	1101	1100	101	0	100100

Although offering reasonable lossless compression, the performance of Lossless JPEG was never widely used outside the research community. The main lesson to be learnt from lossless JPEG is that global adaptations are insufficient for good compression performance. This fact has spurred most researchers to look at more adaptive methods. To advance upon simple schemes like Lossless JPEG, alternative methods for prediction and error modelling are needed.

Most of the previous predictors are linear functions. However, images typically contain non-linear structures. This has lead an effort to find good non-linear predictors.

One of the most widely reported predictor uses switching scheme, called Median Adaptive Predictor (MAP). This methods is used by JPEG−LS.

The JPEG−LS is based on the LOCO-I algorithm. The method uses the same ideas as the Lossless JPEG with the improvement of using context modelling and adaptive

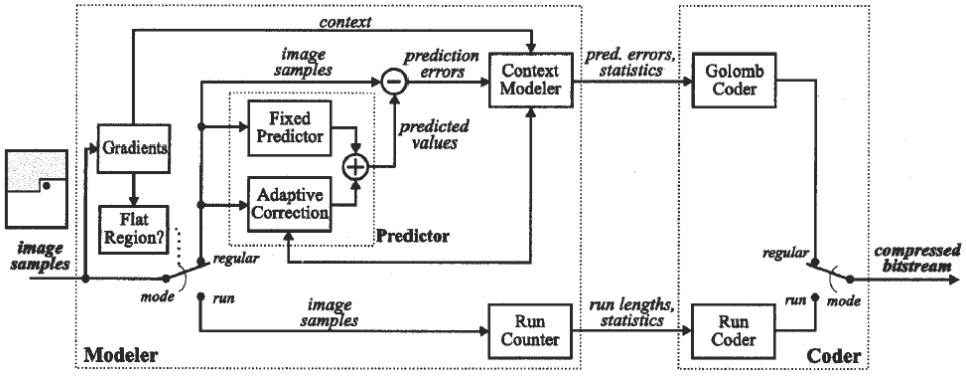


Figure 2.8: Block diagram of JPEG−LS [76].

correction of the predictor. The coding component is changed to Golomb codes with an adaptive choice of the skewness parameter. The main structure of the JPEG−LS is shown in Figure 2.8. The modelling part is divided in three components; prediction, determination of context and probability model for the prediction errors [76].

The prediction process is quite different with the previous model of Lossless JPEG. The next value (x) is predicted as based on the values of already coded in the neighboring pixels. The three nearest pixels (denoted as W, N, NW), shown in Figure 2.8, are used in the prediction as follows:

$$\bar{x} = \begin{cases} min(W,N) & \text{if } NW \geq max(W,N) \\ max(W,N) & \text{if } NW \leq min(W,N) \\ W + N - NW & \text{otherwise} \end{cases} \tag{2.3}$$

The predictor tends to pick W in cases of an horizontal edge above the current location, and N in cases of a vertical edge exists left of the current location. The third choice $(W + N - NW)$ is based on the presumption that there is a smooth plane around the pixel and uses this estimation as the prediction. This description of MAP as a scheme for adapting to edge features, lead to it being called the Median Edge Detection (MED) predictor in [75].

In order to avoid confusion, MED will be used to describe the specific predictor described above, whereas MAP will be used for the concept of using the median value from a set of predictors.

The prediction residual of Equation 2.2 is then input to the context modeler, which will decide the appropriate statistical model to be used in the coding. The context is determined by calculating the three gradient between four context pixels.

In order to help keep complexity low and reduce model cost, JPEG−LS uses a

symbol coder that requires only a single parameter to describe an adaptive code. The scheme used is Golomb-Rice (GR) coding (often known only as Rice coding) and the parameter required is k. GR codes are a subset of the Huffman codes and have been shown to be the optimal Huffman codes for symbols from a geometric distribution. Golomb-Rice codes are generated as two parts; the first is made from the k least significant bits of the input symbol and the latter is the remainder of the input symbol in unary format. Some example of codes are given in Table 2.3. Selecting the correct k to use when coding a prediction error is very important. The ideal value for k is strongly related to the logarithm of the expected prediction error magnitude [75].

Table 2.3: Example of Golomb-Rice codes for three values of k. Note, bits constituting the unary section of the code are shown in bold.

input Symbol	for k=1	for k=2	for k=3
0	**0**	0**0**	0**00**
1	**10**	1**0**	0**10**
2	**110**	0**10**	1**00**
3	**1110**	1**10**	1**10**
4	**11110**	0**110**	00**10**
5	**111110**	1**110**	0**110**
6	**1111110**	0**1110**	1**010**

JPEG−LS as one of lossless image compression still be one of the best methods. But, it has some advantages and disadvantages that will be discussed more detail in Section 2.4.

2.2.2 Joint Bi-level Image Experts Group (JBIG)

JBIG (Joint Bilevel Image Experts Group) is binary image compression standard that is based on context-based compression where the image is compressed pixel by pixel. The pixel combination of the neighboring pixels (given by the template) defines the context, and in each context the probability distribution of the black and white pixels are adaptively determined on the basis of the already coded pixel samples. The pixels are then coded by arithmetic coding according to their probabilities. The arithmetic coding component in JBIG is the QM-coder.

Binary images are a favorable source for context-based compression, since even a relative large number of pixels in the template results to a reasonably small number of contexts. The templates included in JBIG are shown in Figure 2.9.

The number of contexts in a 7-pixel template is $2^7 = 128$, and in the 10-pixel

model it is $2^{10} = 1024$, while a typical binary image of 1728×1188 pixels consists of over 2 million pixels. The larger is the template, the more accurate probability model is possible to obtain. However, with a large template the adaptation to the image takes longer; thus the size of the template cannot be arbitrary large.

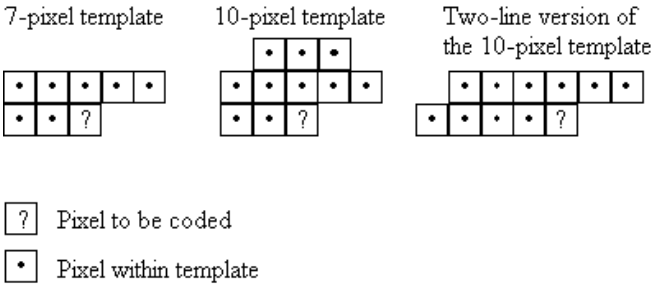


Figure 2.9: JBIG sequential model template.

The emerging standard JBIG2 enhances the compression of text images using pattern matching technique. The standard will have two encoding methods: pattern matching and substitution, and soft pattern matching. The image is segmented into pixel blocks containing connected black pixels using any segmentation technique. The content of the blocks are matched to the library symbols. If an acceptable match (within a given error marginal) is found, the index of the matching symbol is encoded. In case of unacceptable match, the original bitmap is coded by a JBIG-style compressor. The compressed file consists of bitmaps of the library symbols, location of the extracted blocks as offsets, and the content of the pixel blocks.

The main application of this method is for image fax compression. Binary images that less than 6 bpp obtain better compression performances by JBIG, more about this results will be discussed in Section 2.4.

2.2.3 JPEG2000

JPEG2000 standards are developed for a new image coding standard for different type of still images and with different image characteristics. The coding system is intended for low bit-rate applications and exhibit rate distortion and subjective image quality performance superior to existing standards [21].

The JPEG2000 compression engine (encoder and decoder) is illustrated in block diagram form in Figure 2.10. the encoder, the discrete transform is first applied on the source image data. The transform coefficients are then quantized and entropy coded before forming the output code stream (bit stream). The decoder is the reverse of the

encoder. The code stream is first entropy decoded, de-quantized, and inverse discrete transformed, thus resulting in the reconstructed image data. Although this general block diagram looks like the one for the conventional JPEG, there are radical differences in all of the processes of each block of the diagram.

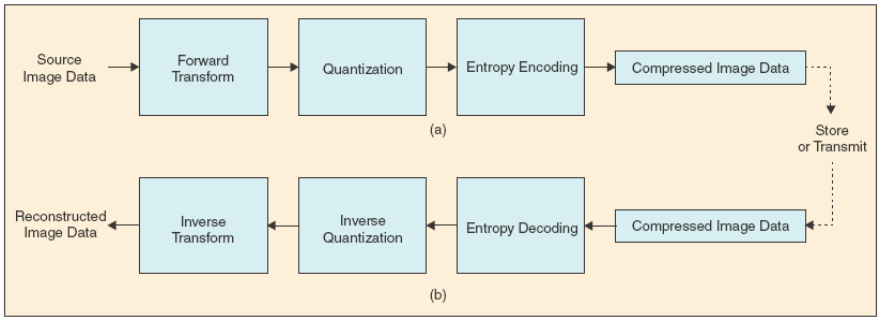


Figure 2.10: The JPEG2000 encoding and decoding process [66].

At the core of the JPEG2000 structure is a new wavelet based compression methodology that provides for a number of benefits over the Discrete Cosine Transformation (DCT) compression method, which was used in the JPEG lossy format. The DCT compresses an image into 8×8 blocks and places them consecutively in the file. In this compression process, the blocks are compressed individually, without reference to the adjoining blocks [60]. This results in "blockiness" associated with compressed JPEG files. With high levels of compression, only the most important information is used to convey the essentials of the image. However, much of the subtlety that makes for a pleasing, continuous image is lost.

In contrast, wavelet compression converts the image into a series of wavelets that can be stored more efficiently than pixel blocks. Although wavelets also have rough edges, they are able to render pictures better by eliminating the "blockiness" that is a common feature of DCT compression. Not only does this make for smoother color toning and clearer edges where there are sharp changes of color, it also gives smaller file sizes than a JPEG image with the same level of compression.

This wavelet compression is accomplished through the use of the JPEG2000 encoder [21], which is pictured in Figure 2.10. This is similar to every other transform based coding scheme. The transform is first applied on the source image data. The transform coefficients are then quantized and entropy coded, before forming the output. The decoder is just the reverse of the encoder. Unlike other coding schemes, JPEG2000 can be both lossy and lossless. This depends on the wavelet transform and the quantization applied.

The JPEG2000 standard works on image tiles. The source image is partitioned into rectangular non-overlapping blocks in a process called tiling. These tiles are compressed independently as though they were entirely independent images. All operations, including component mixing, wavelet transform, quantization, and entropy coding, are performed independently on each different tile. The nominal tile dimensions are powers of two, except for those on the boundaries of the image. Tiling is done to reduce memory requirements, and since each tile is reconstructed independently, they can be used to decode specific parts of the image, rather than the whole image. Each tile can be thought of as an array of integers in sign-magnitude representation. This array is then described in a number of bit planes. These bit planes are a sequence of binary arrays with one bit from each coefficient of the integer array. The first bit plane contains the most significant bit (MSB) of all the magnitudes. The second array contains the next MSB of all the magnitudes, continuing in the fashion until the final array, which consists of the least significant bits of all the magnitudes.

Before the forward discrete wavelet transform, or DWT, is applied to each tile, all image tiles are DC level shifted by subtracting the same quantity, such as the component depth, from each sample1. DC level shifting involves moving the image tile to a desired bit plane, and is also used for region of interest coding, which is explained later. This process is pictured in Figure 2.11.

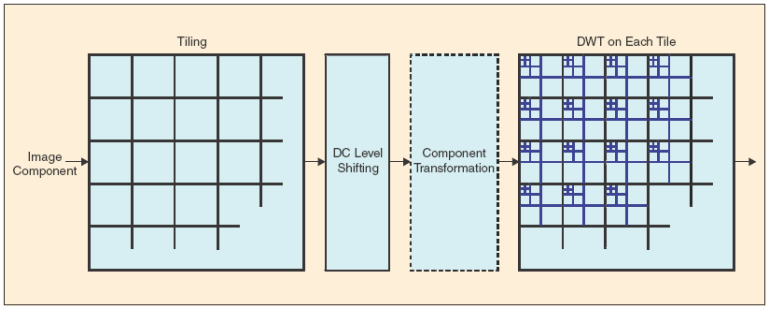


Figure 2.11: Tiling, DC-level shifting, color transformation (optional) and DWT for each image component [66].

Each tile component is then decomposed using the DWT into a series of decomposition levels which each contain a number of subbands. These subbands contain coefficients that describe the horizontal and vertical characteristics of the original tile component. All of the wavelet transforms employing the JPEG2000 compression method are fundamentally one-dimensional in nature [6]. Applying one-dimensional transforms in the horizontal and vertical directions forms two-dimensional transforms. This results in four smaller image blocks; one with low resolution, one with high verti-

cal resolution and low horizontal resolution, one with low vertical resolution and high horizontal resolution, and one with all high resolution. This process of applying the one-dimensional filters in both directions is then repeated a number of times on the low-resolution image block. This procedure is called dyadic decomposition and is pictured in Figure 2.12. An example of dyadic decomposition1 into subbands with the whole image treated as one tile is shown in Figure 2.13.

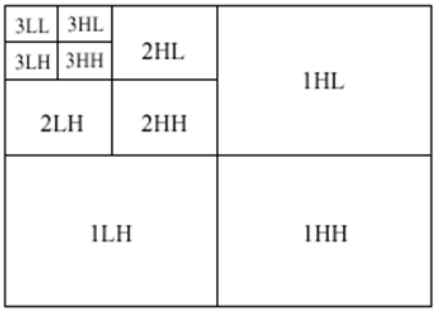


Figure 2.12: Dyadic decomposition.



Figure 2.13: Example of Dyadic decomposition.

To perform the forward DWT, a one-dimensional subband is decomposed into a set of low-pass samples and a set of high-pass samples. Low-pass samples represent a smaller low-resolution version of the original. The high-pass samples represent a smaller residual version of the original; this is needed for a perfect reconstruction of the original set from the low-pass set.

As mentioned earlier, both reversible integer-to-integer and nonreversible real-to-real wavelet transforms can be used. Since lossless compression requires that no data be lost due to rounding, a reversible wavelet transform that uses only rational filter coefficients is used for this type of compression. In contrast, lossy compression allows for some data to be lost in the compression process, and therefore nonreversible wavelet transforms with non-rational filter coefficients can be used. In order to handle filtering at signal boundaries, symmetric extension is used. Symmetric extension adds

a mirror image of the signal to the outside of the boundaries so that large errors are not introduced at the boundaries. The default irreversible transform is implemented by means of the bi-orthogonal Daubechies 9-tap/7-tap filter. The Daubechies wavelet family is one of the most important and widely used wavelet families. The analysis filter coefficients for the Daubechies 9-tap/7-tap filter [21], which are used for the dyadic decomposition.

After transformation, all coefficients are quantized. This is the process by which the coefficients are reduced in precision. Dividing the magnitude of each coefficient by a quantization step size and rounding down accomplishes this. These step sizes can be chosen in a way to achieve a given level of quality. This operation is lossy, unless the coefficients are integers as produced by the reversible integer 5/3 wavelet, in which case the quantization step size is essentially set to 1.0. In this case, no quantization is done and all of the coefficients remain unchanged.

Following quantization, each subband is subjected to a packet partition [46]. Each packet contains a successively improved resolution level on one tile. This way, the image is divided into first a low quality approximation of the original, and sequentially improves until it reaches its maximum quality level. Finally, code-blocks are obtained by dividing each packet partition location into regular non-overlapping rectangles. These code-blocks are the fundamental entities used for the purpose of entropy coding.

Entropy coding is performed independently on each code-block. This coding is carried out as context-dependant binary arithmetic coding of bit planes. This arithmetic coding is done through a process of scanning each bit plane in a series of three coding passes. The decision as to which pass a given bit is coded in is made based on the significance of that bit's location and the significance of the neighboring locations. A location is considered significant if a 1 has been coded for that location in the current or previous bit plane [46].

The first pass in a new bit plane is called the significance propagation pass. A bit is coded in this pass if its location is not significant, but at least one of its eight-connected neighbors is significant. The second pass is the magnitude refinement pass. In this pass, all bits from locations that became significant in a previous bit plane are coded. The third and final pass is the clean-up pass, which takes care of any bits not coded in the first two passes. After entropy coding, the image is ready to be stored as a compressed version of the original image.

One significant feature of JPEG2000 is the possibility of defining regions of in-

terest in an image. These regions of interest are coded with better quality than the rest of the image. This is done by scaling up, or DC shifting, the coefficients so that the bits associated with the regions of interest are placed in higher bit-planes. During the embedded coding process, these bits are then placed in the bit-stream before the part of the image that is not of interest. This way, the region of interest will be decoded before the rest of the image. Regardless of the scaling, a full decoding of the bit-stream results in a reconstruction of the whole image with the highest possible resolution. However, if the bit-stream is truncated, or the encoding process is terminated before the whole image is fully encoded, the region of interest will have a higher fidelity than the rest of the image.

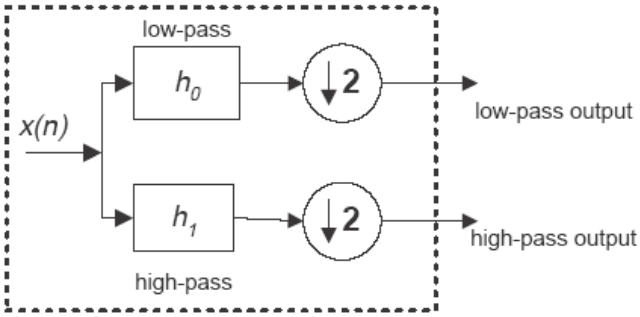


Figure 2.14: The DWT structure.

The lossless compression efficiency of the reversible JPEG2000, JPEG-LS, lossless JPEG (L-JPEG), and PNG is reported in Table 2.4. It is seen that JPEG2000 performs equivalently to JPEG-LS in the case of the natural images, with the added benefit of scalability. JPEG-LS, however, is advantageous in the case of the compound image. Taking into account that JPEG-LS is significantly less complex than JPEG2000, it is reasonable to use JPEG-LS for lossless compression. In such a case though, the generality of JPEG2000 is sacrificed.

Table 2.4: Comparison results of a few lossless image compression techniques.

	JPEG2000	JPEG-LS	L-JPEG	PNG
aerial2	1.47	1.51	1.43	1.48
bike	1.77	1.84	1.61	1.66
cafe	1.49	1.57	1.36	1.44
chart	2.60	2.82	2.00	2.41
cmpnd1	3.77	6.44	3.23	6.02
target	3.76	3.66	2.59	8.70
us	2.63	3.04	2.41	2.94
average	2.50	2.98	2.09	3.52

Overall, the JPEG2000 standard offers the richest set of features in a very efficient way and within a unified algorithm. However, this comes at a price of additional complexity in comparison to JPEG and JPEG−LS. This might be perceived as a disadvantage for some applications, as was the case with JPEG when it was first introduced.

2.2.4 Graphics Interchange Format (GIF)

The first widely used standard for lossless image compression was the Graphics Interchange Format (GIF) standard invented by Compuserve. It is based on Welch's popular extension of the LZ78 coding scheme. GIF uses a color palette, that contains a maximum of 256 entries. Each entry specifies one color using a maximum of 8 bits for each of red, green and blue. The color palette must be built prior to coding and is sent along with the compressed data. Note that images with more than 256 colors cannot be losslessly coded with GIF. The LZW coding is applied directly to the pixel data, therefore there is no mapping stage. Due to the inherently adaptive nature of LZW, it can be seen as combining the modelling and coding stages into one. LZW (and hence GIF) works well for computer generated images (especially icons) which have a lot of pixel sequences repeated exactly. However, for real pictures it performs less well. This is due to the noise, inherent in any form of image capture, breaking the repeating sequences of pixels that LZW depends on. Also, the limitation of 256 different pixel values became a problem as cheaper memory made 24 bit images more popular.

2.2.5 Portable Network Graphics (PNG)

Portable Network Graphics (PNG) [58] is a WWW Consortium for coding of still images which has been elaborated as a patent free replacement for GIF, while incorporating more features than this last one. It is based on a predictive scheme and entropy coding. The entropy coding uses the Deflate algorithm of the popular Zip file compression utility, which is based on LZ77 coupled with Huffman coding. PNG is capable of lossless compression only and supports gray scale, palettes color and true color, an optional alpha plane, interlacing and other features.

2.3 Miscellaneous Techniques Based on predictive methods

Predictive algorithm uses predictor function to guess the pixel intensities and then we calculate the prediction errors, i.e., differences between actual and predicted pixel intensities. Then, it encodes the sequence of prediction errors, which is called the residuum. To calculate the predictor for a specific pixel we usually use intensities of small number of already processed pixels neighboring it. Even using extremely simple predictors, such as one that predicts that pixel intensity is identical to the one in its left-hand side, results in a much better compression ratio, than without the prediction. For typical grey scale images, the pixel intensity distribution is close to uniform. Prediction error distribution is close to Laplacian, i.e., symmetrically exponential [6, 7, 8]. Therefore entropy of prediction errors is significantly smaller than entropy of pixel intensities, making prediction errors easier to compress.

2.3.1 Simple fast and Adaptive Lossless Image Compression Algorithm (SFALIC)

This method compresses continuous tone gray-scale images. The image is processed in a raster-scan mode. It uses 9 set predictors. The 8 mode predictors are the same with lossless JPEG as seen in Table 2.1. The last predictor mode is a bit more complex that actually returns an average of mode 4 and mode 7. Predictors are calculated using integer arithmetic.

The presented predictive and adaptive lossless image compression algorithm was designed to achieve high compression speed. The prediction errors obtained using simple linear predictor are encoded using codes adaptively selected from the modified Golomb-Rice code family. As opposed to the unmodified Golomb-Rice codes, this family limits the codeword length and allows coding of incompressible data without expansion. Code selection is performed using a simple data model based on the model known from FELICS algorithm. Since updating the data model, although fast as compared to many other modelling methods, is the most complex element of the algorithm, they apply the reduced model update frequency method that increases the compression speed by a couple of hundred percent at the cost of worsening the compression ratio. This method could probably be used for improving speed of other algorithms, in which data modelling is a considerable factor in the overall algorithm time complexity [69].

The main purpose of this method is reduced time complexity of others predictive methods such as FELICS, inconsequence less performance than previous methods, see Section 2.4.

2.3.2 Context based Adaptive Lossless Image Codec (CALIC)

Lossless `JPEG` and `JPEG-LS` are not the only coding that used predictor. There are other contenders that used more complex predictor such as Context based Adaptive Lossless Image Codec (CALIC). An optimal predictor is fined-tuned by adaptive adjustments of the prediction value. An optimal predictor would result in prediction error equal zero on average, but this is not really important. There is a technique called bias cancellation to detect and correct systematic bias in the predictor.

This idea of explicitly looking for edges in the image data was also used by Wu in [79]. He uses the local horizontal and vertical image gradients, called Gradient Adjusted Predictor (GAP), given by:

$$\begin{aligned} d_h &= |W - WW| + |N - NW| + |NE - N| \\ d_v &= |W - NW| + |N - NN| + |NE - NNE| \end{aligned} \tag{2.4}$$

to help predict x:

$$\begin{array}{ll}
\text{if } (d_v - d_h > 80) & \texttt{// sharp horizontal edge} \\
\bar{x} = W & \\
\text{else if } (d_v = d_h <= 80) & \texttt{// sharp vertical edge} \\
\bar{x} = N & \\
\text{else } \{ & \\
\bar{x} := (W + N)/2 + (NE - NW)/4 & \texttt{// assume smoothness first} \\
\text{if } (d_v - d_h > 32) & \texttt{// horizontal edge} \\
\quad \bar{x} = (\bar{x} + W)/2 & \\
\text{else if } (d_v - d_h > 8) & \texttt{// weak horizontal edge} \\
\quad \bar{x} = (3\bar{x} + W)/4 & \\
\text{else if } (d_v - d_h <= 32) & \texttt{// vertical edge} \\
\quad \bar{x} = (\bar{x} + N)/2 & \\
\text{else if } (d_v - d_h <= 8) & \texttt{// weak vertical edge} \\
\quad \bar{x} = (3\bar{x} + N)/4 & \\
\} &
\end{array} \tag{2.5}$$

By classifying edges as either strong, normal or weak, GAP does more modelling than MED. This extra modelling gives GAP better performance than MED, although

typically not by a large margin. The extra work also makes GAP more computationally expensive. The use of MED in JPEG−LS indicates that in terms of a joint complexity-performance judgment, MED has the upper hand.

GAP is used by CALIC which puts heavy emphasis on data modelling. It adapts the prediction according to the local gradients, thereby using a non-linear predictor which adapts to varying source statistics. The GAP classifies the gradient of the current pixel x according to the estimated gradients in the neighborhood, which is wider than that used in DPCM and choose the appropriate predictor according to the classification. The prediction error is context modelled and entropy coded.

Furthermore, for coding process, there are CALIC-H which used Huffman Coding and CALIC-A used Arithmetic Coding as the last stages. CALIC-A is relatively more complex because it is using arithmetic entropy coder, but it provides better compression ratios.

2.3.3 Fast, Efficient, Lossless Image Compression System (FELICS)

It is a very simple and fast lossless image compression algorithm by Howard and Vitter that also using Golomb or Golomb-Rice codes for entropy coding [35]. Preceding by raster-scan order, this method also uses predictor as modelling parts. It codes the new pixels using the intensities of two nearest neighborhood of pixel (P) that have already coded as shown in Figure 2.15. Nearest neighbors N_1 and N_2 are used to code the intensity of pixel P. The "range" used in making the in-range/out-of-range decision is [min$\{N_1, N_2\}$; max$\{N_1, N_2\}$], and the "context" used for modelling the probability distribution is $N_1 - N_2$. For points in the center of the image (shaded), the predicting pixels are the pixels immediately to the left of and above P. Along the edges adjustments must be made, but otherwise the calculations are the same. However, the along top and left edges (pixel above and left of the new pixel) are needed to obtain the smaller neighboring value L and the larger value H. The difference of these two values will define Δ as the prediction context of P to code parameter selection.

The idea of the coding algorithm is to use one bit to indicate whether P is in the range from L to H, an additional bit if necessary to indicate whether it is above or below the range, and a few bits, using a simple prefix code, to specify the exact value. This method leads to good compression for two reasons: the two nearest neighbors provide a good context for prediction, and the image model implied by the algorithm closely matches the distributions found in real images. In addition, the method is very

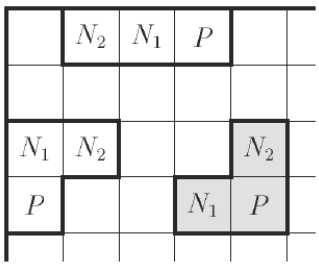


Figure 2.15: Example of FELICS prediction process [35].

fast, since it uses only single bits and simple prefix codes.

The intensities are generally distributed as shown in Figure 2.16. P lies in the range $[L, H]$ about half the time, requiring one bit to encode, and when P is out of range, the above-range/below-range decision is symmetrical, so another one-bit code is appropriate. In-range values of P are almost uniformly distributed, with a slight crest near the middle of the range, so an adjusted binary encoding gives nearly optimal compression when P is in range. The probability of out-of-range values falls off sharply, so when P is out of range it is reasonable to use exponential prefix codes, i.e., Golomb codes or the simpler Rice codes, to indicate how far out of range the value is. This distribution clearly differs from the Laplace distribution commonly assumed in predictive image coding, but it is consistent with the error modelling treatment.

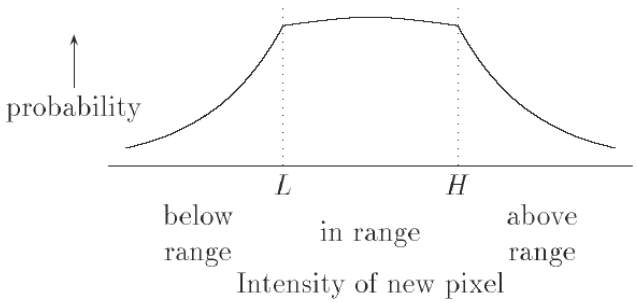


Figure 2.16: Schematic probability distribution of intensity values for a given context $\Delta = H - L$ [35].

2.3.4 TMW

Most of lossless image compression are basically used a few of well known methods, such as predictive coding, context based selection of predictor coefficients and a fading-memory model for prediction error distributions.

Meyer and Tischer present another lossless method for greyscale images. It consists of two stages: Image analysis and Coding stage as seen in Figure 2.17. This method uses several concepts such as the extraction of global image information, the use of multiple predictors with blending in the probability domain and the use of unquantized predicted values.

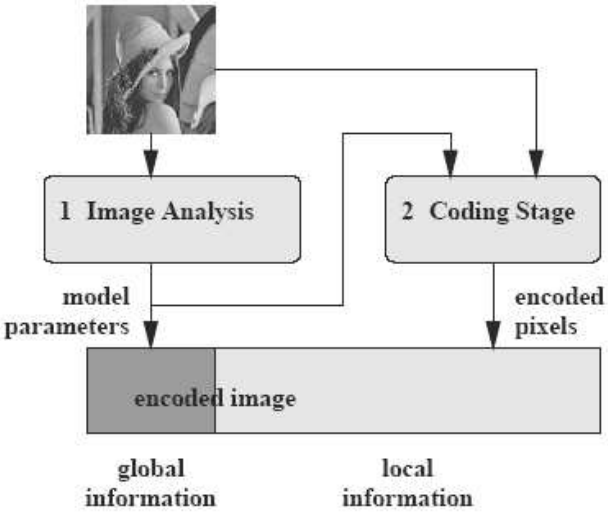


Figure 2.17: TMW block diagram [48].

When applied to image compression, the first message part can be seen as describing characteristics of the image as global information. The second part of the message contains purely local information about the values of individual pixels. Ideally, the first part would capture the "essence" or "meaning" of the image, while the second part would only contain information about the particular values of noise for each pixel. While TMW does not yet achieve this ideal goal, it does constitute a significant step in this direction. The last stage which here is coding process, TMW uses Arithmetic Coding to do entropy coding.

The complexity of the model used in TMW makes it unsuitable for most practical uses, the results achieved both for lossless and near lossless compression prove wrong recent suggestions that image compression has reached its limits. It shows that significant redundancies exist in typical images that are not exploited by current practical methods, and that exploitation of those redundancies is possible [49].

2.3.5 Lossy to Lossless Image Compression based on Reversible Integer DCT

Discrete Codine Transform (DCT) is used by JPEG standard for lossy image compression, and this lossy standard can not be lossless because of the round of DCT process. However, some authors have been proposed reversible integer DCT for lossless mode. It employs an encoding technique different from JPEG. It uses the framework of JPEG, and just converts DCT and color transform to be integer reversible. Integer DCT is implemented by factoring the float DCT matrix into a series of elementary reversible matrices and each of them is directly integer reversible. The integer DCT integrates lossy and lossless schemes nicely, and it supports both lossy and lossless compression by the same method. Experiments show that the performance of JPEG with the integer reversible DCT is very close to that of the original standard JPEG for lossy image coding, and more importantly, with integer DCT, it can compress images losslessly.

2.3.6 Specifics lossless image compression methods

In order to achieve competitive compression, some authors create a specific methods for certain applications. Specific algorithm generally performs better than do general-purpose image-data compression algorithms. It exploits the nature of the image to suit the compression methods.

The first example is CCSDS SZIP. It is a standard of the Consultative Committee for Space Data Systems used by space agencies for compressing scientific data transmitted from satellites and other space instruments [5]. CCSDS SZIP is a very fast predictive compression algorithm based on the extended-Rice algorithm, it uses Golomb-Rice codes for entropy coding, and primarily was developed by Rice. CCSDS SZIP does not employ an adaptive data model. The sequence of prediction errors is divided into blocks. Each block is compressed using a two-pass algorithm. In the first pass, it determines the best coding method for the whole block. In the second pass, the marker output of the selected coding method as a side information along with prediction errors encoded using this method. The coding methods include: Golomb-Rice codes of a chosen rank; unary code for transformed pairs of prediction errors; fixed length natural binary code if the block is found to be incompressible; signaling to the decoder empty block if all prediction errors are zeroes.

Other specific lossless image compression is created for A Large Ion Collider Ex-

periment (ALICE) at CERN. It is a lossless image compression for scientific imaging. The experiment studied the collisions between heavy ions energies. The collisions took place at the center of a set of several detectors, which designed to track and identify the particles produced. The input data is generated by the Time Projection Chamber (TPC) detector of the ALICE experiment at CERN. The algorithm is based on a lossless source code modelling technique (the original TPC signal information) can be reconstructed without errors at the decompression stage. The source model exploits the temporal correlation that is present in the TPC data to reduce the entropy of the source.

This technique is very specific for scientific imaging of ALICE project. It is mainly based on an appropriate probability model for each data field. More precisely, specific probability models for each sample in a bunch are developed. Such models intend to capture both temporal correlation among samples and the characteristic shape of TPC electrical pulses. For what concerns the time information, i.e. position of the bunches they are not represented as an absolute value, but they are differentially coded using the number of zero samples preceding the bunch. Finally, the bunch length is directly entropy coded.

The SPIHT-compression technique (Set Partitioning in Hierarchical Trees) [10] transforms the image to a multi-resolution representation by the wavelet transform or, in the case of lossless compression, by the S transform. This transformation is similar to the sub-band decomposition, but uses only integer operations. The S transform scans the columns of the image and calculates for each successive pair of the pixels the average and the difference. The averages are stored at the upper part of the image (L), and the differences are stored at the lower part (H). The same is repeated for the columns of l and h parts giving LL, LH, HL and LL images. At the next level the ll block will be processed in the same way. This gives a multi resolution pyramid of the image with reduced variance.

Other example of lossless application that used specific algorithm is in astrophysical images. Astronomers often insist that they can accept only lossless compression [40]. Lastri et al. proposed an advanced lossless DPCM scheme. This technique is used in lossless-JPEG. They also proposed near-lossless compression based on the astronomers needs. The results are compared with the existing lossless compression standard such as JPEG-LS and JPEG2000. Otherwise, the proposed methods obtains better compression for lossless and near-lossless compression [40].

2.4 Compression performances

There are many applications for lossless images such as digital photography, museums or art, publishing, scientific imaging, remote sensing, aerial survey, astronomy, GIS, medical imaging, microscopy, machine vision, quality control, parts inspection, defect tracing, CCTV and security, fingerprint or Forensic and remote operation. Each of image has its own characteristics. Some authors have been developed such a specifics methods to exploit image nature to obtain better compression performance. Some authors also have been analyzed a few of lossless image compression for different type of image [59, 68, 61, 77, 56]. Each technique has its advantages and disadvantages, for example Santa Crus in [59] gives a summary of few of image compression techniques as seen in Table 2.5.

Table 2.5: Functionality matrix. A "+" indicates that it is supported, the more "+" the more efficiently or better it is supported. A "-" indicates that it is not supported [60].

	JPEG2000	JPEG-LS	JPEG	PNG
lossless compression performance	+++	++++	+	+++
lossy compression performance	+++++	+	+++	-
progressive bitstreams	+++++	-	++	+
Region of Interest (ROI) coding	+++	-	-	-
arbitrary shaped objects	-	-	-	-
random access	++	-	-	-
low complexity	++	+++++	+++++	+++
error resilience	+++	++	++	+
non-iterative rate control	+++	-	-	-
genericity	+++	+++	++	+++

Starosolski in [68] presents lossless image compression methods such as CALIC, FELIC, JPEG-LS, JPEG2000, PNG, BZIP2, etc for several gray scale images. For natural images the universal compression algorithm BZIP2 obtains ratios worse than CALIC by about 10%. For medical MR images, that all are of 16-bit nominal depth, and that none of them actually contains pixels of more than 2000 levels, ratio of BZIP2 is better than CALIC's by over 40%. For Computed Radiography (CR) and Ultrasound (US) images the best ratios were obtained by CALIC, however, the compression ratio deterioration of MR and CT images is so high, that the average compression ratio of the whole medical group, and of the whole normal group, is best in case of the BZIP2. These are interesting results to dig in Burrows Wheeler Transform (BWT) in lossless medical images. But, evaluation is performed on only few images, it contains all only 84 images for groups natural and medical. Therefore, it

Table 2.6: Compression performances of natural and medical images for different compression techniques [68].

Image group	Lossless JPEG	JPEG -LS	JPEG 2000	PNG	SZIP	CALIC-A	CALIC-H	SFALIC	BZIP2
Natural	8.367	7.687	7.916	10.045	8.432	7.617	7.662	7.953	9.165
Big	7.668	7.083	7.185	9.451	7.773	6.962	7.059	7.274	8.36
Medium	8.446	7.71	7.955	10.079	8.403	7.623	7.699	8.009	9.113
Small	8.986	8.269	8.608	10.605	9.121	8.267	8.227	8.576	10.023
16bpp	12.327	11.776	11.998	13.836	12.459	11.748	11.622	11.867	14.059
12bpp	8.321	7.571	7.823	11.542	8.407	7.491	7.565	7.869	8.877
8bpp	4.451	3.715	3.927	4.756	4.431	3.613	3.797	4.123	4.56
Medical	7.427	6.734	6.891	8.073	7.396	6.651	6.761	7.165	5.181
cr	7.023	6.343	6.394	8.944	6.883	6.229	6.324	6.662	6.479
ct	8.509	7.838	8.044	9.381	8.806	7.759	7.84	8.266	5.577
mr	10.451	10.009	10.024	10.35	10.599	9.975	9.895	10.235	5.929
us	3.724	2.748	3.1	3.616	3.298	2.641	2.985	3.497	2.739
normal	7.83	7.143	7.33	8.918	7.84	7.065	7.147	7.503	6.889

needs further evaluation. More detail about these results can be seen in Table 2.6.

2.5 Summary

This chapter describes state of the arts of lossless image compression. We discussed the standards and non-standards techniques, then pointed out that BWT which is used in BZIP2 as text compression is also an interesting method to be applied in lossless image compression. Nevertheless, these previous results need further analysis to evaluate BWT implementation in lossless image compression.

Chapter 3

Compression Scheme based on Burrows-Wheeler Transform

Contents

3.1 Introduction 40

3.2 BWT: What? Why? Where? 40

3.3 Burrows Wheeler Transform 42

3.4 Global Structure Transform 46

3.4.1 Move-To-Front (MTF) and its variants 47

3.4.2 Frequency Counting Methods 49

3.5 Data Coding 52

3.5.1 Run-level coding 52

3.5.2 Statistical Coding 54

3.6 BWT success : Text Compression 58

3.6.1 `BZIP2` 58

3.7 Innovative Works: Image Compression 59

3.7.1 Lossy 59

3.7.2 Lossless 62

3.7.3 Nearly Lossless 63

3.8 Summary 64

3.1 Introduction

In this chapter we describe Burrows Wheeler Compression Algorithm (BWCA) that used BWT as its main transform. We present pre- and post-processing dedicated to BWT. Furthermore, BWCA implementation in text and image application are detailed. Different existing chains are presented and the compression results are discussed

3.2 BWT: What? Why? Where?

Data compression process can be divided in 3 parts, as shown in Figure 1.1. Each part has been created and developed to improve compression performances. As stated before, the very common transform for the first part of image compression scheme is DCT or DWT. Here, we presented other transform that was originally used to text compression, that is, Burrows Wheeler Transform (BWT).

BWT was introduced by Burrows and Wheeler in 1994 [18] as a transform in data compression algorithm. It is based on block sorting technique and was created for text compression software such as `BZIP2`, after then it is also applied to other fields such as image [22, 13, 41, 78, 72] and DNA files [8, 55].

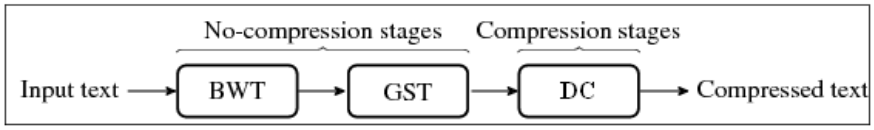


Figure 3.1: Original scheme of BWCA method.

In general, a typical BWT method compression consists of 3 stages as shown in Figure 3.1 [18], where:

- BWT is the Burrows Wheeler Transform itself, that tends to group similar characters together,
- GST is the Global Structure Transform, that transforms the local structure redundancy of the output of BWT to global redundancy using a list of updated table. It produces sequences of contiguous zeros,
- DC is an Data Coding,

and from now on we refer this scheme as Burrows-Wheeler Compression Algorithm (BWCA).

BWT as the main transforms of this method is a particular permutation of input text which is usually followed by GST and Data Coding (DC). Commonly, DC in turn, as the last part of this scheme, consists of 2 codings:

- a Run-level coding, followed by
- a statistical based compression, such as a dynamic Huffman or Arithmetic Coding (AC).

The first and second stages of BWCA chain can be seen as a performance booster for statistical compressors.

(a)	BWT input	:	a b r a c a d a b r a a b r a c a d a b r a
(b)	BWT input (in HEXA)	:	61 62 72 61 63 61 64 61 62 72 61 61 62 72 61 63 61 64 61 62 72 61
(c)	BWT output	:	72 72 64 61 61 64 72 72 63 63 61 61 61 61 61 61 61 61 62 62 62 62
(d)	GST output	:	72 00 65 00 63 00 02 00 65 00 02 00 00 00 00 00 00 00 65 00 00 00
(e)	RLE0 output	:	74 00 67 00 65 00 02 00 67 00 02 00 00 00 67 00 00
(f)	EC output	:	00 0D 01 8D B3 FF 81 00 72 A8 E8 2B

Figure 3.2: Transformed data of the input string `abracadabraabracadabra` by the different stages [4].

Figure 3.2 elucidates BWT chain example for input data `abracadabraabracadabra` based on BWT original method in Figure 3.1. The input data of the BWT stage is shown in Figure 3.2(a) and (b); Figure 3.2(b) represent the input data in hexadecimal.

In the input data, rarely two consecutive symbols are the same (here only a unique occurrence of `aa` appears) but, the BWT output contains many sequences of repeated symbols and has a local structure, i.e., symbols with a similar context form small fragments as seen in Figure 3.2(c) and so local redundancy emerges. Then, the GST stage transforms the local structure of the BWT output to a global structure by using a ranking scheme according to the last recently used symbols and produces sequences of contiguous zeros which are displayed in Figure 3.2(d). There are some of GST algorithm that have been proposed by some authors, and in this example a Move-To-Front (MTF) algorithm is used. More detail about MTF and its variants will be discussed in Section 3.4.

Furthermore, GST stage is followed by EC. As stated before, original scheme of BWT used RLE (to remove symbol redundancies) and statistical coding (to maximize the compression process). The data in Figure 3.2 (e) and (f) are obtained by using Run Length Encoding zero (RLE0) of Wheelers and Arithmetic Coding (AC). The EC

input in Figure 3.2 (f) does not include its header. Detail discussion about this stage will be presented in Section 3.5.2.

More detail about each stage of Figure 3.2 will be discussed in the Section 3.3.

3.3 Burrows Wheeler Transform

Forward of BWT

BWT is one of data compression transforms which does not compress the data, but it tends to regroups similar symbols and so, used as a preprocessing step for data compression. Figure 3.3(a) and (b) show how the forward BWT works. In this example, we use the same example of Figure 3.2. BWT makes all possible cyclic rotations of the input data as seen in Figure 3.3(a). Then as explanation in [7], it sorts the rotations in ascending order. The obtained matrix is represented in Figure 3.3(b). The position of original data is named primary index (equals to 4 in our matrix). This information is required for the reconstruction process. The last column (L) of this matrix and the primary index are the BWT output. As it can seen in this example, from the input data `abracadabraabracadabra` or `61 62 72 61 63 61 64 61 62 72 61 61 62 72 61 63 61 64 61 62 72 61` in hexadecimal gives the sequence of BWT output `72 72 64 64 61 61 72 72 63 63 61 61 61 61 61 61 61 61 62 62 62 62` which tends to groups similar input data together. This small example shows the peculiar BWT output. There is a series of eight symbols of `61`, while only two consecutive symbols in the input.

As seen above, BWT is based on a sorting algorithm. Therefore, sorting becomes the most important issue in BWT process. Sorting process is computationally intensive and its performance becomes worst for homogen input data. There are several methods to improve the performance of sorting process, but they do not influence BWT results. Burrows and Wheeler themselves suggested suffix tree to improve sorting process [18]. Other authors suggest suffix array or their own sorting algorithm [39, 45, 29].

Figure 3.4 shows the relationship between BWT and suffix array for the same input in Figure 3.3. We consider the input data is Im, then n is the length of Im, and so in this example $n = 22$. Suffix array (SA) is constructed as followed. Figure 3.4(a) consists of BWT input and its given suffix, then the BWT process sorts the suffixes data in ascending order as seen in Figure 3.4(b) to get the suffixes array (SA). Here, we can see the correlations between sorted rotations (the conventional of forward BWT in Figure 3.3(b)) and sorted suffix of Figure 3.4(b). These matrixes are similar. Figure 3.3(b)

Position	a	b	r	a	c	a	d	a	b	r	a	a	b	r	a	c	a	d	a	b	r	a
1	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
2	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61
3	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62
4	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72
5	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61
6	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63
7	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61
8	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64
9	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61
10	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62
11	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72
12	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
13	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61
14	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62
15	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72
16	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61
17	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63
18	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61
19	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64
20	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61
21	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62
22	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72

(a) Rotated BWT input `abracadabraabracadabra` in HEX.

Position	**F**																					**L**
11	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72
22	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72
8	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64
1	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
12	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
19	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64
4	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72
15	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72
6	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63
17	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63
9	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61
20	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61
2	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61
13	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61
5	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61
16	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61
7	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61
18	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61
10	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62
21	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62
3	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62
14	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	61	62

(b) Sorted BWT rotations.

Figure 3.3: The BWT forward.

will be exactly the same with Figure 3.4(b) if the added symbols to create the rotations matrix of Figure 3.3(a) are ignored. For example the first line of Figure 3.3(b) was placed in line 11 of the rotated original data, so the rotations matrix for this line starts from the symbol 11^{th} to 22^{th} then added the symbols 1^{st} to 10^{th} to complete the sorted rotations matrix. If the added symbols are omitted, this first line is equal to the first

ID	Suffixes																				
1	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
2		62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
3			72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
4				61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
5					63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
6						61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
7							64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
8								61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
9									62	72	61	61	62	72	61	63	61	64	61	62	72	61
10										72	61	61	62	72	61	63	61	64	61	62	72	61
11											61	61	62	72	61	63	61	64	61	62	72	61
12												61	62	72	61	63	61	64	61	62	72	61
13													62	72	61	63	61	64	61	62	72	61
14														72	61	63	61	64	61	62	72	61
15															61	63	61	64	61	62	72	61
16																63	61	64	61	62	72	61
17																	61	64	61	62	72	61
18																		64	61	62	72	61
19																			61	62	72	61
20																				62	72	61
21																					72	61
22																						61

(a) Original data and its suffix.

SA	Sorted suffixes																				
11	61	61	62	72	61	63	61	64	61	62	72	61									
22	61																				
8	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61						
1	61	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
12	61	62	72	61	63	61	64	61	62	72	61										
19	61	62	72	61																	
4	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61		
15	61	63	61	64	61	62	72	61													
6	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61				
17	61	64	61	62	72	61															
9	62	72	61	61	62	72	61	63	61	64	61	62	72	61							
20	62	72	61																		
2	62	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61
13	62	72	61	63	61	64	61	62	72	61											
5	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61			
16	63	61	64	61	62	72	61														
7	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61					
18	64	61	62	72	61																
10	72	61	61	62	72	61	63	61	64	61	62	72	61								
21	72	61																			
3	72	61	63	61	64	61	62	72	61	61	62	72	61	63	61	64	61	62	72	61	
14	72	61	63	61	64	61	62	72	61												

(b) Suffixes array and sorted suffixes.

Figure 3.4: Relationship between BWT and suffix arrays.

line of sorted suffixes (first line in the third column of Figure 3.4).

The obtained suffix array (SA) can compute BWT output using this formula :

$$L(i) = \begin{cases} Im[SA[i]-1], & \text{if } SA[i] \neq 1 \\ Im[n], & \text{otherwise.} \end{cases} \tag{3.1}$$

Hence, it does not need to create the sorted rotations matrix to obtain BWT output (Figure 3.3(a) and (b)). Nevertheless, suffix sorting algorithms that run in linear in the worst case is still open. Figure 3.5 shows a few different methods for BWT computation [25]. We use the BWT of Yuta Mori that is based on SA-IS algorithm to construct the BWT output [52]. It reduces processing time and decrease memory requirements.

Method	complexity		
	time		space
	Worst-case	Avg-case	Avg-case
Ukkonen's suffix tree construction	O(n)	$O(n)$	-
McCreight's suffix tree construction	$O(n)$	$O(n)$	-
Kurtz-Balkenhol's suffix tree construction	$O(n)$	$O(n)$	$10n$
Farach's suffix tree construction	$O(n \log n)$	$O(n \log n)$	-
Manber-Myers's suffix array construction	$O(n \log n)$	$O(n \log n)$	$8n$
Sadakane's suffix array construction	$O(n \log n)$	$O(n \log n)$	$9n$
Larson-Sadakane's suffix array construction	$O(n \log n)$	$O(n \log n)$	$8n$
Itoh-Tanaka's suffix array construction	$> O(n \log n)$	$O(n \log n)$	$5n$
Nong's SA-IS	$O(n \log n)$	$O(n \log n)$	$6n$
Burrows-Wheeler's sorting	$O(n^2 \log n)$	-	-
Bentley-Sedgewick's sorting	$O(n^2)$	$O(n \log n)$	$5n + stack$
Sedward's sorting	$O(n^2 \log n)$	-	-

Figure 3.5: Different sorting algorithms used for BWT.

Reverse of BWT

The reverse BWT [28] is just principally another permutation of the original data. Figure 3.6 shows how this process works. The second column of this figure is the BWT output of the Figure 3.3. The third column (called here context) is obtained from the second one by sorting its elements in ascending order. The link in the fourth column refers to the position of the context in the input (the second column). For repeated symbols the i^{th} occurrence of a context symbol corresponds to the i^{th} occurrence of the same symbol in the data input. The process is started from position 4 (the primary index) where the first value of the original data is placed. It refers to the context 61 (as

Position	input	context	link
1	**72**	**61**	4
2	**72**	**61**	5
3	**64**	**61**	11
→ 4	**61**	**61**	12 ←
5	**61**	**61**	13
6	**64**	**61**	14
7	**72**	**61**	15
8	**72**	**61**	16
9	**63**	**61**	17
10	**63**	**61**	18
11	**61**	**62**	19
12	**61**	**62**	20
13	**61**	**62**	21
14	**61**	**62**	22
15	**61**	**63**	9
16	**61**	**63**	10
17	**61**	**64**	3
18	**61**	**64**	6
19	**62**	**72**	1
20	**62**	**72**	2
21	**62**	**72**	7
22	**62**	**72**	8

Figure 3.6: The BWT inverse transform.

the first symbol of original data) and also refers to link 12, as a clue of next position to the second of original data. So, the next position is 12 which refers to 62 and gives the next link to the next output. Therefore each step gives the permuted pixel value as output and will process the whole file, because of the cyclic rotations.

The reverse of BWT, that is reconstructing BWT output, is faster than BWT process. This is one of BWT advantages, since normally the decompression process is done more frequent than the compression.

3.4 Global Structure Transform

GST is the Global Structure Transform that is used as second stage in BWT original methods, see Figure 3.1. It converts the output of BWT local structure redundancy to a global redundancy using a ranking list. The interest of this transform is two-fold: runs of similar symbols are changed into runs of zeros and the grey level image distribution is modified in the sense that the probability of lower input values increase and simultaneously the probability of the higher values decrease. Therefore the Entropy Coding is more efficient on the GST output. There are many variants of GST that can be classified in two parts based on its alteration in subsection 3.4.1 and subsection 3.4.2.

3.4.1 Move-To-Front (MTF) and its variants

Move-To-Front (MTF)

MTF maintains a list of all possible symbols (MTF list). The list order is modified during the process. The list can be considered as a stack and used to obtain the output. Each input value is coded with its rank in the list. This stack is then updated: the input value is push at the top of the list. Therefore, the rank of this input symbol becomes zero. Consequently, a run of N identical symbols is then coded with symbol followed by $N - 1$ zeros. Figure 3.7 shows the forward of MTF transforms.

MTF list	a	r	r	a	d	a	d	r	r	c	a	a	a	a	a	a	a	b	b	b	b
	b	a	a	r	a	d	a	d	d	r	c	c	c	c	c	c	c	a	a	a	a
	c	b	b	b	r	r	r	a	a	d	r	r	r	r	r	r	r	c	c	c	c
	d	c	c	c	b	b	b	b	b	a	d	d	d	d	d	d	d	r	r	r	r
	r	d	d	d	c	c	c	c	c	b	b	b	b	b	b	b	b	d	d	d	d
Input	r	r	a	d	a	d	r	r	c	c	a	a	a	a	a	a	a	b	b	b	b
Output	4	0	1	4	1	1	2	0	4	0	3	0	0	0	0	0	0	4	0	0	0

Figure 3.7: Forward of Move-To-Front.

The above example presents the output of BWT in Figure 3.3 in ASCII code. Normally, we use 256 symbols in MTF list, merely in Figure 3.7, since there are 5 input symbols, therefore the presented MTF list is 5 to show the simple scheme assign of MTF code. The first output 4 is obtained since the first input symbol is placed in the fourth placed of the list, then the list is updated by moving the input symbol in the first line. The second output is 0 because the input symbol is placed in front of the list. Once a symbol is being processed the encoder updated its position in the MTF list. This simple example shows that the runs of similar symbols of input stream become runs of *0*, which make it more appropriate for entropy coding (e.g., Huffman or Arithmetic Coding).

The decoding is quite simple, and begins with the same table. It is enough to output the symbol corresponding to the index and update the list passing this symbol in the first position. The list evolves exactly as during the coding process.

The MTF output is dominated by zeros with many occurring in runs as shown in Table 3.1, but there is a drawbacks of this transform. The most common symbol in an alphabet is special. There are some symbols that have probability greater than 0.5, otherwise others may have a probability of occurrence as low as zero [20]. Therefore, moving all the input symbol in front of the list should be avoided as that inversion is potentially more expensive than any other, it caused the most common symbol moved away from the front of the list.

File name	0	1	2	3	4	5	6	7	8	9	10
bib	83.39	4.29	1.98	1.45	1.12	0.92	0.80	0.72	0.66	0.57	0.45
book1	74.88	7.68	3.96	2.64	1.90	1.46	1.18	0.99	0.84	0.73	0.63
book2	80.41	6.28	2.96	1.88	1.38	1.05	0.86	0.73	0.61	0.52	0.45
geo	67.88	3.02	2.39	1.84	1.11	0.70	0.44	0.34	0.32	0.26	0.25
news	78.97	5.12	2.75	1.86	1.41	1.13	0.92	0.78	0.67	0.59	0.52
obj1	75.30	2.91	1.63	1.19	0.94	0.76	0.68	0.54	0.55	0.45	0.40
obj2	84.03	3.02	1.55	1.04	0.78	0.61	0.48	0.41	0.35	0.29	0.28
paper1	79.17	5.64	2.73	1.84	1.41	1.15	0.91	0.78	0.70	0.64	0.56
paper2	77.68	6.14	3.21	2.22	1.64	1.28	1.07	0.94	0.77	0.71	0.58
pic	93.70	1.44	0.66	0.45	0.35	0.30	0.26	0.23	0.21	0.18	0.17
progc	80.17	5.59	2.34	1.57	1.18	0.91	0.74	0.63	0.57	0.48	0.43
progl	86.43	4.33	1.95	1.12	0.82	0.64	0.52	0.48	0.36	0.32	0.26
progp	87.01	4.34	1.78	1.06	0.70	0.60	0.44	0.38	0.35	0.29	0.23
trans	89.62	2.91	1.29	0.81	0.64	0.49	0.42	0.39	0.30	0.29	0.21
Average	81.33	4.48	2.23	1.50	1.10	0.86	0.69	0.60	0.52	0.45	0.39

Table 3.1: Percentage share of the index frequencies of MTF output.

Move One From Front (M1FF) and Move One From Front 2 (M1FF2)

Balkenhol proposed the modification of MTF called Move One From Front (M1FF) [15]. This algorithm changes the works of the list. It avoid to move the common symbol away from the front list. It may mistakenly move the most common symbol away from the front list but does so less often than MTF. The input symbol from the second position in the list is moved to the first position; meanwhile the input from higher positions is moved to the second position. Figure 3.8 shows how this transform works.

	a	a	r	a	a	d	d	r	r	c	c	a	a	a	a	a	a	a	b	b	b
	b	r	a	r	d	a	r	d	c	r	a	c	c	c	c	c	c	c	b	a	a	a
MTF-1 list	c	b	b	b	r	r	a	a	d	d	r	r	r	r	r	r	r	r	c	c	c	c
	d	c	c	c	b	b	b	b	a	a	d	d	d	d	d	d	d	d	r	r	r	r
	r	d	d	d	c	c	c	c	b	b	b	b	b	b	b	b	b	b	d	d	d	d
Input	r	r	a	d	a	d	r	r	c	c	a	a	a	a	a	a	a	a	b	b	b	b
Output	4	1	1	4	0	1	2	1	4	1	3	1	0	0	0	0	0	0	4	1	0	0

Figure 3.8: Forward of Move One From Front.

Furthermore, Balkenhol gives a modification of M1FF called Move One From Front Two (M1FF2). The symbols from the second position is moved to the first position of the M1FF2 list only when the previous transformed symbol was at the first position.

Both of M1FF and M1FF2 are better than MTF [20] in text compression. But neither M1FF nor M1FF2 is easy to analyze. For M1FF, a symbol can become lodged in the first position

Time Stamp (TS) and Best x of $2x - 1$

This algorithm was developed by Albers [9]. The deterministic version of this algorithm is TimeStamp(0) or TS(0). While MTF use 256 symbols in its list, this transform uses a double length list. So the list contains 512 symbols and each symbol occurs twice. When the input data is processed, the position of an item is one plus the number of double symbols in front of that input symbol. Then the list is updated by moving the second symbol to the front. This method is also called "Best 2 of 3" algorithm based on the Chapin's algorithm called "Best x of $2x - 1$" algorithm [19], because TS (0) uses 2×256 symbols in the list. Therefore, the list of a "Best x of $2x - 1$" algorithm contains $x \times 256$ symbols. Here, we have tested a "Best x of $2x - 1$" algorithm for $x = 3$, 5, and 6 (called $Bx3$, $Bx5$ and $Bx6$, respectively for short) .

Time Stamp	a	r	r	a	d	a	d	r	r	c	c	a	a	a	a	a	a	b	b	b	b
	b	a	r	r	a	d	a	d	r	r	c	c	a	a	a	a	a	a	a	b	b	b	b
	c	b	a	r	r	a	d	a	d	r	r	c	c	c	c	c	c	c	a	a	a	a
	d	c	b	a	r	r	a	d	a	d	r	r	c	c	c	c	c	c	c	a	a	a
	r	d	c	b	a	r	r	a	d	a	d	r	r	r	r	r	r	r	c	c	c	c
	a	r	d	c	b	b	r	r	a	d	a	d	r	r	r	r	r	r	r	c	c	c
	b	a	a	d	c	c	b	b	b	a	d	a	d	d	d	d	d	d	r	r	r	r
	c	b	b	b	d	d	c	c	c	b	a	d	d	d	d	d	d	d	d	r	r	r
	d	c	c	c	b	b	b	b	b	c	b	b	b	b	b	b	b	b	d	d	d	d
	r	d	d	d	c	c	c	c	c	b	b	b	b	b	b	b	b	b	b	d	d	d
Input	r	r	a	d	a	d	r	r	c	c	a	a	a	a	a	a	a	b	b	b	b
Output	5	1	2	5	2	3	3	3	5	5	4	3	0	0	0	0	0	0	5	5	0	0

Figure 3.9: Forward of Time Stamp or Best 2 of 3 Algorithm.

3.4.2 Frequency Counting Methods

Frequency counting methods improve MTF algorithms by basing the ranking of symbols on their frequencies. There are several proposed methods that use frequency counting technique, such as Inversion Frequency and Weighted Frequency Count (WFC). It defines a function based on symbol frequencies and other technique also uses the distance to the last occurrence of each symbol within a sliding window.

Inversion Frequencies (IF)

Several GST stages have been unveiled since the birth of the BWCA in 1994. Their purpose is to produce an output sequence which is more compressible by the data coding stage than the output sequence of the original MTF stage. One of these MTF replacements is the algorithm from Arnavut and Magliveras [11], which they named Inversion Frequencies (IF). This technique is not a List Update Algorithm (LUA). It

encodes a BWT output as an inversion of a permutation of a $multiset$. Each element of a $multiset$ is paired with a frequency that indicates the number of element occurrences. For example, for the multiset permutation M = [r, r, a, d, a, d, r, r, c, c, a, a, a, a, a, a, a, b, b, b, b] as the alphabet output of BWT in Figure 3.2. We have S = (a, b, c, d, r). Arnavut and Magliveras in [11] define the *inversion frequency* vector $D = D_k$ for M as follows:

1. $D_0 =<>$.
2. $D_i = D_{i-1} \odot T_i$ with $T_i =< x_1, x_2, ...x_{f_i} >$ where
   - the symbol "$\odot$" denotes *catenation* of data strings.
   - x_1 = position of the first occurrence of i in M.
   - and for $j > 1$, x_j = number of elements y in M, $y > i$ occurring between the $(j-1)^{st}$ and j^{th} occurrence of i in M.

A brief description of IF, with an above example is given in Table 3.2. The IF output is D = D_4=<2, 1, 5, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0, 8, 0, 3, 1, 0, 0, 4, 0>.

Table 3.2: Inversion Frequencies on a sample sequence.

i	T_i	D_i
0	-	<>
1	<2, 1, 5, 0, 0, 0, 0, 0, 0, 0>	<2, 1, 5, 0, 0, 0, 0, 0, 0, 0>
2	<18, 0, 0, 0>	<2, 1, 5, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0>
3	<8, 0>	<2, 1, 5, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0, 8, 0>
4	<3, 1>	<2, 1, 5, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0, 8, 0, 3, 1>
5	<0, 0, 4, 0>	<2, 1, 5, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0, 8, 0, 3, 1, 0, 0, 4, 0>

The decoding process needs the knowledge of the multiset that described by F = $(f_1, f_2, ..., f_k)$ and D. In this case, F=(10, 4, 2, 2, 4). Using vectors S, F and D, the reconstruction the multiset M as follows:

- it first creates a vector M = |D|.
- it determines the elements of M from the ordered set S and F. S provides the element of M and F is the number of each element. In this example, the first element in S is a and the first value of F is 10. So, there are ten of a.
- D provides the location of the element. Hence, the first location is 2 for the first element (a). Therefore, M = [_, _, a, _, _, _, ..., _].

- the second position entry of D (D_2) is 1, so there is one element which is greater than a, between the first and second occurrence of a. Hence, the decoder inserts the second a in the second blank position next to the first a, so M = $[_, _, a, _, a, _, ..., _]$.

- the third entry of D (D_3) is 5, so there are 5 elements between the second and the third a, thus M = $[_, _, a, _, a, _, _, _, _, a..., _]$.

- repeating the above procedure, the decoder can fully reconstruct M.

Arnavut's empirical studies in [10] present that IF technique yields better compression gain than the recency ranking (MTF coder). Moreover, Abel in [4] compares the share of the zeros of the file book1 of Calgary Corpus over the file position for both the MTF stage output and for the IF stage output. The average share of zeros in the output of IF is rising towards the end of the file until it reaches 100% at the end. In the output of MTF, the average share of zeros fluctuates around 60%.

Weighted Frequency Count algorithm (WFC)

Another GST stage is the Weighted Frequency Count algorithm (WFC) presented by [24]. The WFC is a representative of a List Update Algorithm (LUA) and is closer to MTF than to IF. It replaces the input symbol with a corresponding ranking value. In MTF process, a MTF list of alphabet symbol is updated upon each request of an input symbol without taking the former frequency distribution of the input itself. Thus, it might push more frequent symbols aside by less frequently used symbols leading to sub-optimal compression ratios. WFC calculates the frequency distribution of input symbols. It concerns the symbols frequencies and the distance of the last occurrences of an input inside a sliding window. Each position inside the sliding window is assigned a weight. The weights of the closer distances are higher than the farther one. The occurrences of each alphabet symbol weights inside the window are summed up into a corresponding counter. The counter list is sorted in descending order. The position 0 is given to the largest counter. The weighting and sorting have to be recalculated for every input symbol. Thus, the frequent symbols obtain lower index value. However, the WFC process is more complex and high time consumption than the previous GST. So, Abel in [3] proposes Incremental Frequency Count (IFC) to reduce the complexity of WFC. It also uses counters for symbols within a sliding window, but it updates only once counter for each symbol processed.

Beside the aforementioned algorithms, there have been other variants of GST, such as Distance Coding (DC) from Binder, Wavelet trees of Foschini, sorted rank coding, etc [4, 7].

3.5 Data Coding

There are two kind of Data Coding that Burrows and Wheeler use in their original paper. First, they proposed Run Length Encoding Zeros (RLE0) after MTF, since there are a lot of zeros. Thus, RLE0 codes only the symbol zero to reduce the data size. The second algorithm is statistical coding which code a fixed number of source symbols into a variable numbers of output symbols. The final succession of coded output symbol with variable length will be on average smaller than that obtained with fixed length of input symbols.

The length of each code word is not identical for all the symbols: the most frequent symbols (those which appear most often) are coded with short code words, while the most uncommon symbols receive longer binary codes.

3.5.1 Run-level coding

There are many type of Run-level coding that have been developed by some authors to increase compression performance. As stated above, the original scheme of BWT used Run Length Encoding Zero (RLE0) since GST stage produces runs of zeros. Some authors developed their own RLE0 or other Run-level coding to improve compression performance.

Run-level coding aims to shrinks long runs of same symbols. Long runs have an important impact for the BWT and also statistical coding performances. The main function of Run-level coding is to support the probability estimation of the next stage. In order to improve the probability estimation of the DC stage, the common BWCA schemes position Run-level coding stage directly in front of the DC stage. One common Run-level coding stage for BWT based compressors is the Zero Run Transformation (RLE0) from Wheeler.

Some authors suggested a Run-level coding before the BWT stage for speed optimization, but such a stage deteriorates the compression rate in general. Since there are sorting algorithms now known which sort the runs of symbols practically in linear time such as suffix array, there is no reason to use such a stage before the BWT [28].

Wheeler's Run Length Encoding

Wheeler proposed RLE0 that code only the zero-runs. It extends the character set by one value. Figure3.3 present an example of Wheeler's run length coding.

Input stream	length	output code
x0y	1	x0y
x00y	2	x1y
x000y	3	x00y
x0000y	5	x10y
x0000000y	7	x000y

Table 3.3: Example of Wheeler's RLE [28].

This coding never expand the input data. Symbol 0 is still coded with single 0, while all longer runs are decreased in length by the coding. Symbol 0 and 1 are used to represent the length of zeros.

Run Length Encoding 2 symbols

Gringeler had the idea to position the RLE stage directly after the BWT stage instead of in front of the EC stage [2]. There are two reasons for the new order. Since the length of RLE output is usually smaller than its input, the GST stage has to process less symbols than the input of RLE. In addition, an RLE stage is usually faster than a GST stage, so the whole compression process becomes faster. The second reason is that the coding of the runs lowers the pressure of runs already at the GST stage and that leads usually to a better compressible GST output sequence [2].

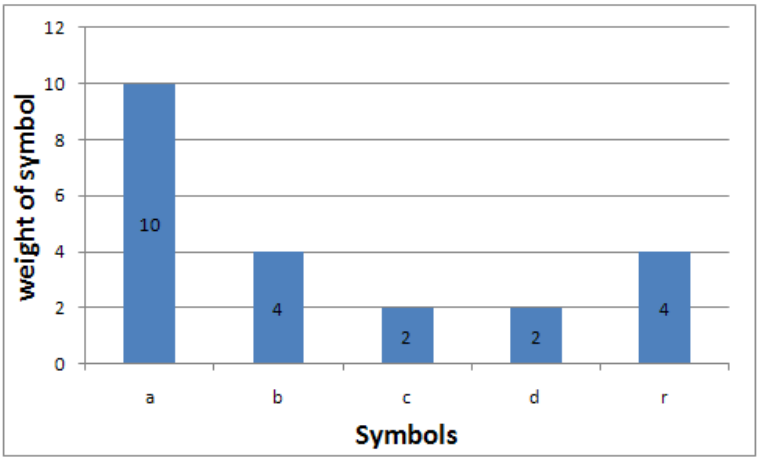
One of the example of this type is Run Length Encoding 2 symbols (RLE2). This method is used by Lehman et al [41] in their BWT chain. RLE2 stage is placed after BWT stage. So, this chain switch the RLE stage with GST stage. RLE2 stage coded only runs of size 2 or more into 2 symbols. The length of the runs is placed in a separate data stream and compressed directly using Arithmetic Coding. The separation of run length information aims to avoid the interfere of the length of the runs and the main data. Since the length of the runs is variable, it is not suited for GST stage. This process eases the pressure of runs, and the effects of symbol statistics. Otherwise, the short runs become a drawbacks of this method. It adds a cost since the runs should be encoded with zero cost.

3.5.2 Statistical Coding

Image compression schemes tend to be innovative in the mapping and modelling stages. Whereas the coding stage of most image coders is generally based on a traditional coding technique. Probably the three most influential of these coding schemes are Huffman Coding, Arithmetic Coding and the Lempel-Ziv based methods. All of these schemes are documented in any good book on data compression, however due to their importance, some of statistical coding methods will be discussed here.

Huffman code

This algorithm assigns a codeword to each symbol, so that the most frequent symbols receive a shorter codeword. Each code have to be uniquely coded.



Symbol	Weight
a	10
b	4
c	2
d	2
r	4

(a) (b)

Figure 3.10: Example of Huffman coding. (a) Histogram. (b) Data list.

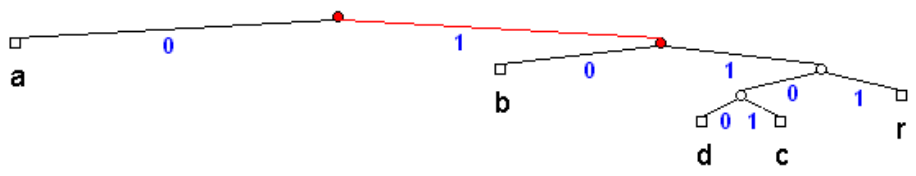


Figure 3.11: Sample of Huffman tree.

A brief description of Huffman coding, with an input example is "$rradadrrccaaaaaaaabbbb$" is given in Figure 3.10 and Figure 3.11. The original data string length equals 176 bits for 8 bpc. Firstly, Huffman coder counts the

amount of times each character appears as seen in Figure 3.10. Then, the following steps are process repeatedly until there is only one node left:

- find the two lowest weights of nodes. In this case, c and d have the lowest weights,
- create a parent node for these two nodes. The sum of the two node,
- remove the two nodes from the list and add the parent node.

The nodes with the highest weight will be near the top of tree and have shorter codes, thus the symbol a in this example. We obtain finally the code value for each symbol from the Huffman tree as described in Figure 3.11. The code can be read from root or from the top. For this example, the code is read from the root. Hence, the code a = 0, b = 10, c = 1101, d = 1100, and e = 111. Therefore, the compression bit string are:

111 111 0 1100 0 1100 111 111 1101 1101 0 0 0 0 0 0 0 0 10 10 10 10.

The compression stream equals to 46 bits, or 73.864% of space savings.

There are several variants of Huffman coding, among them are Dynamic Huffman (frequencies are calculated dynamically), Block-based Huffman (groups of symbols are coded instead of single ones), Huffword (coded the words instead of symbols), multi-Huffman, etc. Meanwhile, the previous example is a static Huffman coding.

Burrows and Wheeler used Huffman code in their proposed methods [18], because it is simple. They suggested using Arithmetic Coding (AC) because this code is more compressible but also complex than Huffman.

Arithmetic Coding

Arithmetic coding uses a probabilities of one-dimensional table. The idea is started by looking for a proper way to encode a message without assigning a fixed binary code to each symbol. So, all probabilities fall into the range $[0, 1)$, while their sum equals one in every case. This interval contains an infinite amount of real numbers, so it is possible to encode every possible sequence to a number in $[0, 1)$. We can divide the interval according to the probability of the symbols. By iterating this step for each symbol in the message, we can refine the interval to a unique result that represents the message. Any number in this interval would be a valid code.

Let M be a model that assigns a probability P_M (a_i) to each symbol ai that appears in the message. Now we can split the interval $[0, 1)$ using these values since the sum

always equals one. The size of the $i-th$ sub-interval corresponds to the probability of the symbol a_i .

As the same input as Huffman in the previous section, let's M as a model using the alphabet A = a,b,c,d,r. The probabilities of the symbols as shown in Figure 3.10 as follows:

$P_M(a)$=0.455, $P_M(b)$=0.182, $P_M(c)$=0.091, $P_M(d)$=0.091, $P_M(r)$=0.182.

Now, the interval $[0, 1)$ would be split as emphasized in Figure 3.12.

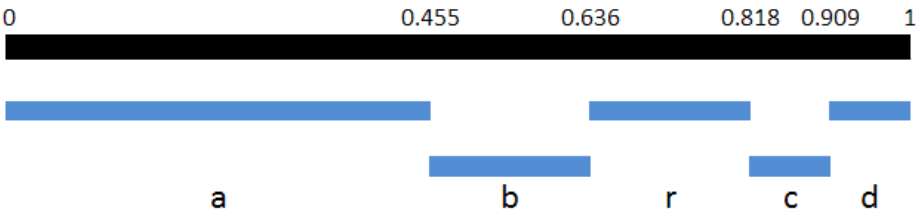


Figure 3.12: Creating an interval using given model.

The upper and lower bounds of the entire current are called interval high and low. The bounds of the sub-intervals are calculated from the cumulative probabilities:

$$K(a_k) = \sum_{i=1}^{k} P_M(a_i) \tag{3.2}$$

The values high and low change during the encoding process. Otherwise, the cumulative probabilities remain constant and are used to update high and low. The subdivision as described in Figure 3.12, depends on the model. However, for now we assume that it is given by a constant table containing the cumulative probabilities $K(a_i)$. This type of model also exists in real applications and is called static [16].

The coding process begins with the initialization of the interval $I := [low, high)$ by low = 0 and high = 1. When the first symbol s_1 is read, the interval I can be resized to a new interval I' according to the symbol. The boundaries of I' are also called low and high. I' is used as the boundaries of s_1 in the model. Then these boundaries are calculated with the following equations.

$$low := \sum_{i=1}^{k-1} P_M(a_i) = K(a_k - 1) \tag{3.3}$$

$$high := \sum_{i=1}^{k} P_M(a_i) = K(a_k) \tag{3.4}$$

Then, the new interval I' is set to $[low, high)$. The sub-interval I' becomes larger for more probable symbols s_1 . The larger the interval the lower the number of fractional places which results in shorter code words. All following numbers generated by the next iterations will be located in the interval I' since it is used as base interval as the previous process which is $[0, 1)$ (see Table 3.4).

Table 3.4: Probability table.

Character	Probability		range
a	10/22	0.45	0.00 - 0.45
b	4/22	0.18	0.45 - 0.64
c	2/22	0.09	0.64 - 0.73
d	2/22	0.09	0.73 - 0.82
r	4/22	0.18	0.82 - 1.00

The process is proceeded by changing the scale and shift the boundaries to math a new interval. Scaling is accomplished by a multiplication with $high$ - low, the length of the interval. Shifting is performed by adding low, based on the following equations.

$$low' := low + \sum_{i=1}^{j-1} P_M(a_i) \cdot (high - low) = low + K(a_{j-1}) \cdot (high - low)); \quad (3.5)$$

$$high' := low + \sum_{i=1}^{j-1} P_M(a_i) \cdot (high - low) = low + K(a_j) \cdot (high - low)). \quad (3.6)$$

Table 3.5 shows encoding process.

The next matter is the actual code. We have to specify the calculated interval. So we could simply save the upper and lower bound, but this is rather inefficient. Knowing that the whole interval is unique for this message, we can safely store only a single value inside the interval.

The decoding process has to apply the encoder backwards. The value is given and we can restore the original sequence S. We assume that the message length is known and equals l. In the first iteration we compare output coding with each interval I' = $[K(a_k - 1), K(a_k))$ to restore Table 3.5. It corresponds to the first symbol of the sequence, s_1 . To compute the next symbol, we have to modify the probability partition in the same way we did while encoding:

$$low' := low + K(a_i - 1) \cdot (high - low)), \quad (3.7)$$

Table 3.5: Low and high value.

New Character	Low value	High Value
	0.0	1.0
r	0.82	1.00
r	0.9676	1.0000
a	0.9676	0.98218
d	0.9782434	0.9795556
a	0.9782434	0.97883389
d	0.9786744577	0.9787276018
r	0.978718035862	0.9787276018
r	0.97872587993116	0.9787276018
c	0.978726981927218	0.978727136895413
c	0.978727081106863	0.978727095054000
a	0.978727081106863	0.9787270873830750
a	0.978727081106863	0.978727083931158
a	0.978727081106863	0.978727082377796
a	0.978727081106863	0.978727081678783
a	0.978727081106863	0.978727081364227
a	0.978727081106863	0.978727081222677
a	0.978727081106863	0.978727081158979
a	0.978727081106863	0.978727081130315
b	0.978727081117416	0.978727081121872
b	0.978727081119422	0.978727081120268
b	0.978727081119802	0.978727081119963
b	0.978727081119875	0.978727081119905

$$high' := low + K(a_i) \cdot (high - low)). \tag{3.8}$$

Like Huffman, there are a dynamic version of Arithmetic Coding where frequencies are calculated dynamically during the input reading.

3.6 BWT success : Text Compression

The researchers focused their attention since 1994 on improving compression ratio achieved by Burrows Wheeler Compression Algorithm (BWCA). As seen in Figure 3.1, there are at least three main stages that have been improved by some authors. This section presents a few of BWT methods that have been implemented in literatures.

3.6.1 `BZIP2`

`BZIP2` is an open source text compression that uses BWT as its main transforms. It uses a combination of different techniques to compress data in a lossless way. An input file to be compressed is divided into fixed-size blocks that can be process independently. Figure 3.13 presents the operation principle of the sequential `BZIP2` algorithm.

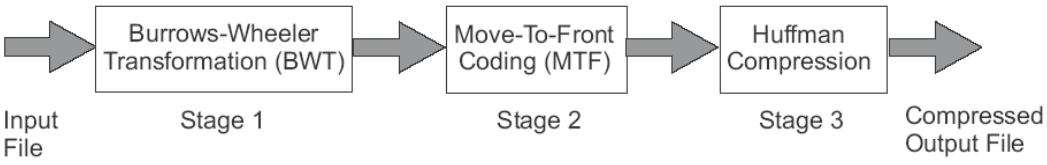


Figure 3.13: BZIP2 principle method.

`Bzip2` compresses large files in blocks. The block size affects both the compression ratio achieved, and the amount of memory needed for compression and decompression. It specifies a block size to be 100 through 900 kilobytes respectively. Larger block sizes give rapidly diminishing marginal returns. Most of the compression block size is around 300 kilobytes.

The most complex problem in this algorithm is BWT itself. Therefore, some authors proposed using Run Length Encoding. Otherwise, Seward as a creator of `BZIP2` stated that run-length encoder which is the first of the compression transformations, is entirely irrelevant. Its main purpose was to protect the sorting algorithm from the very worst case input. Moreover, Gilchrist in [31] presents a parallel data compression with `BZIP2` to increase compression performance. The results show that a significant, near-linear speedup is achieved by using parallel `BZIP2` program on systems with multiple processors, that will greatly reduce compression time for large amounts of data while remaining fully compatible with the sequential version of `BZIP2`.

3.7 Innovative Works: Image Compression

BWCA has been implemented not only for text compression but also in image compression. Some authors have been proposed their chain using BWT in image. The compression results are commonly compared with the existing image compression standard such Joint Picture Experts Group or `JPEG` (`Lossless JPEG` or `JPEG-LS`) and Joint Picture Experts Group 2000 (`JPEG2000`). Both of these standards can be implemented in lossy or lossless compression.

3.7.1 Lossy

Some authors have been implemented BWT in lossy image compression. Most of authors added BWT in `JPEG` chain to improve `JPEG` performance.

3.7.1.1 BWT based JPEG

Baik et al. in [14, 34] have been proposed an improvement of classic JPEG by adding BWT stage. The authors proposed 2 different methods . Their first proposed method performs BWT before applying Entropy Coding stage as shown in Figure 3.14. Adding BWT process slows down the operation, meanwhile it increases the average compression ratio. However, not all of the tested images improve their compression performances.

In their paper, Baik et al. present experimental results performed on 30 test images and analyzes the results. The 30 test images are those which are frequently used in image processing and compression areas. They include a variety of images (such as humans, scenes, animations) with a wide range of complexity. They implemented the proposed method by integrating BWT and the entropy coding process into PVRG−JPEG codec. Huffman coding is used in data coding stage.

Their proposed method increases the compression ratio for 15 images out of 30 test images. The highest performance ratio achieved for the proposed method is 88%, which means that the proposed method reduces the file size by 12% compared with JPEG. The average performance ratio of the 15 images is 35.3%. However, the proposed method fails to compress further over JPEG for the other 15 images, which are mostly complex images. The average performance ratio for the other 15 images is low. The average performance ratio of the 30 images is 11%. Hence, in general, the proposed method increases the compression ratio of images over JPEG.

Baik et al. improve their first proposed methods by adding "Degree Of Repetition" (DOR), see Figure 3.15. DOR analyzes the repetition data in order to avoid the weakness of first methods.

Wisemann [78] combines DCT and BWT. DCT is used before BWT. The proposed methods of Wisemann is shown in Figure 3.16.

A method enhances common JPEG standard compression efficiency by exploiting the Burrows-Wheeler compression technique. As shown in Figure 3.16, the traditional Huffman is changed by the Burrows-Wheeler compression. This paper also analyzes the BWT block size impacts. Obviously, it influences the compression rates whereas larger block provides better compression rates. The authors only tested a block size from 100 KB to 900 KB. It determines that 900 KB block size gives the best performance.

The overall result shows that high quality images are yield a better compression ratio and also even a poor quality of a synthetic image can be compressed better. This

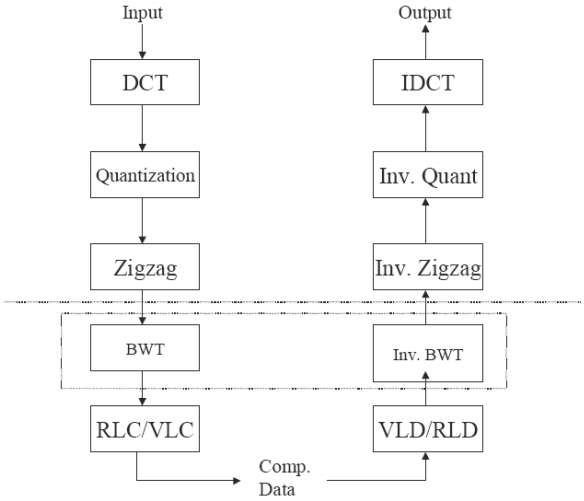


Figure 3.14: The first proposed methods of Baiks [14].

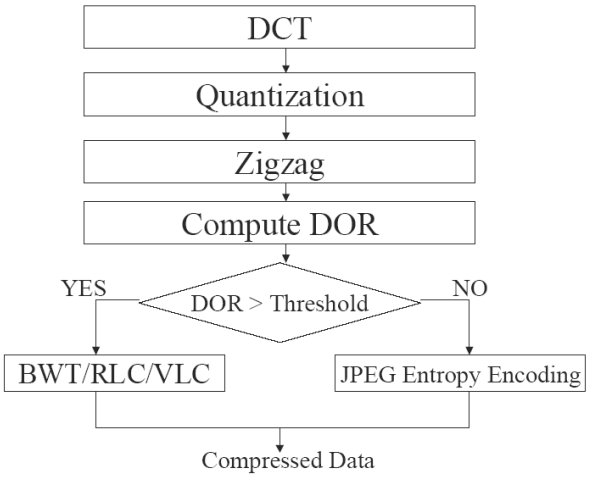


Figure 3.15: The second proposed methods of Baik [34].

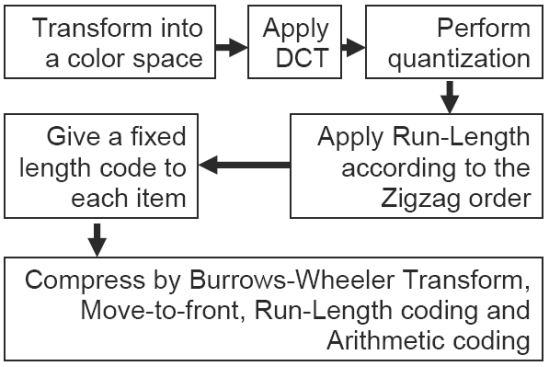


Figure 3.16: Wisemann proposed method [78].

methods can also be applied in lossless compression by emitting the quantization process.

3.7.2 Lossless

3.7.2.1 DWT and BWT

Guo in [33] has been proposed a waveform and image compression scheme by combining Discrete Wavelet Transform (DWT) and Burrows Wheeler Transform (BWT). BWT is used after DWT to improve the compression performance for many natural signals and images. This techniques significantly improve its compression performance [33].

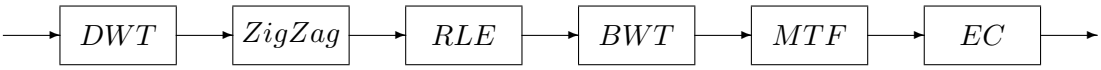


Figure 3.17: Guo proposed scheme [33].

Guo's method is shown in Figure 3.17. The image is first processed using wavelet transform, then it is converted from 2D to 1D sequence by zigzag scan. This scheme has been confirmed well compared with the embedded zerotree wavelet compression.

3.7.2.2 A Study of Scanning Paths for BWT Based Image Compression

[38] studies BWT performance on digital images by analyze the reordering of 2-dimensional image data. BWT is a linear transformation that was created for text. Therefore, a conversion of an image from 2D to 1-D is needed in order to apply it to BWT. There are several method of image reordering or convert it from 2D to 1D. One of the focus of this thesis is in finding efficient ways to linearize the data, and also analyzing block segmentation to exploit the spatial redundancies in pixel values present in close neighborhood. The experimental results determine that linearized using the snake scan at 32×32 block size, and this phase improves compression performance. Otherwise, the average compression result does not yield better than `JPEG2000`. There are two rendered images that give better results than `JPEG2000` among 10 image tested. The proposed method is better than BWT classic or other text compression methods.

3.7.2.3 The Impact of Lossless Image Compression to Radiographs.

Lehman et al. [41] present a BWCA methods implementing to radiographs. They modified BWCA original method by using Incremental Frequency Count (IFC) as MTF alternating and also Run Length Encoding 2 symbols to change Run Length Encoding zero in Original BWCA chain. Figure 3.16 shows the proposed method. They proposed to put modified RLE after BWT. The modified RLE coded all runs of size 2 or more into 2 symbols and the length information is separated with data stream so it does not disturb the main output stream. The next transform which is IFC (as variant of MTF) adapt modified RLE to improve compression ratios. The last stage of this method uses Arithmetic Coding that gives better compression result than Huffman.

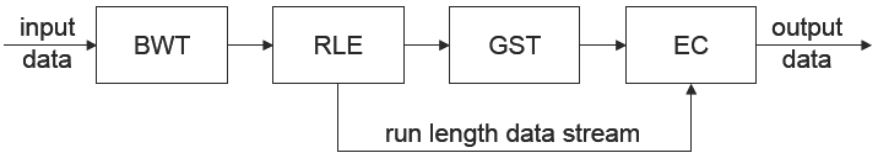


Figure 3.18: Lehmann proposed methods [41].

Their overall results show that BWCA is slightly more effective that JPEG2000.

3.7.3 Nearly Lossless

3.7.3.1 Segmentation-Based Multilayer Diagnosis Lossless Medical Image Compression

Bai proposed method [13] that has a segmentation process as its pre-processing. It divides the image in 3 parts based on its diagnosis importance level; Primary region of interest or ROI Layer (PRL) (all diagnostically important regions) and this part may vary depending on different clinical studies; Normal ROI Layer (NRL) (unimportant regions which surround the PRL regions) and it may help the clinician to easily observe and locate PRL within the original image, and evaluate possible interactions with surrounding organs; Background Region Layer (BRL) (regions other than PRL and NRL regions), this part mainly locate outside human body / organs and without any diagnostic value. The proposed method can be seen in Figure 3.19. The primary layer is processed using BWCA that has to be lossless, and the other part which is not important is processed as lossy.

The disadvantage of this method is the nearly lossless. It needs some knowledge to determine which the part of the image is not important to be saved. Otherwise, there

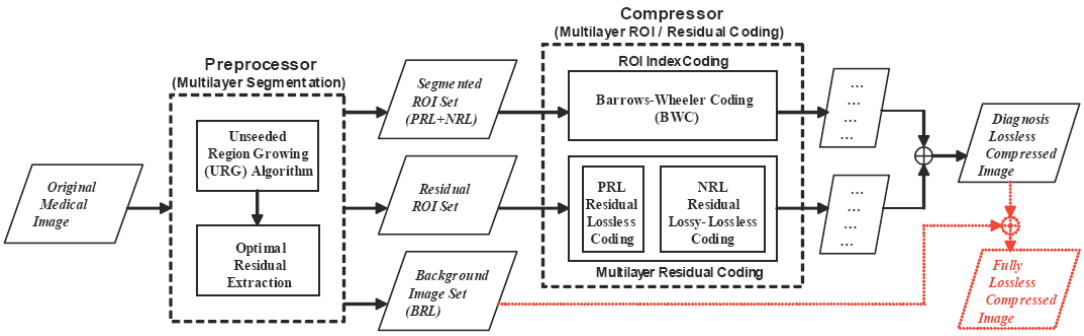


Figure 3.19: Bai et al. proposed methods [13].

are some advantages of this methods. It has better compression ratios than lossless methods, since it has been segmented in 3 parts, progressive transmission is a highly desirable feature to send the most important part first for many image compression applications, especially telemedicine.

3.8 Summary

BWT is originally dedicated to text compression but some authors also have been applied this algorithm to image compression. A few of authors proposed to combine it with the existing image compression technique such as JPEG, others improve the performance of BWT by adding pre- and/or post-processing stage. Best result is obtained in the second solution. Therefore, each stage of BWCA chain should be dedicated and optimized for certain application, thus here is in image compression. We will analysis the impact of each algorithm in BWCA chain for image compression application.

Chapter 4

Improvements on classical BWCA

Contents

4.1 Introduction 66

4.2 Corpus 66

4.3 Classic BWCA Versus Text and Image Compression Standards 68

4.4 Some of Pre-processing Impacts in BWCA 72

4.4.1 Image Block 72

4.4.2 Reordering 75

4.5 GST Impact 77

4.6 Data Coding Impact 80

4.6.1 Run Length Encoding Impact 80

4.6.2 Statistical Coding 83

4.7 Nearly Lossless BWCA Compression 91

4.8 Bit Decomposition in BWCA 94

4.1 Introduction

This chapter explores the BWT in more detail. We explain why BWT can achieve high compression. We examine some modifications in pre- and post-processing of BWT. The modifications is started from BWT classical chain, then the updated algorithms that have been proposed by some authors and that we propose. This chapter is began with corpus that is commonly used by some authors in data compression, then some corpus that have been implemented using BWCA.

4.2 Corpus

Data compression measurement in empirical experiments is the a amount of compression achieved on some set of files. Researchers also often report the speed of compression, and the amount of primary memory required to perform the compression. The speed and memory requirements can be different for the encoding and decoding processes, and may depend on the file being compressed. This can result in a daunting number of factors that need to be presented.

A number of authors have used the Canterbury Corpus (shown in Table 4.2) and its predecessor, the Calgary Corpus (as seen in Table 4.1). Both of them consist of some collections of typical files that usually use in the evaluation of lossless compression methods.

Table 4.1: The Calgary Corpus.

No.	Files	Description	Size (bytes)
1	BIB	725 bibliographic references	111,261
2	BOOK1	unformatted ASCII text	768,771
3	BOOK2	ASCII text in UNIX	610,856
4	GEO	seismic data	102,400
5	NEWS	USENET batch file	377,109
6	OBJ1	compilation of PROGP.	21,504
7	OBJ2	Macintosh exe. program	246,814
8	PAPER1	UNIX "troff" format	53,161
9	PAPER2	UNIX "troff" format	82,199
10	PIC	1728 x 2376 bitmap image	513,216
11	PROGC	Source code in C	39,611
12	PROGL	Source code in Lisp	71,646
13	PROGP	Source code in Pascal	49,379
14	TRANS	transcript of a terminal session.	93,695

There are 18 files of Calgary Corpus, though several empirical studies provide only 14 files. The four of less commonly used files version are text files in UNIX "troff" format, PAPER3 through PAPER6 that obviously represented by other PAPER1 and PAPER2. Meanwhile, the Canterbury Corpus consists of 11 files, and the explanation

Table 4.2: The Canterbury Corpus [12].

No.	Files	Abbrev.	Category	Size (bytes)
1	alice29.txt	text	English text	152089
2	asyoulik.txt	play	Shakespeare	125179
3	cp.html	html	HTML source	24603
4	fields.c	Csrc	C source	11150
5	grammar.lsp	list	LISP source	3721
6	kennedy.xls	Excl	Excel Spreadsheet	1029744
7	lcet10.txt	tech	Technical writing	426754
8	plrabn12.txt	poem	Poetry	481861
9	ptt5	fax	CCITT test set	513216
10	sum	SPRC	SPARC Executable	38240
11	xargs.1	man	GNU manual page	4227

of how the files were chosen, and why it is difficult to find typical files to represent any compression method, can be found in [12]. Previously, compression software was tested using a small subset of one or two non-standard files. This was a possible source of bias to the experiments, as the data used may have caused the programs to exhibit anomalous behavior.

These corpora have become de facto standards for lossless compression evaluation. Moreover, the Calgary corpus is commonly used to test data compression programs. Certainly both corpora have gained a great deal of support in the compression community.

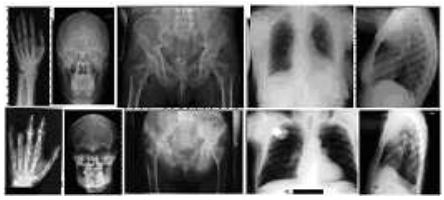


Figure 4.1: Example of tested images. Upper row: directly digital, lower row: secondarily captured. From left to right: Hand; Head; Pelvis; Chest, Frontal; Chest, Lateral.

Table 4.3: IRMA Corpus.

No.	Image Category	Nb. of Image	Size			
			smallest	dim.	biggest	dim.
1	Hands primary	120	234434	251 x 934	3723270	1571 x 2370
2	Hands secondary	120	169880	310 x 548	2960920	1580 x 1874
3	Heads primary	120	1076400	920 x 1170	4037040	1680 x 2403
4	Heads secondary	120	1854504	1198 x 1548	9069966	2718 x 3337
5	Pelvis primary	120	1301760	1280 x 1017	5435486	2786 x 1951
6	Pelvis secondary	120	601026	919 x 654	13093760	3992 x 3280
7	Thoraces frontal primary	120	685500	914 x 750	8400232	2836 x 2962
8	Thoraces frontal secondary	120	5088864	2607 x 1952	7470000	3000 x 2490
9	Thoraces lateral primary	120	2186181	1347 x 1623	8219800	2920 x 2815
10	Thoraces lateral secondary	120	220580	538 x 410	7086540	2490 x 2846

The BWT compression method that we present, has been applied for lossless text

compression area. Nevertheless, it can be applied successfully to image. The lossless approach is obviously appropriate to medical image compression, which is expected to be lossless. Therefore, a few medical images are used to analyzed the BWT implementation. The experiments use medical images from IRMA (Image Retrieval in Medical Applications) data-base [42] and Lukas Corpus [1]. IRMA data-base consists 1200 directly digital (image primary) and secondarily digitized (image secondarily) X-ray films in Portable Network Graphics (`PNG`) and Tagged Image File Format (`TIFF`) format, 8 bits per pixel (8 bpp), examples of images are shown in Figure 4.1. There are 10 categories of IRMA data-base as seen in Table 4.3. Each category consists of 120 images. Image size is between 169 kilo bytes (in the Hand Secondary category) till 1309 kilo bytes (in the Pelvis Secondary). These images are also used by Lehmann in their BWCA modified [41].

Another medical image corpus is Lukas Corpus. This Corpus is used to evaluate the practical performance of lossless compression algorithms in the medical imaging field. We use 20 files of two dimensional 8 bit radiographs in `TIF` format. The file sizes are between 1088 to 3752 kilo bytes.

4.3 Classic BWCA Versus Text and Image Compression Standards

The classic BWCA is described in Figure 3.1, and it was created for text compression. In this scheme, Burrows and Wheeler propose using MTF as the second stage (GST stage), then RLE zero and finally Huffman or Arithmetic Coding as the last stage. Our first tested chain of BWCA uses MTF as a second stage, then a simple Run Length Encoding zero (RLE0) and Arithmetic Coding as Data Coding stage (see Figure 4.2.

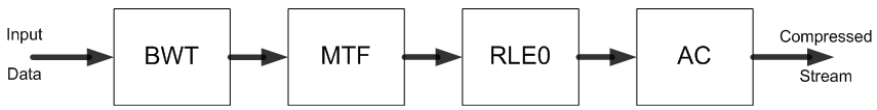


Figure 4.2: The first tested classic chain of BWCA.

This paragraph aims to compare the intrinsic performances of this classical chain with the existing standards in the compression field especially on texts and images.

The compression performances of this classic chain with similar scheme based on combinatorial transform are proposed in two simulations presented in Table 4.4 and Table 4.5. These two tests are based on all the documents respectively available in

Table 4.4: Comparison results of Canterbury Corpus using our BWCA classic chain and BZIP2.

Files	Orig. Size	Classic Chain of BWCA				BZIP2			
		comp. size	comp. ratio (CR)	bpc	Space savings (%)	comp. size	comp. ratio (CR)	bpc	Space savings (%)
alice29.txt	152089	47113	3.228	2.48	69.02	43202	3.520	2.27	71.59
asyoulik.txt	125179	43107	2.904	2.75	65.56	39569	3.164	2.53	68.39
cp.html	24603	8324	2.956	2.71	66.17	7624	3.227	2.48	69.01
fields.c	11150	3397	3.282	2.44	69.53	3039	3.669	2.18	72.74
grammar.lsp	3721	1453	2.561	3.12	60.95	1283	2.900	2.76	65.52
kennedy.xls	1029744	168825	6.099	1.31	83.61	130280	7.904	1.01	87.35
lcet10.txt	426754	117760	3.624	2.21	72.41	107706	3.962	2.02	74.76
plrabn12.txt	481861	158468	3.041	2.63	67.11	145577	3.310	2.42	69.79
ptt5	513216	54941	9.341	0.86	89.29	49759	10.314	0.78	90.30
sum	38240	13951	2.741	2.92	63.52	12909	2.962	2.70	66.24
xargs.1	4227	1933	2.187	3.66	54.27	1762	2.399	3.33	58.32
comp. efficiency	2810784	619272	3.815	2.462	69.22	542710	4.303	2.225	72.18

Table 4.5: Compression performances of Calgary Corpus using classical BWCA, BW94, and BZIP2.

Files	Orig. Size	First Chain of BWCA				BW94				BZIP2			
		comp. size	comp. ratio (C.R.)	bpc	Space savings (%)	comp. size	comp. ratio (C.R.)	bpc	Space savings (%)	comp. size	comp. ratio (C.R.)	bpc	Space savings (%)
bib	111261	29957	3.714	2.15	73.08	28750	3.870	2.07	74.16	27467	4.051	1.97	75.31
book1	768771	252116	3.049	2.62	67.21	238989	3.217	2.49	68.91	232598	3.305	2.42	69.74
book2	610856	171137	3.569	2.24	71.98	162612	3.757	2.13	73.38	157443	3.880	2.06	74.23
geo	102400	63486	1.613	4.96	38.00	56974	1.797	4.45	44.36	56921	1.799	4.45	44.41
news	377109	128459	2.936	2.73	65.94	122175	3.087	2.59	67.60	118600	3.180	2.52	68.55
obj1	21504	11102	1.937	4.13	48.37	10694	2.011	3.98	50.27	10787	1.994	4.01	49.84
obj2	246814	82004	3.010	2.66	66.77	81337	3.034	2.64	67.05	76441	3.229	2.48	69.03
paper1	53161	18013	2.951	2.71	66.12	16965	3.134	2.55	68.09	16558	3.211	2.49	68.85
paper2	82199	27230	3.019	2.65	66.87	25832	3.182	2.51	68.57	25041	3.283	2.44	69.54
pic	513216	54941	9.341	0.86	89.29	53562	9.582	0.83	89.56	49759	10.314	0.78	90.30
progc	39611	13646	2.903	2.76	65.55	12786	3.098	2.58	67.72	12544	3.158	2.53	68.33
progl	71646	17393	4.119	1.94	75.72	16131	4.442	1.80	77.49	15579	4.599	1.74	78.26
progp	49379	11975	4.124	1.94	75.75	11043	4.472	1.79	77.64	10710	4.611	1.74	78.31
trans	93695	19839	4.723	1.69	78.83	18383	5.097	1.57	80.38	17899	5.235	1.53	80.90
Comp. efficiency	3141622	901298	3.643	2.57	67.82	856233	3.669	2.18	72.75	828347	3.793	2.11	73.63

Table 4.6: Experimental results on Calgary Corpus using different BWCA chain.

Author	Date	Algorithm	bpc
Fenwick	1995	Order-0 Arithmetic	2.520
Fenwick	1996	Structured Arithmetic	2.340
Deorowicz	2000	Arithmetic Context	2.270
Schindler (szip)	1996	Hierarchical Arithmetic	2.360
Balkenhol	1998	Cascaded Arithmetic	2.300
Balkenhol	1999	Cascaded Arithmetic	2.260
Seward (BZIP2)	2000	Huffman blocks	2.370
Burrows & Wheeler	1994	16000 Huffman blocks	2.430
Fenwick	2002	VL codes, 1000 blocks	2.570
Fenwick	1998	Sticky MTF, Structure Arithmetic	2.300
Deorowicz	2000	Weighted Frequency Count	2.250
Wirth, Moffat	2001	No MTF; PPM Context	2.350
Arnavut	2002	Inversion Ranks	2.300

Canterbury and Calgary data-bases. A comparison of the results obtained with classic BWCA chain and `BZIP2` from Seward [62] is proposed in both corpus. Nevertheless, the original Burrows and Wheeler's results (BW94) [18] are only based on the Calgary Corpus. The compression results of Table 4.4 ansd Table 4.5 are presented in three different types of compression measurements which are:

- compression ratio (CR) as seen in Equation 1.3,
- average number of bits stored in bits per pixel (bpp) for an image and per character (bpc) for text (see Equation 1.4),
- size relative reduction based on its original data (see Equation 1.5),

as explained before in Section 1.3.

Results of `BZIP2` and BW94 are better than our first simulation for both Canterbury and Calgary Corpus. All of these methods use BWT as its main transform. Nevertheless, they use different data coding as explained in Section 3. Our classic BWCA chain uses AC as the last stage, however BW94 and `BZIP2` use Huffman Coding. As stated in Section 3.5.2, generally AC obtains better CR than Huffman but often, as seen in Table 4.4 and Table 4.5, are not systematically the case. And even though, BW94 and `BZIP2` use the same coding, they do not give similar results. Some authors have been working to improve compression efficiency of BWCA chain.

Table 4.6 regroups the performances of a few compression schemes that use BWT as its main transform. The compared results have been based on Calgary Corpus. The 3^{rd} column shows the main algorithm differences for each method. For example, by improving Cascaded Arithmetic in 1999, Balkenhol obtains better compression ratio than his previous tests in 1998. Otherwise, the best CR is obtained by Deorowicz by changing MTF with Weighted Frequency Count (WFC). Therefore, this table highlights that all this studies mainly focuss on the last which are indeed crucial in the resulted CR. Obviously, the two stages are going to be mainly consider to establish on our scheme.

Our main idea at the start of this thesis was to apply these schemes designed for text compression to images compression. Therefore, classic BWCA chain has also been tested on the well known medical image data-base named IRMA and Lukas Corpus. A sample of 100 images have been randomly extracted from these corpus for the following compression as seen in Table 4.7 [72]. These results show that BWCA method can also be implemented in some applications, not just text, but also in images. Classic chain of BWCA can get better compression ratio than `Lossless JPEG` but

JPEG2000 and JPEG-LS are significantly better than classic scheme of BWCA. The average compression ratio is

- 2.280 for Lossless JPEG,
- 2.978 for JPEG-LS,
- 2.923 for JPEG2000, and
- 2.516 for BWCA original scheme.

Table 4.7: Compression results using BWCA Classic chain for Lukas corpus[72].

Images	Size of raw images	Lossless JPEG		JPEG-LS		JPEG2000		BWCA org. sch.	
		Comp. Size	C.R	Comp. Size	C.R	Comp. Size	C.R	Comp. Size	C.R
Hand1	2 235 688	994 043	2.249	721114	3.100	746 812	2.994	921 077	2.427
Hand2	1 120 960	553 455	2.025	393690	2.847	404 790	2.769	503 559	2.226
Hand3	431 172	201 901	2.136	153269	2.813	157 759	2.733	201 396	2.141
Hand4	1 667 040	761 412	2.189	564838	2.951	573 070	2.909	608 922	2.738
Head1	1 515 533	760 802	1.992	592282	2.559	593 391	2.554	681 419	2.224
Head2	2 839 656	1 284 695	2.210	968504	2.932	966 688	2.938	1 119 363	2.537
Head3	2 788 500	1 179 829	2.363	937784	2.973	951 033	2.932	1 041 038	2.679
Head4	3 256 000	1 357 005	2.399	1177592	2.765	1 277 882	2.548	1 143 073	**2.848**
Pelvis1	3 239 730	1 877 742	1.725	1584479	2.045	1 589 535	2.038	1 770 899	1.829
Pelvis2	3 126 784	1 740 236	1.797	1486126	2.104	1 485 588	2.105	1 661 580	1.882
Pelvis3	1 076 768	506 967	2.124	421486	2.555	420 919	2.558	501 369	2.148
Pelvis4	7 036 956	3 374 061	2.086	3022968	2.328	3 230 414	2.178	2 267 335	**3.104**
Thorac Frontal1	3 713 600	2 046 205	1.815	1814835	2.046	1 830 742	2.028	2 011 249	1.846
Thorac Frontal2	3 405 076	1 806 522	1.885	1606460	2.120	1 611 065	2.114	1 780 515	1.912
Thorac frontal3	6 957 060	2 651 775	2.624	2018238	3.447	2 047 942	3.397	2 431 091	2.862
Thorac frontal4	7 006 860	3 027 914	2.314	2525286	2.775	2 543 669	2.755	2 607 353	2.687
Thorac litoral11	6 184 913	2 590 276	2.388	2094656	2.953	2 115 375	2.924	2 430 634	2.545
Thorac litoral12	2 186 181	1 227 943	1.780	1053219	2.076	1 053 533	2.075	1 170 793	1.867
Thorac litoral13	5 859 510	1 957 078	2.994	1379998	4.246	1 429 536	4.099	1 773 996	3.303
Thorac litoral14	220 580	112 457	1.961	91029	2.423	93 861	2.350	114 544	1.926
Av. 20 images			2.153		2.703		2.650		2.387
Av. 100 images			2.280		2.978		2.923		2.516

These tests show that BWCA performances are, in average (for the 20 or 100 images tested), between Lossless JPEG and JPEG2000. For 100 images tested, 10 images have better compression ratios than JPEG2000, meanwhile 22 images provide lower compression rate than Lossless JPEG. Even though, JPEG2000 performances are always more performance than Lossless JPEG, for some images, such as image Pelvis where the compression ratio using BWCA original is 3.104, while JPEG2000 could get only 2.178. The difference of compression ratio between JPEG2000 and BWCA original scheme in this image is quite significant, which is 0.926. These results can be consider as very promising for BWCA scheme. Since this method was applied to text compression, therefore there are some of phases that we must analyze to accommodate it into image compression. This original scheme has

a few flaws. Employing RLE-0 is not effective to decrease data, because many consecutive characters still exist after RLE-0. Employing Move-To-Front (MTF) as one of GST before RLE-0 could not reduce this phenomenon effectively, because MTF transforms one string of symbols into another string of symbols of the same length with different distribution. Otherwise, there are some MTF variants that have to be considered to improve compression performance.

We tested all data of IRMA for a few of data compression standards which are `Lossless JPEG`, `JPEG-LS`, `JPEG2000` and classic BWCA chain (i.e using BWT as its main transform) as shown in Table 4.8. Here, classic chain of BWCA obtains better comparison ratios for one image category, which is Head secondary. Otherwise, `JPEG-LS` obtains better compression ratios for 7 image categories and `JPEG2000` in other 2 categories. These preliminary promising results have confirmed the potential interest of using combinatorial transforms into a image compression scheme. Our strong believe has been confirmed with several publications at the time. Our investigation based on these preliminary results on image and our experience on the text compression have conducted us to focus on not only in the different stages of coding (statistical or not statistical), but also in pre- and post-processing of BWT.

Table 4.8: CR comparable results between classical BWCA and existing image standards for IRMA corpus.

No.	Images	Nb. of Image	`JPEG` Lossless (Matlab)	`JPEG 2000` (Jasper)	`JPEG -Ls` (HP)	BWCA clasical chain	The best Value
1	Hands primary	120	2.019	2.557	**2.587**	2.123	`JPEG-LS`
2	Hands secondary	120	2.379	3.134	**3.209**	2.693	`JPEG-LS`
3	Heads primary	120	2.188	**2.970**	2.939	2.496	`JPEG2000`
4	Heads secondary	120	2.245	2.575	2.665	**2.936**	BWCA
5	Pelvis primary	120	1.745	**2.019**	2.016	1.795	`JPEG2000`
6	Pelvis secondary	120	2.360	2.788	**2.863**	2.763	`JPEG-LS`
7	Thoraces frontal primary	120	2.048	2.383	**2.396**	2.035	`JPEG-LS`
8	Thoraces frontal secondary	120	2.612	3.394	**3.456**	2.982	`JPEG-LS`
9	Thoraces lateral primary	120	2.296	2.826	**2.840**	2.379	`JPEG-LS`
10	Thoraces lateral secondary	120	2.528	3.171	**3.232**	2.753	`JPEG-LS`
	Av. comp. ratio	1200	2.242	2.782	**2.820**	2.496	`JPEG-LS`

4.4 Some of Pre-processing Impacts in BWCA

4.4.1 Image Block

Section 3.3 stated that BWT is based on a sorting algorithm. There are several methods that have been proposed to improve BWT sorting process, but this improvement does not influence the compression performance.

BZIP2 is using block splitting to decrease time processing, we consider the impact of this preprocessing on CR. Some authors stated that this input diminution influences compression performances [18, 78], there will be fewer repetitions symbols. Burrows and Wheeler in their original paper give an example of block size of file book1 of Calgary Corpus as seen in Table 4.9. The compression efficiency decreases systematically from 2.49 bpc for a block size 750 KB (the whole data) to 4.34 bpc for 1 KB of block size.

Table 4.9: The effect of varying block size (N) on compression of book1 from the Calgary Corpus [18].

Block size or N (bytes)	book1 (bpc)
1k	4.34
4k	3.86
16k	3.43
64k	3.00
256k	2.68
750k	2.49

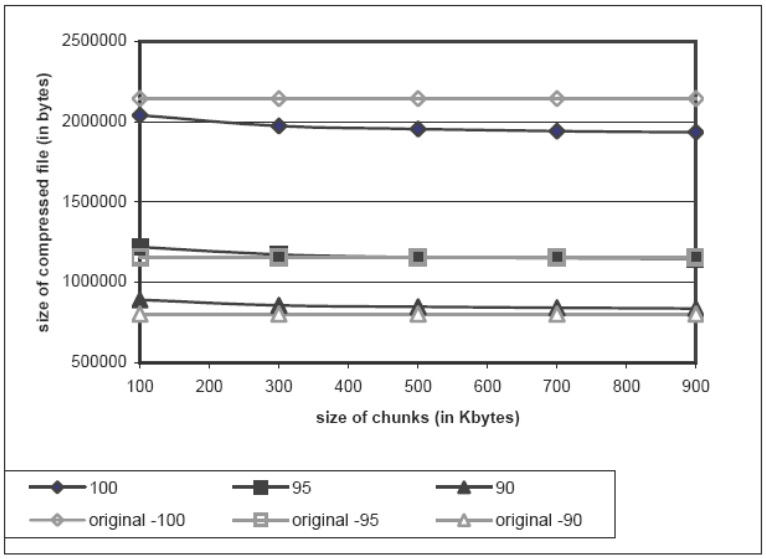


Figure 4.3: Block size impact to compression efficiency in images [78]. Tests are obtained with three JPEG parameters, which are 100, 95, and 90.

Figure 4.3 presents Wiseman's [78] empirical studies. It shows also, for a few block size (N) tests, that splitting a file into few blocks decreases compression performance. The block size (size of chunks) tested was from 100 KB to 900 KB and was used to the lossy compression methods. The compression obviously becomes better using

larger block size of chunks. Otherwise, descending quality of image produces a lot of zeroes and also decrease BWT performances, since fewer repetitions occur, this is shown by parameter 90, 95 and 100 which are the JPEG image quality parameters, thus the compression efficiency becomes poorer, therefore the compression performance depends on image nature, Wiseman suggest to apply this method only for high quality real life images.

Meanwhile, Lehmann stated in [41] that the block size ranges are generally from 1 MB to 10 MB and all blocks are processed separately. Our previous paper [72] also present this studies. Alleviation of BWT input decreases its complexity, but also its compression performances. Figure 4.4 shows an example of image that is split into 10 block. The size of image, for this example, is 512×512. Therefore, the image with size 962×2324 is divided into 10 blocks images as seen in Figure 4.4. Each block is processed separately. The detail results of each block is presented in Table 4.10. The compression ratio for 10 blocks images is 2.131, meanwhile the compression ratio for whole image is 2.427. So, the block size reduces the compression efficiency.

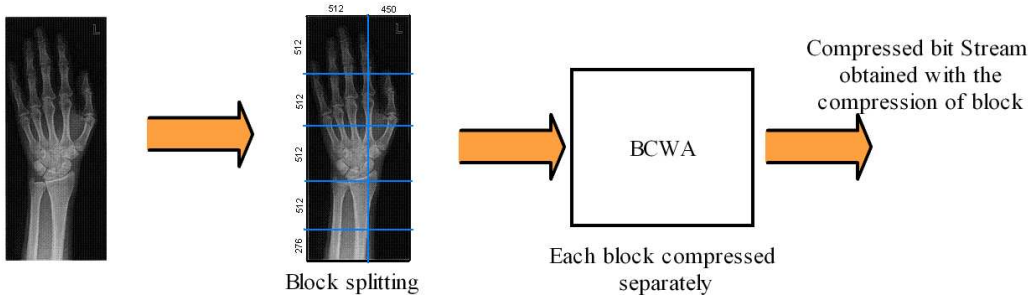


Figure 4.4: Example of an image divided into a small blocks.

Table 4.10: Results of image blocks in Figure 4.4.

Block	dimension	orig block size	comp size	C.R
1	512×512	262,144	98,309	2.667
2	450×512	230,400	69,222	3.328
3	512×512	262,144	140,923	1.860
4	450×512	230,400	113,890	2.023
5	512×512	262,144	135,903	1.929
6	450×512	230,400	129,343	1.781
7	512×512	262,144	130,594	2.007
8	450×512	230,400	109,198	2.110
9	512×276	141,312	71,461	1.977
10	450×276	124,200	50,119	2.478
		2,235,688	1,048,962	2.131

The block size or the maximum amount of data to sort is also a varied parameter of BWT. If computing resources are insufficient to sort all the data in one large block, the

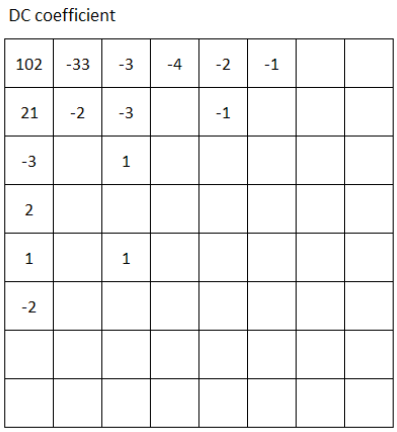
data can be split into multiple blocks. This process decreases compression efficiency. However, different type of data should not be lumped into the same block as compression efficiency will decrease. An example is the Calgary Corpus. Higher compression of the corpus is achieved by compressing each file separately rather than concatenated together in one large block [20].

Time processing decreasing has shown in our paper [72], however the CR also decreases. Nevertheless, Section 3.3 has been described a few BWT sorting improvement. We propose using the Nongs method [52] that has been implemented by Yuta Mori and provides better performance for BWT process. And based on our simulation process, the worst time to compress a whole image is 3 ms for 8 MB image size. Meanwhile, the decoding process is faster than coding. So, we do not consider image block process for our next simulation.

4.4.2 Reordering

BWT method is used to compress two-dimensional images, but the input of BWT is a one dimensional sequence. Thus, the image has to be converted from 2 dimensional image into one dimensional sequence. This conversion is referred to as reordering or scan path. Some coding, as Arithmetic Coding or BWT itself, depend on the relative order of gray scale values, they are therefore sensitive to the reordering method used.

Figure 4.5: Block of quantised coefficients [57].



The optimum method of data reordering depends on the data distribution [57] and the compression method. For example, in the DCT process, the reordering quantized data is evenly distributed about the top left of the array (see Figure 4.5) to clustered the non-zero values together. So, zigzag method is commonly used for “DC" coefficient.

BWT is different approach. The reordering process is used for BWT input. Figure 4.6 shows several example of data reordering. For this small image, the most convenient reordering method for BWT input is the first method (Left), because there are repetition symbols. Nevertheless, the data distribution of the medical image tested is unknown. Therefore, the empirical studies are needed to find the suitable technique for these images using BWT.

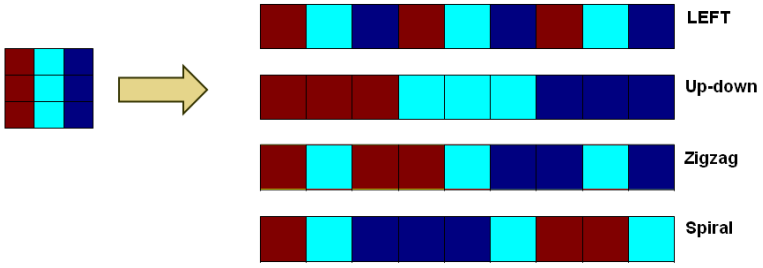
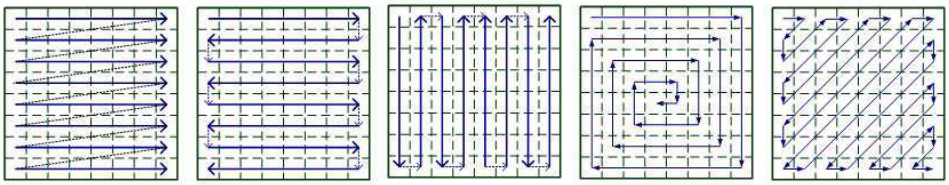


Figure 4.6: 6 examples of data reordering.



(a) Left (b) Left-Right (c) Up-Down (d) Spiral (e) Zigzag

Figure 4.7: A few of reordering methods.

Some of the popular reordering schemes are given in Figure 4.7. We have tested 8 different methods; namely: scanning image from left to right (L), left to right then right to left (LR), up to down then down to up (UD), zigzag (ZZ), spiral, divide image into small blocks 8×8, small blocks 8×8 in zigzag, and small blocks 3×3 in zigzag.

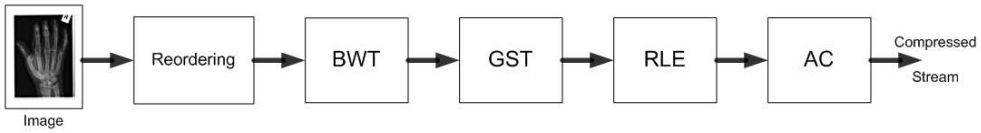


Figure 4.8: Method 1 of BWCA.

Our preliminary test uses BWT original method (Figure 4.7) and also adds different image reordering methods to see this pre-processing impact in compression performance, thus the scheme is modified as seen in Figure 4.8 (from now on, it is consider

Table 4.11: Image compression ratios for different type of reordering methods [70, 71].

Image	L	LR	UD	Spiral	ZZ	8×8	8×8ZZ	3×3zz	JPEG -LS	JPEG 2000
Hand1	2.372	2.366	**2.547**	2.536	2.257	2.204	2.262	2.336	3.100	2.994
Hand2	2.260	2.251	**2.390**	2.382	2.091	2.052	2.073	2.138	2.847	2.769
Hand3	2.114	2.123	**2.253**	2.221	1.991	1.933	1.994	2.049	2.813	2.733
Hand4	2.685	2.679	**2.830**	2.802	2.551	2.411	2.510	2.557	2.915	2.909
Head1	2.219	2.216	**2.274**	2.273	2.155	2.108	2.174	2.220	2.559	2.554
Head2	2.481	2.480	2.527	**2.538**	2.366	2.337	2.392	2.466	2.932	2.938
Head3	2.566	2.565	2.527	**2.563**	2.350	2.303	2.349	2.422	2.937	2.932
Head4	2.726	2.721	2.721	**2.764**	2.544	2.502	2.542	2.622	2.765	2.548
Pelvis1	1.808	1.808	**1.842**	1.835	1.760	1.750	1.782	1.814	2.045	2.038
Pelvis2	1.850	1.848	**1.890**	1.876	1.806	1.791	1.829	1.863	2.104	2.105
AV. 10	2.308	2.306	**2.380**	2.379	2.187	2.139	2.190	2.249	2.709	2.665
Av. 100	2.516	2.515	**2.577**	2.575	2.357	2.315	2.364	2.442	2.978	2.924

as the first proposed method (M1)). We use MTF as GST stage, simple Run Length Encoding zero (RLE0) and Arithmetic Coding (AC) for EC stage. Table 4.11 shows the results of these tests for 8 different reordering techniques. It shows that scan path influences BWT methods performance. Image scan vertically (up-down) is slightly better than spiral method. For 100 tested images, 49 images give better result using this method, 47 images using scan image spiral. The worst value of compression ratios are obtained using scan image in a block 8×8 (85 images), and 14 images using zigzag mode, see Figure 4.7(c). This preprocessing stage can increase the compression ratios around 4% than conventional scanning. In other words, the neighborhood of pixels influences the permutation yields by the BWT.

This result also shows that current BWT method improves its performance and it is better than `Lossless JPEG` but still inferior than `JPEG2000`. For 100 images tested, 91 images determine better CR than `Lossless JPEG`. This previous result shows that using scan image up-down or spiral improves compression ratios. Therefore our next tests will use spiral or up-down stage.

4.5 GST Impact

There are several transforms that can be used in GST stage. In Section 3.4 presents a few of these algorithms. Burrows and Wheeler themselves proposed MTF as the second stage of BWCA, and some variants of this stage have been developed to improve BWCA performance. Furthermore, this section discuses more the efficiency of each transform to the BWCA method. This empirical studies use BWCA chain in Figure 4.8 and scan image in spiral or up-down.

Table 4.12 and Table 4.13 in the 2^{nd} and the 3^{rd} columns give the results of BWT method for image compression without and with MTF. Here we can see that MTF part is significant. The compression ratios decrease for 6 images. Nevertheless, using scan image up-down, the average of compression ratios increase of approximately 3% for 10 images presented and 4% for the total data-base. And for scan image spiral, the compression ratio increases around 2% for 10 images and also 4% for total image data-base.

Table 4.12: Comparative results of MTF and its variants using up-down reordering [70].

Image	no-MTF	MTF	M1FF	M1FF2	Ts(0)	Bx3	Bx5	Bx6	FC	WFC	AWFC	IFC
Hand1	2.616	2.547	2.541	2.538	2.624	2.649	2.666	2.670	2.663	2.659	2.706	2.656
Hand2	2.196	2.390	2.387	2.387	2.449	2.466	2.476	2.477	2.470	2.484	2.513	2.475
Hand3	1.669	2.253	2.247	2.246	2.319	2.345	2.363	2.368	2.375	2.352	2.410	2.347
Hand4	2.589	2.830	2.825	2.824	2.920	2.939	2.946	2.945	2.938	2.979	3.000	2.954
Head1	2.251	2.274	2.263	2.262	2.329	2.349	2.364	2.367	2.357	2.348	2.400	2.359
Head2	2.561	2.527	2.518	2.518	2.590	2.613	2.632	2.637	2.631	2.611	2.672	2.614
Head3	2.554	2.527	2.52	2.519	2.606	2.632	2.650	2.654	2.646	2.645	2.695	2.635
Head4	2.751	2.721	2.714	2.714	2.801	2.824	2.841	2.843	2.832	2.857	2.902	2.839
Pelvis1	1.978	1.842	1.835	1.835	1.888	1.905	1.919	1.922	1.911	1.898	1.908	1.909
Pelvis2	2.026	1.890	1.881	1.880	1.936	1.952	1.966	1.969	1.961	1.951	1.950	1.960
Av.10	2.319	2.380	2.373	2.372	2.446	2.467	2.482	2.485	2.478	2.478	2.516	2.475
Av.100	2.475	2.577	2.570	2.570	2.654	2.679	2.696	2.699	2.694	2.694	2.724	2.687

Table 4.13: Comparative results of MTF and its variants using spiral reordering[70].

Image	no-MTF	MTF	M1FF	M1FF2	Ts(0)	Bx3	Bx5	Bx6	FC	WFC	AWFC	IFC
Hand1	2.616	2.536	2.530	2.527	2.614	2.639	2.657	2.662	2.655	2.648	2.701	2.646
Hand2	2.193	2.382	2.380	2.380	2.453	2.472	2.486	2.488	2.479	2.481	2.517	2.476
Hand3	1.657	2.221	2.214	2.214	2.284	2.307	2.326	2.330	2.338	2.319	2.375	2.312
Hand4	2.587	2.802	2.800	2.798	2.898	2.923	2.937	2.939	2.926	2.954	2.985	2.927
Head1	2.251	2.273	2.263	2.263	2.329	2.350	2.365	2.370	2.358	2.347	2.399	2.359
Head2	2.576	2.538	2.528	2.528	2.602	2.626	2.646	2.651	2.645	2.623	2.689	2.627
Head3	2.578	2.563	2.555	2.553	2.642	2.669	2.689	2.693	2.685	2.684	2.736	2.674
Head4	2.784	2.764	2.757	2.756	2.848	2.874	2.894	2.898	2.885	2.905	2.957	2.887
Pelvis1	1.974	1.835	1.828	1.828	1.882	1.899	1.913	1.917	1.905	1.892	1.902	1.903
Pelvis2	2.018	1.876	1.867	1.867	1.922	1.939	1.954	1.957	1.948	1.937	1.932	1.946
Av.10	2.323	2.379	2.372	2.371	2.447	2.470	2.487	2.491	2.482	2.479	2.519	2.476
Av.100	2.477	2.575	2.568	2.567	2.651	2.677	2.694	2.698	2.692	2.691	2.721	2.685

The results for second family of MTF (M1FF and M1FF2) do not increase BWCA performance. The 4^{th} and 5^{th} columns of Table 4.12 and Table 4.13 determine these results. Meanwhile, these results are better than BWCA chain without MTF. Therefore, GST stage is really needed in BWCA method.

The GST step of Albers is called Time Stamp (TS) [9] provides better performances than the previous GST. Then, Chapin has been improved this algorithm, called a "Best x of $2x - 1$" algorithm [19]. The results of these transforms are shown in 6^{th} to 9^{th} columns of Table 4.12 and Table 4.13. The compression ratios using these transforms

are better than those using MTF algorithm. The compression performance average increases till 9% for a "$Bx6$" algorithm using image scan up-down for the whole tested images.

Other variant of GST that is quite different because it has a computation process to count the symbols ranking, called Frequency Counting (FC) and its variants, such as WFC and AWFC. These transforms are more complex than previous GST and most of them do not determine better compression ratios than a "$Bx6$". It can be seen in the 10^{th} and 11^{th} columns of Table 4.12 and Table 4.13. The improvement compression performance is obtained by a WFC modified called Advanced Weighted Frequency Count (AWFC). It counts a weight of symbol distribution. It is more complex than WFC, but it gives better compression ratios. These results can be seen in the 12^{th} column of Table 4.12 and Table 4.13. Abel presents other GST method called Incremental Frequency Count (IFC) [3]. It is quite similar to WFC, but it is less complex but more performance than WFC. The results of these transform can be seen in 13^{th} column in the Table 4.12 and Table 4.13.

Table 4.12 and Table 4.13 show that AWFC is slightly better than a "$Bx6$". But AWFC is more complex than a "$Bx6$" transform. It takes more time to count the distance of symbol.

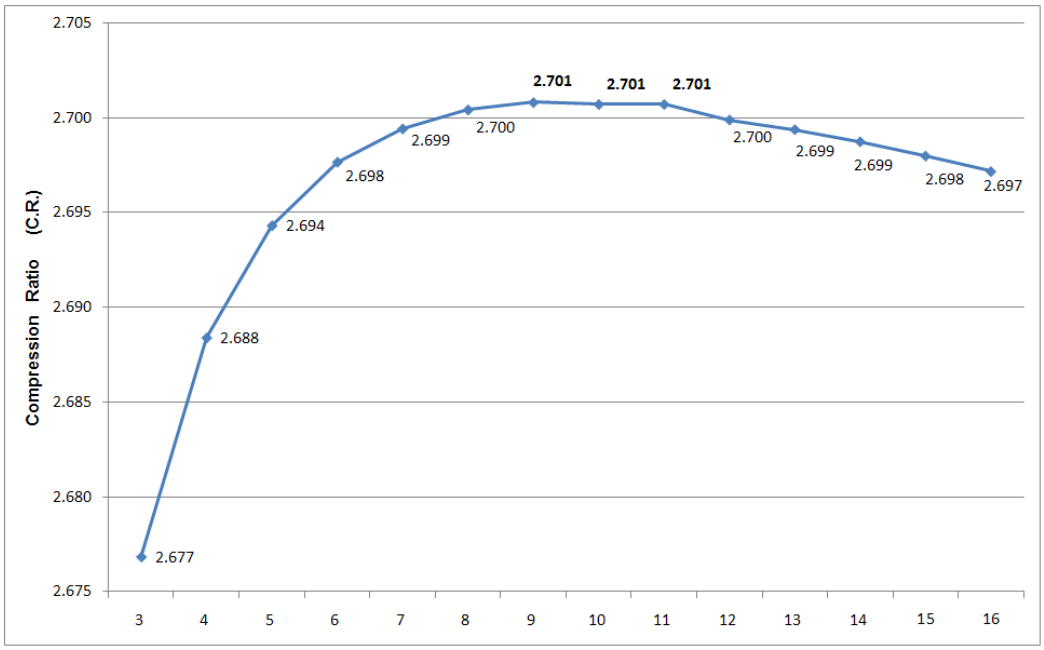


Figure 4.9: The "Best x of $2x-1$" analysis for x=3,4,5,...,16.

Furthermore, Figure 4.9 shows the performances of "Best x of $2x-1$" for $x=3$

to $x = 16$. The pick values of compression efficiency are for $x = 9$, $x = 10$ and $x = 11$. Otherwise, the different is not really significant. For simplifying the simulation, "$Bx11$" and AWFC will be used in the next simulation to analyze the Entropy Coding stage and to improve BWT performances.

4.6 Data Coding Impact

Data Coding (DC) is in charge of the compression process of BWCA chain. The previous stage transforms the input data to enable compression improvement to be obtained on the DC stage. The studies presented in the previous paragraphs enable the number of these stages in charge of data transformation to be limited. Two pre-processing have been selected: reordering (Up-down or Spiral), GST (Best x of $2x$-1 for x=11 and AWFC). Moreover, in the following sections, we discuss about the impact of the DC methods classically used into the BWCA compression scheme. Hence, we focus on the impact of RLE and statistical coding.

4.6.1 Run Length Encoding Impact

Run Length Encoding (RLE) aims to shrink long runs GST results. Several RLE variants were described (see Section 3.5.1). We will review RLE of Lehmann et al. [41] named Run length Encoding two Symbols (RLE2S) since it have been implemented in medical image compression (see Section 3.7.2.3). RLE2S separates the data stream and the runs so it does not interfere with the main data coding.

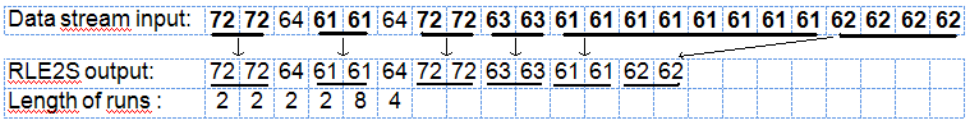


Figure 4.10: Example of Run Length Encoding two symbols (RLE2S) .

Two or more consecutive symbols are transformed in two symbols in the data stream. The associated length is reported in a separated stream. If the symbol is not repeated, therefore it is included in the data stream but nothing in the stream of the runs as shown in Figure 4.10.

Schemes based on the state of the art approach slightly improved by including a pre-processing (4% on both chains). These chains represented our reference schemes. Thus, we combined Lehmann proposed method (see Figure 3.18) and our previous test

of Figure 4.8. Figure 4.11 presents the new scheme using RLE2S and is consider as the second method (M2). The 6^{th} to 11^{th} column of Table 4.14 presents compression performances for both method (M1 and M2). RLE2S increases compression efficiency systematically. The results of these two methods have a same tendency. CR improvement for different GST are similar, which are:

- M1-Bx11 to M2-Bx11 is also around 6.34%,
- M1-AWFC to M2-AWFC is around 5.17%.

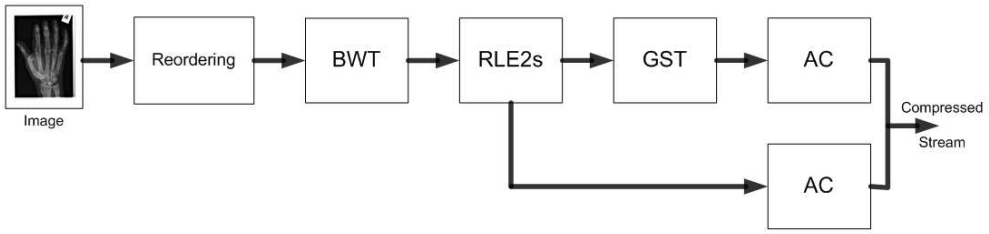


Figure 4.11: Second proposed method of BWCA (M2).

The best compression performances of M1 is obtained using AWFC as a GST stage, meanwhile M2 is using Bx11. Nevertheless, the margin of CR for these two variants of GST for the same chain (M1 or M2) are really minor, that is why we keep using these two GST algorithms for following tests.

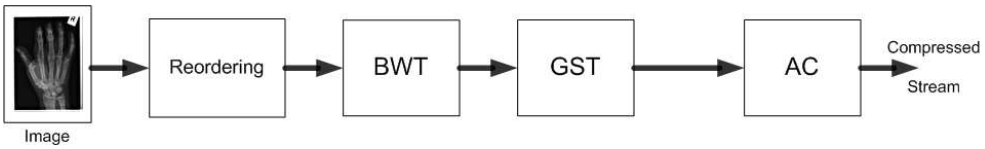


Figure 4.12: The third proposed method of BWCA (M3).

The RLE stage has been traditionally included in the BWCA schemes. This algorithm has been proved [44] to increase the performances of the text compression schemes based on the BWT. Lehmanm et al. have included this stage in their image compression scheme to improve the CR. Nevertheless, as the chain has been modified with the reordering stage, we test the impact of RLE on the chain by removing it (as shown in Figure 4.12 and called the 3^{rd} method or M3).

Table 4.14: M1-M4 results for IRMA images using Spiral reordering.

No.	Image category	Nb. of Image	JPEG 2000 (Jasper)	JPEG -Ls (HP)	Spi-Bwt-Gst-R0-Ac		Spi-Bwt-Rl2s-Gst-Ac		Spi-Bwt-Gst-__-Ac		Spi-Bwt-Gst-__-ModAc		Best Value
					Bx11	AWFC	Bx11	AWFC	Bx11	AWFC	Bx11	AWFC	
					M1		M2		M3		M4		
1	Hands primary	120	2.557	**2.587**	2.341	2.356	2.463	2.461	2.502	2.471	2.544	2.506	JPEG-LS
2	Hands secondary	120	3.134	3.209	2.934	2.972	3.105	3.106	3.166	3.136	**3.263**	3.215	M4-Bx11
3	Heads primary	120	**2.970**	2.939	2.665	2.685	2.813	2.824	2.847	2.813	2.877	2.837	JPEG2000
4	Heads secondary	120	2.575	2.665	3.130	3.173	3.371	3.364	3.536	3.464	**3.624**	3.534	M4-Bx11
5	Pelvis primary	120	**2.019**	2.016	1.893	1.881	1.987	1.987	1.995	1.954	2.001	1.957	JPEG2000
6	Pelvis secondary	120	2.788	2.863	2.955	2.979	3.165	3.154	3.289	3.212	**3.347**	3.252	M4-Bx11
7	Thoraces frontal primary	120	2.383	**2.396**	2.174	2.201	2.310	2.308	2.330	2.312	2.345	2.322	JPEG-LS
8	Thoraces frontal secondary	120	3.394	3.456	3.191	3.241	3.407	3.387	3.484	3.427	**3.529**	3.469	M4-Bx11
9	Thoraces lateral primary	120	2.826	**2.840**	2.548	2.566	2.712	2.710	2.746	2.718	2.758	2.726	JPEG-LS
10	Thoraces lateral secondary	120	3.171	3.232	2.958	2.991	3.155	3.143	3.257	3.212	**3.305**	3.251	M4-Bx11
	Av. comp. ratio	1200	2.782	2.820	2.679	2.705	2.849	2.844	2.915	2.872	**2.959**	2.907	M4-Bx11

Table 4.15: M1-M4 results for IRMA images using Ud-down reordering.

No.	Image category	Nb. of Image	JPEG 2000 (Jasper)	JPEG -Ls (HP)	Spi-Bwt-Gst-R0-Ac		Spi-Bwt-Rl2s-Gst-Ac		Spi-Bwt-Gst-__-Ac		Spi-Bwt-Gst-__-ModAc		Best Value
					Bx11	AWFC	Bx11	AWFC	Bx11	AWFC	Bx11	AWFC	
					M1		M2		M3		M4		
1	Hands primary	120	2.557	**2.587**	2.354	2.373	2.478	2.479	2.519	2.492	2.560	2.525	JPEG-LS
2	Hands secondary	120	3.134	3.209	2.940	2.979	3.112	3.114	3.172	3.144	**3.270**	3.222	M4-Bx11
3	Heads primary	120	**2.970**	2.939	2.647	2.669	2.793	2.795	2.826	2.794	2.857	2.818	JPEG2000
4	Heads secondary	120	2.575	2.665	3.083	3.132	3.318	3.317	3.473	3.413	**3.567**	3.484	M4-Bx11
5	Pelvis primary	120	**2.019**	2.016	1.893	1.881	1.986	1.986	1.995	1.954	1.999	1.955	JPEG2000
6	Pelvis secondary	120	2.788	2.863	2.941	2.968	3.148	3.139	3.269	3.197	**3.330**	3.237	M4-Bx11
7	Thoraces frontal primary	120	2.383	**2.396**	2.165	2.191	2.299	2.298	2.320	2.302	2.334	2.311	JPEG-LS
8	Thoraces frontal secondary	120	3.394	3.456	3.167	3.221	3.381	3.364	3.455	3.412	**3.504**	3.447	M4-Bx11
9	Thoraces lateral primary	120	2.826	**2.840**	2.541	2.559	2.704	2.702	2.739	2.710	2.749	2.718	JPEG-LS
10	Thoraces lateral secondary	120	3.171	3.232	2.959	2.994	3.155	3.145	3.257	3.214	**3.308**	3.254	M4-Bx11
	Av. comp. ratio	1200	2.782	2.820	2.669	2.697	2.837	2.834	2.902	2.863	**2.948**	2.897	M4-Bx11

The 10^{th} and 11^{th} columns of Table 4.14 present M3 results. These results are better than two previous tests (M1 and M2). In average the performances of M3 increase up to 8.83% and to 3.37% respectively compared with the M2 method and the JPEG−LS standard. Moreover the CR improve compared to M2 method for each categories of the image data-base. This method provide ever better performances than the JPEG2000 and JPEG−LS standards on four categories.

Table 4.15 presents also these methods (M1-M3) using Up-down reordering. However, these results are lower than spiral reordering technique. The results difference are low. Nevertheless, the two reordering techniques provide better performance than the existing compression standards.

The improvement have been obtained by including the reordering stage. This preprocessing enables the 2D image nature to be considered. Nevertheless, each new stage include in the scheme can modify the behavior of others ones. Therefore, the suppression of RLE stage in the modified Lehmann's scheme enables the performances of this chain to be improved. We propose to focus on the EC stage in the following paragraph.

4.6.2 Statistical Coding

Burrows and Wheeler suggest using Arithmetic Coding for the last step [18]. However, we cannot use a simple arithmetic coder with a common order-n context. GST stage increases the probability of lower input values (and so it decreases the probability of the higher values). Our previous test have been confirmed it by comparing the classic BWCA chain that uses AC with BZIP2 and BW94 that use Huffman Coding in the last stage (see Section 4.3). Some authors have been stated that a specific AC should be use [26, 27, 3].

The coding type of the GST output inside the EC stage has a strong influence on the compression rate. It is not sufficient to compress the index stream just by a simple arithmetic coder with a common order-n context. The index frequency of the GST output has a nonlinear decay. Even after the use of an RLE2S stage, the index 0 is still the most common index symbol on average. As discussed by Fenwick [26], a hierarchical coding model offers better compression results for skew distributions. Similar to the model of Fenwick, this implementation uses a hierarchical coding model, consisting of three levels, for the index stream of the GST output. The first level encodes the most common indices, which range from 0 to 2, and which make up more than 50% of the output indices. Larger indices are encoded by all three levels. If the

current index is smaller than 3, the information encoded by the first level is sufficient to uniquely decode the index and the following levels are not needed. If the index is larger than or equal to 3, an escape symbol is output in the first level and the index is categorized into seven disjoint groups. The third level handles the offset of the current index in each group. In order to exploit the properties of the GST output indices in an efficient manner, the first level uses three binary arithmetic coders instead of one arithmetic coder with an alphabet of size of four.

Therefore, Fenwick proposes an Arithmetic Coder with hierarchical coding model for this skew distribution. We also tested these coding and referred as M4 in Table 4.14. M4 is respectively similar to the M3 scheme but using modified Arithmetic Coding. Therefore, the first one uses a traditional AC and the second one, a modified AC.

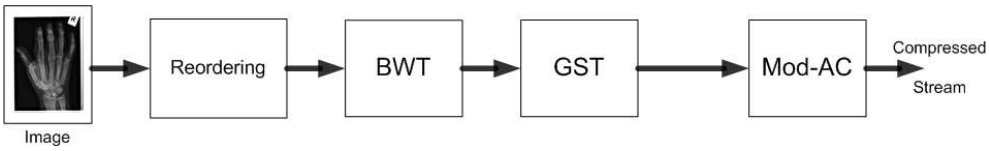


Figure 4.13: The fourth proposed method of BWCA (M4).

The results presented in Table 4.14 show that M4 performances are slightly improved in average and therefore to be considered. Average gain of M4 compared with M3 is around 2%, meanwhile it is around 4% compared with M1. The average CR of M4 using Bx11 is the best CR than other techniques. The compression performance is improved with M4 compared to `JPEG-LS` and `JPEG2000`, 5 and 6% can respectively be obtained. This significant result is obtained considering the average CR. Another significant result can be observed with the detail analysis which consider particularly each category of IRMA's data-base.

Detailed analysis, the 1200 images of the data-base are split into 10 categories as shown in Section 4.2. The Table 4.16 details the results obtained for M4-Bx11 method for each categories. It also enables a comparison performance with `JPEG-LS` and `JPEG2000`.

Table 4.16 presents the gain of each method for IRMA's data-base. The 4^{th}, 5^{th}, and 6^{th} columns are the number of the best result obtained by the corresponding method (i.e. the head of the column), meanwhile the 7^{th}, 8^{th}, and 9^{th} columns are the percentage of previous values.

For 1200 images, 565 images obtain better CR using `JPEG-LS`, 399 using M4-Bx11, and 236 using `JPEG2000`, or in the percentage, we have:

- 19.67% of images give better CR using `JPEG2000`,

Table 4.16: Results comparison of M4-Bx11, JPEG−LS and JPEG2000.

No.	Images category	Nb. of Image	Best Value					
			M4-Bx11	JPEG 2000	JPEG −LS	M4-Bx11	JPEG 2000	JPEG −LS
1	Hands primary	120	15	29	76	12.50	24.17	63.33
2	Hands secondary	120	45	0	75	37.50	0.00	62.50
3	Heads primary	120	2	85	33	1.67	70.83	27.50
4	Heads secondary	120	108	0	12	90.00	0.00	10.00
5	Pelvis primary	120	4	69	47	3.33	57.50	39.17
6	Pelvis secondary	120	90	4	26	75.00	3.33	21.67
7	Thoraces frontal primary	120	1	26	93	0.83	21.67	77.50
8	Thoraces frontal secondary	120	73	0	47	60.83	0.00	39.17
9	Thoraces lateral primary	120	0	22	98	0.00	18.33	81.67
10	Thoraces lateral secondary	120	61	1	58	50.83	0.83	48.33
	Total	1200	399	236	565	33.25	19.67	47.08
	Total all		1200 images			100%		

- 33.25% of images give better CR using M4-Bx11, and
- 47.08% of images give better CR using JPEG−LS.

The most favorable result for M4 is the 4^{th} category (Head Secondary Image) whereas 90% images obtain better CR than JPEG−LS and JPEG2000. But, M4-Bx11 can not get better CR in Thoraces Lateral Primary category.

There are more images that give better CR using JPEG−LS than using M4-Bx11. However, the best average CR for all IRMA's data-base is obtained using M4-Bx11.

Table 4.17 is details these gains by presented the difference of CR between the best method and M4-Bx11.

Table 4.17: Detail results of M4-Bx11.

No.	Images category	Nb. of Image	Max. Loss of M4-Bx11	Av. Loss of M4-Bx11	Max Gain of M4-Bx11
1	Hands primary	120	0.251	0.055	0.082
2	Hands secondary	120	**0.465**	*0.146*	2.143
3	Heads primary	120	0.241	0.095	0.029
4	Heads secondary	120	0.301	*0.018*	2.053
5	Pelvis primary	120	0.069	0.021	0.063
6	Pelvis secondary	120	0.283	0.032	**2.532**
7	Thoraces frontal primary	120	0.189	0.054	0.010
8	Thoraces frontal secondary	120	0.282	0.043	0.948
9	Thoraces lateral primary	120	0.182	0.084	*-0.001*
10	Thoraces lateral secondary	120	0.432	0.092	1.589

The 4^{rd} column represents maximum gap discrepancy of CR between JPEG2000 or JPEG−LS and M4-Bx11 of an image for each category, that is, the maximum among the following two values:

- if the best CR is obtained by JPEG−LS, the difference of CR then between JPEG−LS and M4-Bx11, or

- if the best CR is obtained by JPEG2000, the difference of CR then between JPEG2000 and M4-Bx11.

The maximum discrepancy is obtained by an image in the Hand Secondary category which is 0.465. The detailed CR of this image for each technique are:

- JPEG2000 for 4.509,
- JPEG−LS for 4.617, and
- M4-Bx11 for 4.151.

Other values represent that M4-Bx11 lose values are not really significant. Moreover, the 6^{th} column of Table 4.17 presents the maximum improvement value of M4-Bx11 for an image in each category. It can reach to 2.532 for a Pelvis Secondary image, where the detailed CR for each technique are:

- 2.015 for JPEG2000,
- 2.249 for JPEG−LS, and
- 4.781 for M4-Bx11.

The average CR for the category of this image is also large compared with other standards (3.534 using M4-Bx11 and 2.665 for JPEG−LS).

However, the minimum value is -0.001 where M4-Bx11 can not provide better CR compared with JPEG−LS and JPEG2000.

Moreover, the 5^{th} column of Table 4.17 presents the difference average between the maximum CR value and M4-Bx11 (in where M4-Bx11 method can not obtained better performances). These values are low, in other words, the average lose value of M4-Bx11 method for IRMA's data-base is low.

Detailed information about these results are presented by Table 4.18. M4-Bx11 is compared separately with JPEG−LS and then with JPEG2000.

Furthermore, the comparison of M4-Bx11 with each image compression standards is presented in Table 4.18.

The 3^{th} till 6^{th} columns present CR comparison of M4-Bx11 and JPEG−LS. For all IRMA Corpus, there are 787 images that provide better CR using JPEG−LS and 413 images using M4-Bx11. In percentage, 65.58% of images are obtained better CR using JPEG−LS, so 34.42% using M4-Bx11. Contrarily, the average CR of JPEG−LS for 1200 images is only 2.820 that is slightly lower than M4-Bx11 which is 2.959 (see

Table 4.18: Results summary of M4-Bx11.

No.	Image Category	compare with	M4-Bx11			JPEG -LS	compare with	M4-Bx11			JPEG 2000
			Nb. of Image		CR	CR		Nb. of Image		CR	CR
			%	num.				%	num.		
1	Hands primary	> J-LS	15.8	19	2.844	2.814	> J2K	37.5	45	2.829	2.789
		< J-LS	84.2	101	2.488	2.544	< J2K	62.5	75	2.373	2.418
		Total	100.0	120	2.544	2.587	Total	100.0	120	2.544	2.557
2	Hands secondary	> J-LS	37.5	45	3.473	2.941	> J2K	45.0	54	3.446	2.936
		< J-LS	62.5	75	3.137	3.370	< J2K	55.0	66	3.113	3.296
		Total	100.0	120	3.263	3.209	Total	100.0	120	3.263	3.134
3	Heads primary	> J-LS	1.7	2	3.033	3.013	> J2K	1.7	2	3.033	3.013
		< J-LS	98.3	118	2.875	2.938	< J2K	98.3	118	2.875	2.969
		Total	100.0	120	2.877	2.939	Total	100.0	120	2.877	2.970
4	Heads secondary	> J-LS	90.0	108	3.672	2.586	> J2K	90.0	108	3.672	2.495
		< J-LS	10.0	12	3.195	3.372	< J2K	10.0	12	3.195	3.297
		Total	100.0	120	3.624	2.665	Total	100.0	120	3.624	2.575
5	Pelvis primary	> J-LS	7.5	9	2.331	2.310	> J2K	3.3	4	2.647	2.637
		< J-LS	92.5	111	1.974	1.992	< J2K	96.7	116	1.978	1.997
		Total	100.0	120	2.001	2.016	Total	100.0	120	2.001	2.019
6	Pelvis secondary	> J-LS	75.0	90	3.457	2.770	> J2K	81.7	98	3.450	2.744
		< J-LS	25.0	30	3.015	3.143	< J2K	18.3	22	2.888	2.985
		Total	100.0	120	3.347	2.863	Total	100.0	120	3.347	2.788
7	Thoraces frontal primary	> J-LS	5.0	6	2.339	2.336	> J2K	1.7	2	2.257	2.253
		< J-LS	95.0	114	2.345	2.400	< J2K	98.3	118	2.346	2.385
		Total	100.0	120	2.345	2.396	Total	100.0	120	2.345	2.383
8	Thoraces frontal secondary	> J-LS	60.8	73	3.486	3.294	> J2K	76.7	92	3.492	3.294
		< J-LS	39.2	47	3.596	3.706	< J2K	23.3	28	3.649	3.724
		Total	100.0	120	3.529	3.456	Total	100.0	120	3.529	3.394
9	Thoraces lateral primary	> J-LS	0.0	0	0.000	0.000	> J2K	1.7	2	1.995	1.979
		< J-LS	100.0	120	2.758	2.840	< J2K	98.3	118	2.771	2.840
		Total	100.0	120	2.758	2.840	Total	100.0	120	2.758	2.826
10	Thoraces lateral secondary	> J-LS	50.8	61	3.416	3.092	> J2K	61.7	74	3.415	3.102
		< J-LS	49.2	59	3.190	3.377	< J2K	38.3	46	3.128	3.282
		Total	100.0	120	3.305	3.232	Total	100.0	120	3.305	3.171
	IRMA Corpus	> J-LS	34.4	413	3.443	2.868	> J2K	40.1	481	3.425	2.869
		< J-LS	65.6	787	2.705	2.795	< J2K	59.9	719	2.648	2.723
		Total	100.0	1200	2.959	2.820	Total	100.0	1200	2.959	2.782

Nb.: number; num. : numerical.

Table 4.14). Therefore, the 3^{rd} column of Table 4.18 split number of image that provide better and worsen CR between M4-Bx11 and JPEG−LS in the 4^{th} column. The same purpose is done for the 8^{th} column result that compares M4-Bx11 and JPEG2000 results.

For the first category, M4-Bx11 gives better CR for only 19 images. And the differences of CR for these images between M4-Bx11 and JPEG−LS is not really crucial. For these 19 images, the improvement of M4-Bx11 is only 1.04% compared with JPEG−LS. Meanwhile, M4-Bx11 performance decrease around 2% for the rest of images in this category. CR of M4-Bx11 for 101 images is 2.488, meantime JPEG−LS is 2.544. Otherwise for all images in this category, JPEG−LS CR improves around 1.66%, which is really trivial. Differently with other category where M4-Bx11 performances are significantly improved comparing with JPEG−LS. There are 5 categories where M4-Bx11 provide better performances. And the improvement of M4-Bx11 for a few secondary categories are really crucial. Especially for Heads and Pelvis Secondary categories. In Heads Secondary category, there are 108 images provide better performances than JPEG−LS, where CR for:

- JPEG−LS is 2.586 and
- M4-Bx11 is 3.672.

CR differences for these images is around 1.086. In other words, M4-Bx11 improves around 41.98% than JPEG−LS. Moreover, JPEG−LS performances for the rest of images (12 images) in this category increase moderately. It is only 5.22% comparing with the M4-Bx11 improvement, 41.98%. The results of Pelvis Secondary category is close to Heads Secondary category. 75% of images in this category provide better CR than JPEG−LS. And the CR differences between the two methods are really crucial, around 25%. Nevertheless, the improvement of few images which give better CR using JPEG−LS is only 4%.

Moreover, the margin for other categories where M4-Bx11 performances is lower than JPEG−LS, is not really important. JPEG−LS can only improve around 7% for 75 images in the second category. Merely, better average CR for the whole images for this category is provided by M4-Bx11.

This empirical analysis is continued by comparing M4-Bx11 with JPEG2000. However, JPEG2000 results is less performance than JPEG−LS. The tendency of JPEG2000 is close to JPEG−LS. Similarly with JPEG−LS results, there are more images providing better CR using JPEG2000 than M4-Bx11. However, the average

CR of M4-Bx11 is better than JPEG2000. Average CR of M4-Bx11 improves around 6.38% compared by JPEG2000. Average CR of all directly digital images provide better CR using JPEG2000, nevertheless average CR for all secondary images give better CR using M4-Bx11.

The difference of CR between the two methods are really significant for secondary images and trivial for directly digital category as well. Figure 4.14 and Figure 4.15 present the gain in percent of M4-Bx11 comparing with JPEG−LS and JPEG2000 for each IRMA category.

M4-Bx11 that is based on combinatorial method provides much higher compression rate on some images. This fact has been observed on 40.08% of the tested images comparing with JPEG−LS. For these images, the CR is in average 20% higher than the JPEG−LS standard. Meanwhile, the other images (59.92%) provide a CR which is only lower than 3.33%.

The proposed method is a completely different approach compared to JPEG−LS or JPEG2000. Therefore, a significant improvement of CR is obtained on some images comparison with these two standards. This improvement is highly correlated to the image's nature.

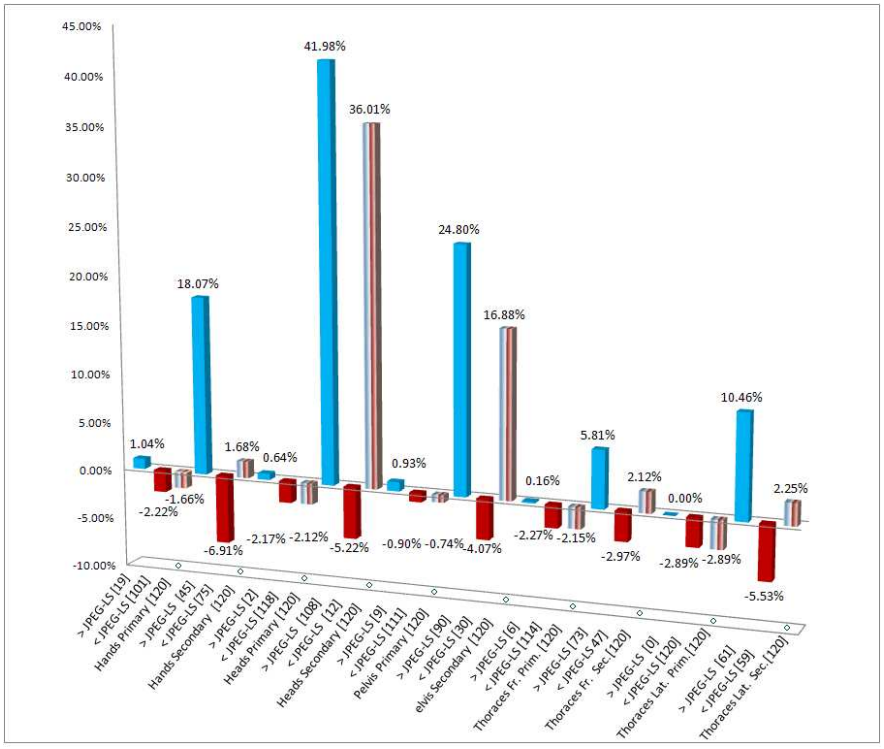


Figure 4.14: Gain percentage of M4-Bx11 and JPEG−LS.

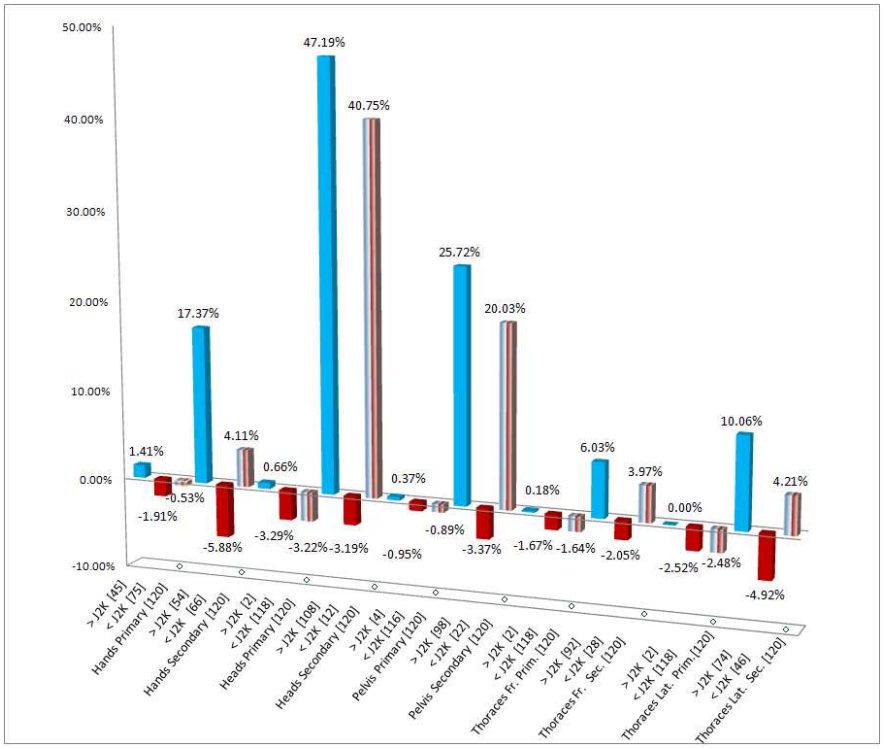


Figure 4.15: Gain percentage of M4-Bx11 and JPEG2000.

We consider developing an automatic image classification technique based on the image features (as the image texture) to split the images in two classes: those which provide the best results with the Burrows Wheeler compression scheme and those where the standard JPEG2000 is more efficient.

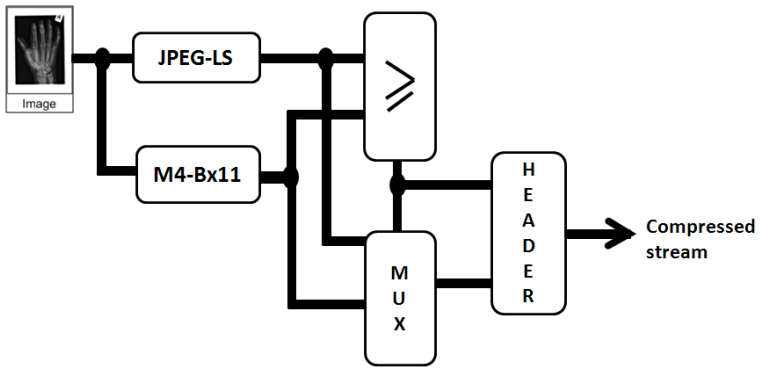


Figure 4.16: Our proposed method of JPEG−LS and M4-Bx11 combination.

This classification is not compulsory. Indeed the association of our compression scheme with a standard as JPEG−LS would represent a promising solution. A simple specific codec which embeds the two algorithms is presented in Figure 4.16. In this

codec, the two stream are compressed in parallel. The comparison is done on the size of two compressed output. The final compressed stream is the smallest output. A header is included to precise which method has been used.

4.7 Nearly Lossless BWCA Compression

Lossless compression techniques ensure complete data fidelity but with low compression ratios, while the lossy techniques aim to achieve much higher compression ratios by allowing some acceptable degradation in the recovered image. However, the medical professionals have been reluctant using lossy compression since any information loss or error caused by compression process could influence clinical diagnostic decisions and thus legal challenge could be raised. The medical community has therefore instead relied on lossless compression methods, though efficient compression techniques are highly needed for medical applications. The main purpose is to maximize compression while maintaining clinical relevance.

A possible solution to the above dilemma is to offer hybrid compression schemes. A general theme is to provide lossless compression for diagnostically important regions (Regions of Interest or ROI), while allowing lossy compression for other regions that are unimportant regions. Such hybrid compression can also be referred to regionally lossless coding or diagnosis lossless compression [13].

We also propose this technique (hybrid lossless and lossy method) to improve BWCA compression efficiency. Some researchers have investigated and proposed various hybrid lossless and lossy medical image compression techniques [80, 13].The results of some proposed methods are different. The choice of ROI techniques will gives different compression Ratios. It depends on segmentation process to improve image compression performances. ROI process can be manual or automatic. The aim of ROI depends on its applications, that can be a rough background-foreground distinction or a more focused segmentation.

Our empirical aims to improve the compression efficiency. The segmentation process as the pre-processing part of BWCA, is done by Open Computer Vision (OpenCV).

OpenCV is developed by Intel as an open source computer programming library. It supports vision computer applications that has many functions for capture, analysis and manipulation of visual data.

We use OpenCV library to distinct the background and the foreground of an im-

age. First process is to obtain the image background as seen in Figure 4.17. There are some of OpenCV library for segmentation process. This process involves separating an image into regions or contours depending on its applications. General region segmentation is identifying by its common properties, such as intensity. A simple way to segment such a region is through "thresholding" or separation of light and dark regions. "Thresholding" creates binary images by turning all pixels below a threshold value to zero and all pixels above that threshold to one.

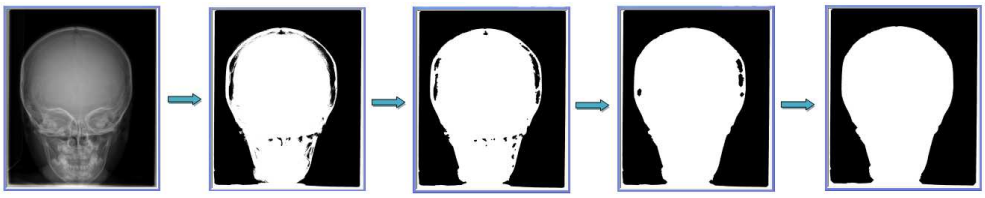


Figure 4.17: The background segmentation process.

However, this process consider only the intensity. Therefore, the extraneous pixels that are part of the undesired region aren't excluded. Hence, an ideal thresholding are developed to allow the threshold itself smoothly vary across the image. There are several types of thresholding functions in OpenCV. In this case the function determines the optimal threshold value using Otsu's algorithm [53]. The function returns the computed threshold value. Figure 4.17(b) shows the results of Otsu's method. Unfortunately, there are still several holes in the segmented image. Image morphology is commonly used to reduce these holes. It uses dilation and erosion process to reduce the holes in the ROI and morphological closing and opening operation is used to get rid of these holes as seen in Figure 4.18 and Figure 4.19. This process needs several iterations to eliminate all the holes as seen Figure 4.17(c), (d), and (e).

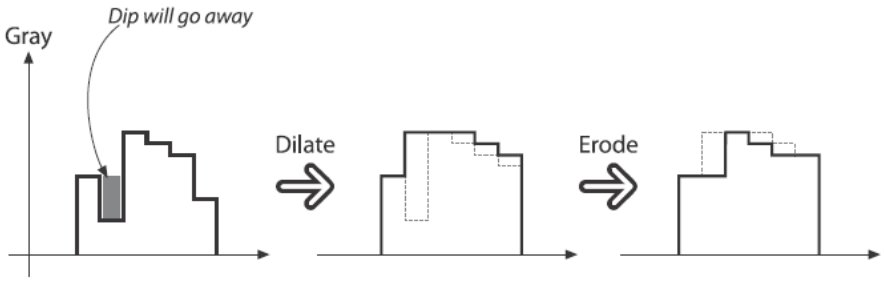


Figure 4.18: Morphological closing operation: the downward outliers are eliminated as a result [17].

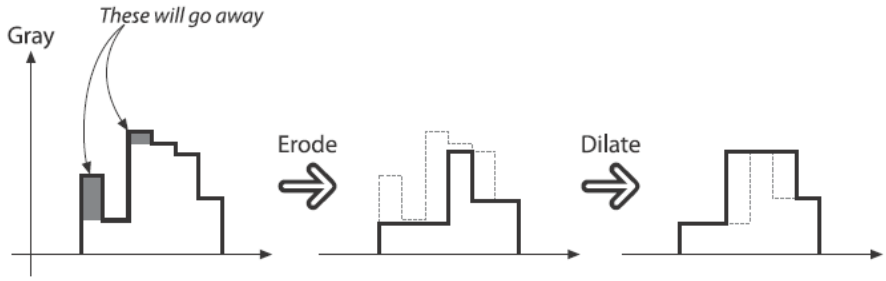


Figure 4.19: Morphological opening operation: the upward outliers are eliminated as a result [17].

However, segmentation process is not effective for all of the image. It depends on image nature. In this case, the segmentation aims to distinct background and foreground of an image. Some images are not important to do the distinction since they don't have or only have limited background such as Chest Frontal and Lateral images in Figure 4.1.

Table 4.19: Compression ratios of nearly-lossless using M4-Bx11.

No.	Image	Nearly-Lossless			Lossless		
		M4-Bx11	JPEG 2000	JPEG -LS	M4-Bx11	JPEG 2000	JPEG -LS
1	Hands prim.	8.828	7.817	8.732	4.900	5.001	5.152
2	Hands sec.	7.175	6.053	6.800	3.721	2.222	2.325
3	Heads prim.	4.103	4.324	4.365	3.589	3.805	3.751
4	Heads sec.	3.269	3.528	3.674	2.391	2.651	2.693
5	Pelvis prim.	4.068	3.630	3.742	2.760	2.749	2.696
6	Pelvis sec.	12.222	10.893	11.257	4.781	2.015	2.249
7	Thoraces fr. prim.	2.690	2.641	2.687	2.350	2.351	2.340
8	Thoraces fr. sec.	4.146	4.329	4.441	3.814	4.017	4.096
9	Thoraces lat. prim.	3.157	3.245	3.318	2.896	3.020	3.078
10	Thoraces lat. sec.	3.341	3.685	3.827	2.923	3.237	3.356
	Av. 10	5.300	5.014	5.284	3.412	3.107	3.174
	Av. 20	4.962	4.737	4.943	3.277	2.839	2.895

The simulation results show that this technique improves BWCA compression efficiency systematically. It increases also CR of JPEG-LS and JPEG2000 as seen in Table 4.19. This empirical studies use 20 images of IRMA data-base from 10 categories. From 20 tested images, 10 images obtain better CR using JPEG-LS, 9 images using M4-Bx11, and only 1 image using JPEG2000. However, the CR difference between BWCA and JPEG-LS are not significant. The tested images are taken from the best and the the worst obtained CR of M4-Bx11 compare with JPEG2000 and JPEG-LS from previous results.

4.8 Bit Decomposition in BWCA

In Section 4.4.1 stated that BWT works better in a large file. Bai in [13] also stated that BWT achieves much higher compression ratios for bi-level (binary) images, because of symbol alphabets alleviation and repeated patterns. Therefore, we propose in this section to study the effect of symbols reduction by focussing on the data-format and arrangements.

The selected data-format aims to increase the redundancies. Mainly, it consists of splitting the pixel, which is coded on one byte, into two or several blocks. Bit decomposition is not just splitting the pixel's dynamic, but also to consider the reordering of different blocks. The data's redundancy is therefore increasing, nevertheless the compression effort must be higher as the input-file size is systematically larger. Several simulations are done to analyze the bit decomposition impact in BWCA, such as splitting a byte into 2 blocks of 5 bits for MSB and 3 bits for LSB, splitting a byte into 4 blocks of 2 bits BWCA input, etc. Each obtained block is formatted in a byte by pre-pending zeros. This data-expansion enables the algorithm programming to be simplified. We present several empirical studies of these techniques, respectively named T1 until T7.

The first simulation scheme T1 consists of splitting each byte (coding a pixel) into two blocks of 4 bits each: the Most Significants Bits (MSB) and the Least Significants Bits (LSB). As explained above, each block is extended with zeros, as described in the Figure 4.20. The input data size of BWCA is therefore doubled. It is not surprising that among the MSB, the redundancy is increasing since two adjacent pixels are often similar. By contrast the LSB part contains small neighborhood changes, and on its own, the content of this part can be considered as almost random information. Consequently, the compression of each part do not provide the same contribution in the output file. We have evaluate each contribution. The MSB part represents only in average 15% of the output file. The compression of the MSB part is the crucial part of the encoding with this data-arrangement.

The statistic of the input file provided to the arithmetic codec is obviously modified by the new data-arrangement. Therefore the impact of the data-arrangement should be analyzed according to the configuration of the arithmetic coding. For a fixed size of the output format, usually (32 bits-integer, 64-bits integer or 32-bits float), the number of symbols in the coded sequence alphabet, named D, enables the coding to data-redundancy to be adjusted. Figure 4.21 shows the evolution of all the proposed methods in function of D. The MSB part, which is the crucial part, looks very similar to

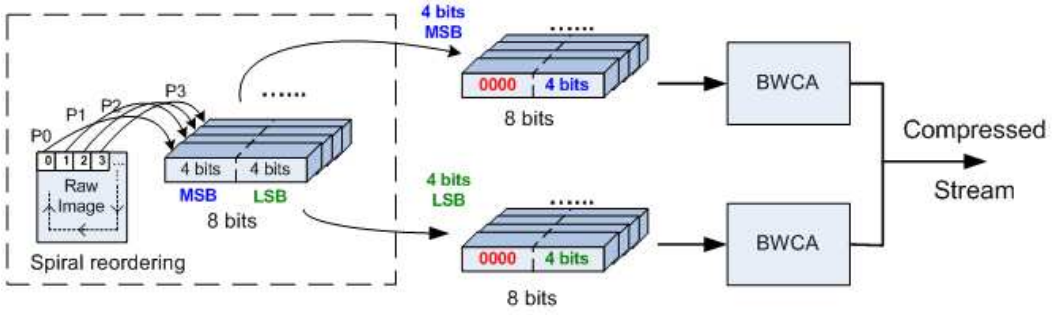


Figure 4.20: Method T1 with bit decomposition - MSB LSB processed separately.

the original image. Therefore, it is not surprising that the best performance is achieved using the same arithmetic codec parameters (D=1) than the original chain which does not used the bit decomposition. The general idea of the bit decomposition does not permit to reach the performance of the original compression scheme using 8-bits pixel as shown in the Figure 4.21. Nevertheless, we have extended this study. Indeed, this kind of approach, and particularly the decomposition in binary blocks would allow low level operations to be considered and therefore to propose hardware implementations. Consequently, we propose some modifications of the data-arrangement to show that the performance of the new approach can be improved.

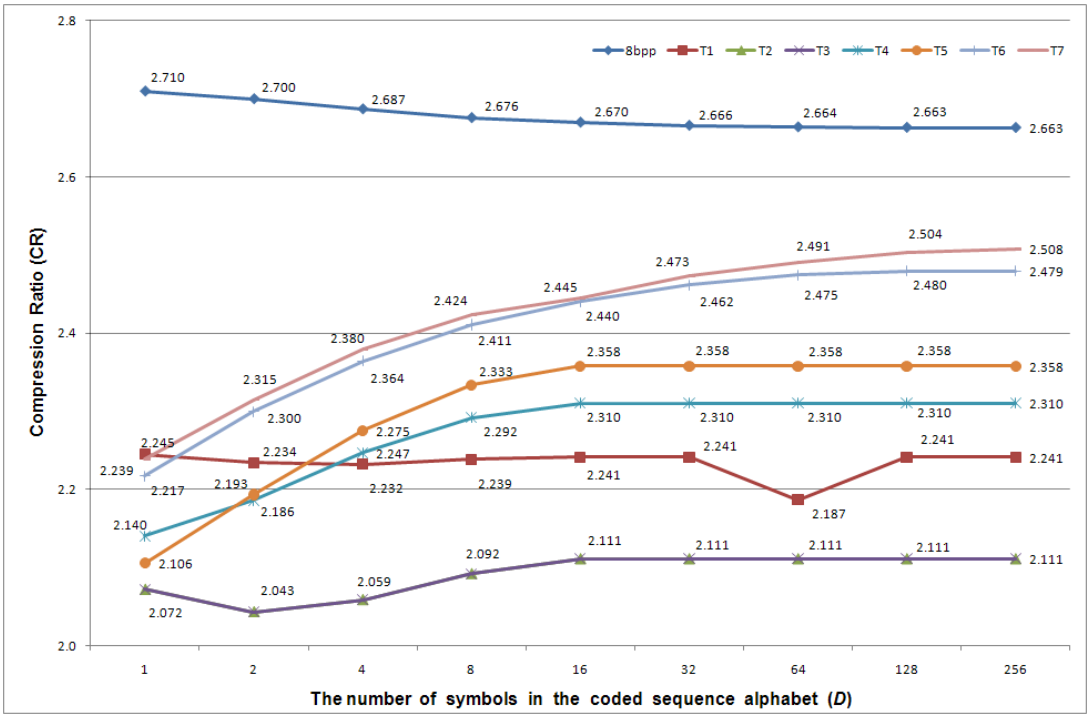


Figure 4.21: Average CR for different of AC parameters.

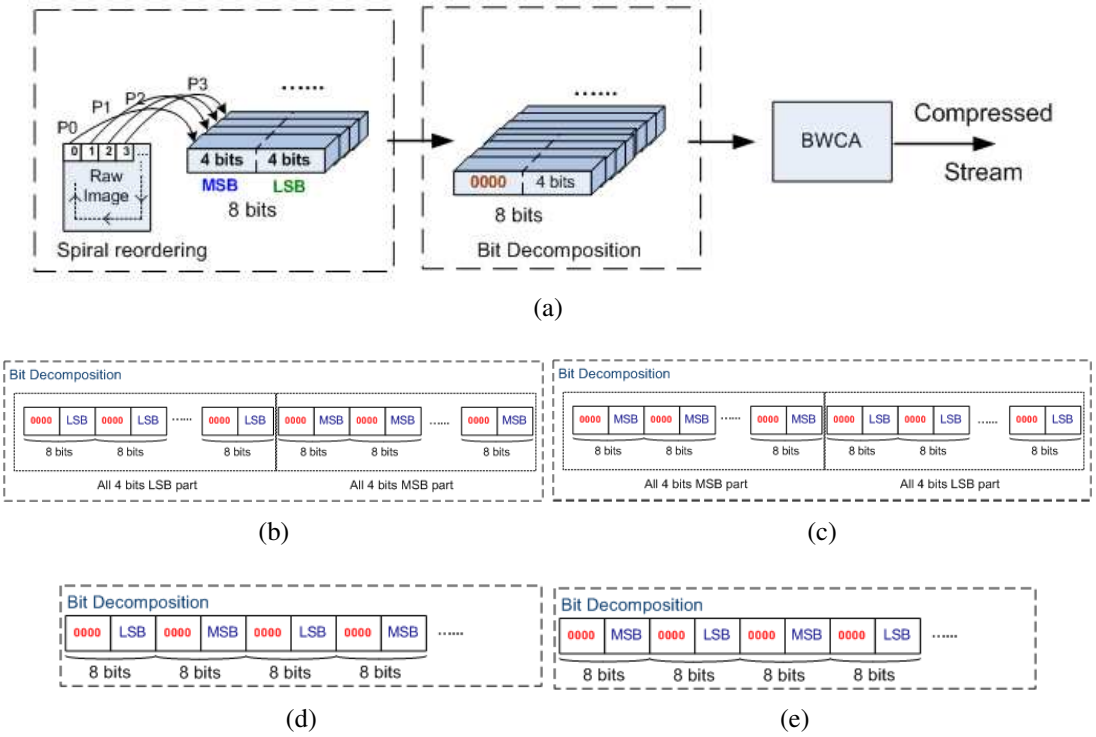


Figure 4.22: Four proposed methods by arranging MSB and LSB blocks.
(a) 4 bits decomposition by rearrange decomposition bit.
(b) T2: each LSB group is followed by its corresponding MSB group.
(c) T3: each MSB group is followed by its corresponding LSB group.
(d) T4: alternating LSB and MSB blocks, starting with LSB block.
(e) T5: alternating MSB and LSB blocks, starting with MSB block.

As we discussed earlier, in Section 4.4.1, the increasing of the image size enables, depending of the image's nature, better compression ratio to be obtained with BWCA scheme. Therefore, we propose to regroup the MSB and LSB parts in one pointer as an input of BWCA as depicted in the Figure 4.22. The MSB part can be followed by the LSB part, or the order is permuted and hence the LSB is placed before the MSB part. These two ways have been respectively implemented in methods T2 and T3 (see Figure 4.22b and Figure 4.22c). The results of these two approaches are, as expected, very similar.

Then, we propose to shuffle these data parts to improve the compression. The data-redundancy between LSB and MSB can be more exploited by the BWT stage. The principle of the two resulting methods named T4 and T5 is presented in Figure 4.22d and Figure 4.22e. In these two approaches, a block MSB alternates with a LSB. In T4, the input starts with a MSB block contrary to the T5 method where LSB precedes the corresponding MSB block.

By increasing D, the number of symbols in the coded sequence alphabet, the data-redundancy is better exploited by the arithmetic coding. The compression performances of the methods T2, T3, and particulary T4 and T5 are therefore increased as shown in Figure 4.21.

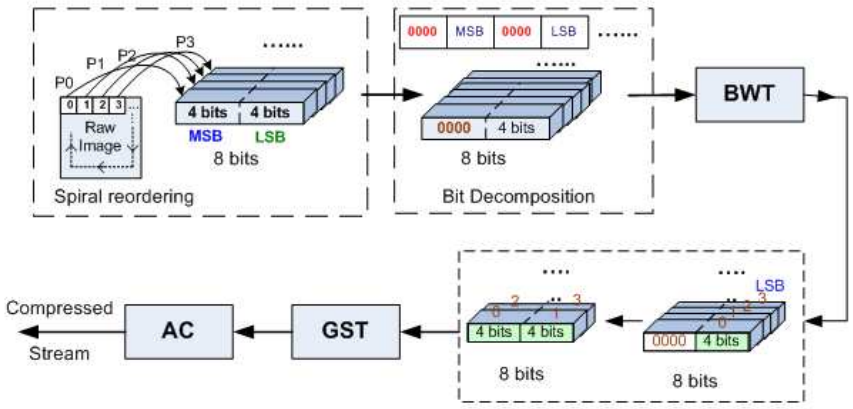


Figure 4.23: T6: 4 bits image decomposition by separating MSB and LSB consecutively. The two neighbor BWT output are combined into a byte.

Finally, the BWT stage regroups the similar symbols. After this stage the extension with the four zero in each sample can be suppressed. This elimination can occur either before or after the GST stage. Two methods named T6 and T7 aim to realize this operation as described in Figure 4.23 and Figure 4.24. The GST stage increases the redundancy therefore we recommend to suppress the additional zeros after this stage. The experimental results confirmed our assumption. The performance are increasing significantly in comparison with the previous methods. The parameters of the arithmetic codec should be considered to improve the compression ratio. We consider this bit decomposition still as an open problem. Also an adapted arithmetic coding must be considered in order to take advantage of data-structure. All of them need a deeper and more accurate analysis and are promising research directions.

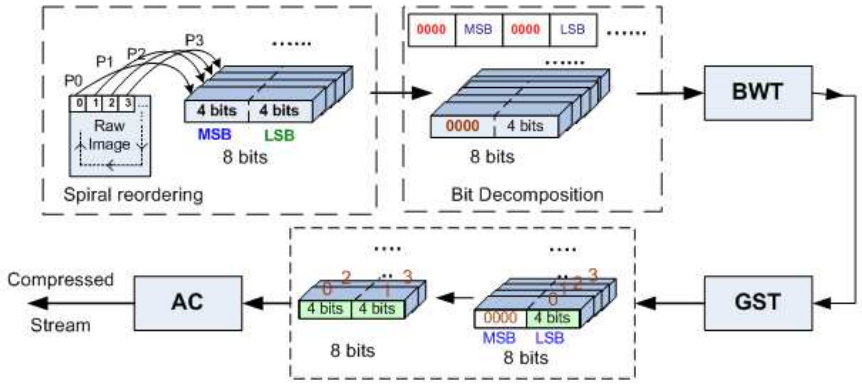


Figure 4.24: T7: 4 bits image decomposition by separating MSB and LSB consecutively. The two neighbor of GST output are combined into a byte.

Chapter 5

Conclusion and Perspectives

We presented the Burrows Wheeler Transform (BWT) state of the art in compression field, which originally was used in text compression, and propose a lossless image compression scheme based on this transform. We analyzed the impact of each stage of this chain that has important role to improve compression performance.

The compression scheme used in text compression should be largely modified. We should consider pre-processing, where this stage improves CR till 4%. Moreover, there are a few improvements of BWT post-processing which allow us to improve the classic chain of BWT.

On the 1200 IRMA medical images of the data-base, a substantial 5% improvement of the average compression ratio has been obtained with the proposed method compared to existing standards which are `JPEG 2000` or `JPEG-LS`. Moreover, this new approach based on combinatorial method provides much higher compression rate on some images. For instance, comparing with `JPEG-LS` which is well-known as the best lossless compression standard, 34.4% of images obtained better results using M4-Bx11 and the CR in average is 20% higher than this standard. Meanwhile, the other images (64.6%) provide an average CR which is only lower than 3.3%.

Therefore, the development of a specific codec, which embeds both algorithms working in parallel, could be considered to improve the current lossless image codec. The short compressed output file would be selected to be the final compressed bit-stream.

We proposed also nearly-lossless compression technique. The segmentation process is added as the pre-processing stage to obtain region of interest (ROI) of an image. The ROI part as an important is compressed losslessly, meanwhile the unimportant part (such as image background) is lossyly. This technique can doubled the compression

efficiency that definitely depends on image nature and method's segmentation.

There are others BWCA preprocessing steps that can increase compression performances. One of them is the bit decomposition of each pixel. We presented a preliminary study on it. However the corresponding results do not reach the previous performances, and thus these techniques are an interesting method to be exploited.

Another open problem is to characterize images that are more likely to be efficiently compressed using BWCA chain.

This studies described in this manuscript or document highlight some significant improvement in lossless image compression. Moreover, some developments, as specific coding stage particularly the bit decomposition approach, represent some potential improvements. With open problem still to be solved, the use of BWT in a compression scheme is, from our concern, a promising research way to be continued.

Bibliography

[1] Lukas corpus. http://www.data-compression.info/Corpora/LukasCorpus/index.htm.

[2] J. Abel. Improvements to the Burrows-Wheeler compression algorithm: After BWT stages, 2003.

[3] J. Abel. Incremental Frequency Count—a post BWT–stage for the Burrows-Wheeler compression algorithm. *Softw. Pract. Exper.*, 37(3):247–265, 2007.

[4] J. Abel. Post BWT stages of the Burrows-Wheeler compression algorithm. *Softw. Pract. Exper.*, 40:751–777, August 2010.

[5] T. Acharya and P. Tsai. `JPEG2000` *Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*. Wiley-Interscience, 2004.

[6] M. D. Adams and R. Ward. Wavelet Transforms in the `JPEG2000` Standard. In *In Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 160–163, 2001.

[7] D. Adjeroh, T. Bell, and A. Mukherjee. *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*. Springer US, june 2008.

[8] D. Adjeroh, Y. Zhang, A. Mukherjee, M. Powell, and T. Bell. DNA Sequence Compression Using the Burrows-Wheeler Transform. In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*, CSB '02, page 303, Washington, DC, USA, 2002. IEEE Computer Society.

[9] S. Albers. Improved randomized on-line algorithms for the list update problem. *SIAM J. Comput.*, 27(3):682–693, 1998.

[10] Z. Arnavut. Move-to-Front and Inversion Coding. In *Proceedings of the Conference on Data Compression*, DCC '00, pages 193–, Washington, DC, USA, 2000. IEEE Computer Society.

[11] Z. Arnavut and S. S. Magliveras. Block sorting and compression. In *Proceedings of the IEEE Data Compression Conference*, pages 181–190, 1997.

[12] R. Arnold and T. Bell. A corpus for the evaluation of lossless compression algorithms. *Data Compression Conference*, page 201, 1997.

[13] X. Bai, J. S. Jin, and D. Feng. Segmentation-based multilayer diagnosis lossless medical image compression. In *VIP '05: Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 9–14, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[14] H. Baik, D. S. Ha, H. Yook, S. Shin, and M. Park. Selective Application of Burrows-Wheeler Transformation for Enhancement of `JPEG` Entropy Coding, December 1999.

[15] B. Balkenhol and Y. M. Shtarkov. One attempt of a compression algorithm using the BWT. SFB343: Discrete Structures in Mathematics, Preprint, Faculty of Mathematics, University of Bielefeld, 1999.

[16] E. Bodden, M. Clasen, and J. Kneis. Arithmetic coding revealed - a guided tour from theory to praxis. Technical Report 2007-5, Sable Research Group, McGill University, May 2007.

[17] G. Bradski and A. Kaehler. *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]* . O'Reilly Media, 1. ed. edition, 2008.

[18] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, System Research Center (SRC) California, May 10, 1994.

[19] B. Chapin. Switching between two on-line list update algorithms for higher compression of Burrows-Wheeler transformed data. In *Data Compression Conference*, pages 183–192, 2000.

[20] B. Chapin. *Higher Compression from the Burrows-Wheeler Transform with New Algorithms for The List Update Problem*. PhD thesis, May 2001.

[21] C. Christopoulos, A. Skodras, and T. Ebrahimi. The `JPEG2000` Still Image Coding System: An Overview. *IEEE Transactions on Consumer Electronics*, 46:1103–1127, 2000.

[22] M. Ciavarella and A. Moffat. Lossless image compression using pixel reordering. In *ACSC '04: Proceedings of the 27th Australasian conference on Computer science*, pages 125–132, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[23] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley and Sons, 1991.

[24] S. Deorowicz. Second step algorithms in the burrows-wheeler compression algorithm. *Software Practice and Experience*, 2002.

[25] S. Deorowicz. *Universal lossless data compression algorithms*. PhD thesis, Silesian University of Technology Faculty of Automatic Control, Electronics and Computer Science Institute of Computer Science, 2003.

[26] P. Fenwick. Block sorting text compression–final report. Technical Report 130, Department of Computer Science, The University of Auckland, April 1996.

[27] P. Fenwick. Burrows–Wheeler compression with variable length integer codes. *Softw. Pract. Exper.*, 32(13):1307–1316, 2002.

[28] P. M. Fenwick. Burrows–Wheeler compression: Principles and reflections. *Theor. Comput. Sci.*, 387(3):200–219, 2007.

[29] P. Ferragina, R. Giancarlo, G. Manzini, and M. Sciortino. Boosting textual compression in optimal linear time. *ACM*, 52(4):688–713, July 2005.

[30] J. F. Gantz and S. Minton. The diverse and exploding digital universe an updated forecast of worldwide. 2008.

[31] J. Gilchrist. Parallel Compression with `BZIP2`. pages 559–564. IEEE Computer Society, 2004.

[32] M. Gormish and M. Marcellin. The `JPEG2000` Standard. (invited midday presentation), Data Compression Conference, 2000.

[33] H. Guo. *Wavelets for approximate Fourier Transformand data compression*. PhD thesis, Rice University, May, 1997.

[34] S. C. Shin M. S. Park H. K. Baik, H. G. Yook and D. S. Ha. A New Method to Improve the Performance of `JPEG` Entropy Encoding Using Burrows-Wheeler Transformation. Int. Symp. on Computer and Information Sciences, Oct 1999.

[35] P. G. Howard and J. S. Vitter. Fast and efficient lossless image compression. In *in Proc. 1993 Data Compression Conference, (Snowbird)*, pages 351–360, 1993.

[36] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952.

[37] ITU-T. Recommendation T.81 - digital compression and coding of continuous-tone still images. Geneva, Switzerland, September 1992.

[38] N. R. Jalumuri. A Study of Scanning Paths for BWT Based Image Compression. Master's thesis, West Virginia University, 2004.

[39] S. Kurtz and B. Balkenhol. Space efficient linear time computation of the Burrows and Wheeler-Transformation. In *complexity, Festschrift in honour of Rudolf Ahlswede's 60th Birthday*, pages 375–384, 1999.

[40] C. Lastri, B. Aiazzi, L. Alparone, and S. Baronti. Virtually Lossless Compression of Astrophysical Images. *EURASIP Journal on Applied Signal Processing*, 2005(15):2521–2535, 2005.

[41] T. M. Lehmann, J. Abel, and C. Weis. The impact of lossless image compression to radiographs. In *Proc. SPIE*, volume 6145, pages 290–297, March 2006.

[42] T. M. Lehmann, M. O. Güld, C. Thies, B. Fischer, K. Spitzer, D. Keysers, H. Ney, M. Kohnen, H. Schubert, and B. B. Wein. Content-based image retrieval in medical applications. 2004.

[43] X. Li and M. T. Orchard. Edge-directed prediction for lossless compression of natural images. *IEEE Transactions on Image Processing*, 10:813–817, 2001.

[44] G. Manzini. An analysis of the Burrows-Wheeler transform. *Journal of the ACM*, 48:2001, 1999.

[45] G. Manzini. Two space saving tricks for linear time lcp array computation. In *Proc. SWAT. Volume 3111 of Lecture Notes in Computer Science*, pages 372–383. Springer, 2004.

[46] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An Overview of `JPEG2000`. pages 523–544, March 2000.

[47] S. Marusic and G. Deng. Adaptive prediction for lossless image compression. *Signal Processing: Image Communication*, 17(5):363 – 372, 2002.

[48] B. Meyer and P. Tischer. TMW - a new method for lossless image compression. In *in Proc. of the 1997 International Picture Coding Symposium (PCS97*, pages 533–538, 1997.

[49] B. Meyer and P. Tischer. Extending TMW for Near Lossless Compression of Greyscale Images. In *Proceedings of the Conference on Data Compression*, pages 458–470, 1998.

[50] G. Motta, J. A. Storer, and B. Carpentieri. Lossless image coding via adaptive linear prediction and classification. *IEEE Proceedings of the IEEE*, 88(11):1790–1796, 2000.

[51] A. P. Neil. *Extending Lossless Image Compression*. PhD thesis, University of Cambridge, 2001.

[52] G. Nong, S. Zhang, and W. H. Chan. Linear suffix array construction by almost pure induced-sorting. *Data Compression Conference*, 0:193–202, 2009.

[53] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, January 1979.

[54] A. J. Penrose and N. A. Dodgson. Error resilient lossless image coding. In *In ICIP, Kobe*, pages 426–429, 1999.

[55] R. Pokrzywa and A. Polanski. BWtrs: A tool for searching for tandem repeats in DNA sequences based on the Burrows-Wheeler transform. *Genomics*, 96(5):316 – 321, 2010.

[56] A. Przelaskowski. Experimental comparison of lossless image coders for medical applications. In *Computer Vision and Graphics*. Springer Netherlands.

[57] I. Richardson. *Video Codec Design: Developing Image and Video Compression Systems*. Wiley, 2004.

[58] G. Roelofs. *PNG: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[59] D. Santa-Cruz and T. Ebrahimi. An Analytical Study of `JPEG2000` Functionalities. In *IEEE ICIP*, pages 49–52, 2000.

[60] D. Santa-Cruz, T. Ebrahimi, T. Ebrahimi A, J. Askelof, M. Larsson B, and C. A. Christopoulos B. JPEG2000 still image coding versus other standards. Howpublished = Public Draft FCD 14495, ISO / IEC JTC1 SC29 WG1 (JPEG / JBIG), 2000.

[61] G. Schaefer, R. Starosolski, and S. Ying Zhu. An evaluation of lossless compression algorithms for medical infrared images, 2005.

[62] J. Seward. BZIP2. (invited midday presentation), Data Compression Conference 2000.

[63] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, July, October 1948.

[64] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois, 1949.

[65] D. D. Shuai. Parallel lossless data compression: A particle dynamic approach. In *Proceedings of the 4th international conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications - with Aspects of Theoretical and Methodological Issues*, ICIC '08, pages 266–274, 2008.

[66] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG2000 still image compression standard. *IEEE Signal processing Magazine*, 18:36–58, 2001.

[67] S. W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, San Diego, CA, USA, 1997.

[68] R. Starosolski. Performance evaluation of lossless medical and natural continuous tone image compression algorithms. In *in Proc. SPIE Medical Imaging Conf*, page 5959, 2005.

[69] R. Starosolski. Simple fast and adaptive lossless image compression algorithm, 2006.

[70] E. Syahrul, J. Dubois, A. Juarna, and V. Vajnovszki. Lossless Compression Based on Combinatorial Transform: Application to Medical Images. In *International Congress on Computer Applications and Computational Science (CACS)*. IRAST, 2010.

[71] E. Syahrul, J. Dubois, and V. Vajnovszki. Combinatorial Transforms : Application in Lossless Image Compression. *Journal Of Discrete Mathematical Sciences & Cryptography*, November 2010.

[72] E. Syahrul, J. Dubois, V. Vajnovszki, T. Saidani, and M. Atri. Lossless image compression using Burrows Wheeler Transform (methods and techniques). In *SITIS '08*, pages 338–343, 2008.

[73] I. Tabus and J. Astola. Adaptive boolean predictive modelling with application to lossless image coding. In *In SPIE - Statistical and Stochastic Methods for Image Processing II*, pages 234–245, 1997.

[74] G. K. Wallace. The JPEG still picture compression standard. *Commun. ACM*, 34:30–44, April 1991.

[75] M. J. Weinberger, G. Seroussi, and G. Sapiro. LOCO-I: A low complexity, context-based, lossless image compression algorithm, 1996.

[76] M. J. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 9(8):1309–1324, 2000.

[77] G. Weinh, H. Stögner, and A. Uhl. Experimental study on lossless compression of biometric sample data. In *Image and Signal Processing and Analysis, 2009. ISPA 2009.*, pages 517–522, 2009.

[78] Y. Wiseman. Burrows-Wheeler based JPEG. *Data Science Journal*, 6:19–27, 2007.

[79] X. Wu. An algorithmic study on lossless image compression. In *DCC '96: Proceedings of the Conference on Data Compression*, page 150, Washington, DC, USA, 1996. IEEE Computer Society.

[80] M. J. Zukoski, T. Boult, and T. Iyriboz. A novel approach to medical image compression. *Int. J. Bioinformatics Res. Appl.*, 2:89–103, March 2006.