Improvements of Lossless Image Compression with Kernel based Global Structure Transform

By
M. Asif Ali
[2009-NUST-MS PhD-ComE-06]

Submitted to the Department of Computer Engineering
In partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Engineering

Advisor
Brig. Dr Muhammad Younus Javed

College of Electrical & Mechanical Engineering
National University of Sciences and Technology
2011

DECLARATION

We hereby declare that no portion of the work referred to in this Project Thesis has been submitted in support of an application for another degree or qualification of this of any other university or other institute of learning. If any act of plagiarism found, we are fully responsible for every disciplinary action taken against us depending upon the seriousness of the proven offence, even the cancellation of our degree.

COPYRIGHT STATEMENT


بسم الله الرحمن الرحيم

This thesis is dedicated to me

Acknowledgements

All praise to the Almighty Allah, the most Merciful and the most gracious one. Without whose help and blessings, I would not have been able to complete this research. Many thanks to my project supervisor, **Dr. Mohammad Younus Javed**, whose constant motivation, unflagging efforts and uninvolved words of wisdom ever proved a lighthouse for me; it was earnestly felt whenever we swayed. Despite his never ending assignments of university management, student counseling, project supervision and teaching, he did never mind whenever we went for an advice, within or without the time slot allocated for us.

Acknowledgement is also due to my teachers for dedicatedly instilling and imparting enlightenment to me during the course of studies and afterwards for our project. I am also very thankful to my parents for their tacit and avowed support, patience and understanding.

I would like to thank my friends who gave me confidence to face the difficulties of life. They all gave me good company and everlasting memories.

Abstract

Image compression addresses the problem of reducing the amount of data required to represent an image, whereas Lossless Image Compression refers to the application of the data compression in which the information of the original image is retained intact after compression hence reducing the storage space and transmission bandwidth. In our everyday life, images are processed, transmitted and stored digitally in various devices such as Digital cameras, iPods, medical images (Magnetic resonance imaging and Computer Tomography-scans) and digital telescopes.

The Burrows-Wheeler Compression Algorithm (BWCA) is the type of block sorting transform which was first introduced for lossless data compression, at first BWCA was used for text compression, but rapidly applications of BWCA was developed in the field of lossless image compression, many improvements has been offered by number of researchers since the creation of BWCA in 1994. Other studies treat the entropy coding of the data stream. Finally, many publications concern the middle part of the algorithm, where the BWT output symbols are prepared for the following entropy coding.

The Global Structure Transform (GST) is the stage in BWCA which transforms the local data in global context, such as in image compression the gray-levels can lay in variety of groups and bands, but the GST. Several researches have been conducted on techniques to improve the efficiency of BWCA; most of the studies focus on a specific stage improvement. We focus on enhancement of lossless image compression due to the aimed applications especially in the medical field; nevertheless this scheme can be applied for lossless image compression as well as for lossy image compression.

Table of Contents

DECLARATION **2**

COPYRIGHT STATEMENT **2**

ACKNOWLEDGEMENTS **5**

ABSTRACT **6**

TABLE OF CONTENTS **7**

LIST OF FIGURES **10**

LIST OF TABLES **12**

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO LOSSLESS IMAGE COMPRESSION **13**

1.2 DIGITAL IMAGE AND DIGITAL IMAGE PROCESSING **15**
1.2.1 Image formation model 16
1.2.2 Digitization and Representation of Image 18

1.3 IMAGE COMPRESSION **20**
1.3.1 Image and Data Compression fundamentals 20
1.3.2 Coding Redundancy 21
1.3.3 Interpixel Redundancy 22
1.3.4 Psychovisual Redundancy 23

1.4 SUMMARY **24**

CHAPTER 2

LITERATURE REVIEW

2.1 THE QUEST FOR HIGHER COMPRESSION **25**

2.2 BURROWS WHEELER TRANSFORM **26**

2.3 INVERSE BURROWS WHEELER TRANSFORM 29

2.4 BWT FOR IMAGES 33

2.5 GLOBAL STRUCTURE TRANSFORM 35
2.5.1 MTF Encoding 36
2.5.2 MTF Decoding 37
2.5.3 Higher Order Processing with MTF 38

2.6 ZERO RUN-LENGTH ENCODING (RLE-0) 40

2.7 ENTROPY CODER 41
2.7.1 Huffman coding 41
2.7.2 Arithmetic Coding 43

2.8 IMPROVEMENTS AND VARIANTS OF BWT OVER THE TIME 44

2.9 SUMMARY 45

CHAPTER 3

PROPOSED SCHEME

3.1 OVERVIEW 46
3.1.1 Memory Utilization 46
3.1.2 Speed 47

3.2 KMTF BASED BWCA 48
3.2.1 Data Characteristics in output of BWT 48
3.2.2 Kernel MTF Encoding 50
3.2.3 Reduced Encoding Map 51
3.2.4 Improvement in Entropy Coder stage 53

3.3 SUMMARY 55

CHAPTER 4 56

IMPLEMENTATION AND EXPERIMENTAL RESULTS 56

4.1 IMPLEMENTATION OVERVIEW 56

4.2 SUMMARY 65

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS 66

5.2 RECOMMENDATIONS AND FUTURE WORK 67

REFERENCES 68

List of Figures

Figure 1.1: Typical Process of image acquisition [7] 17

Figure 1.2: Process of sampling and quantization [7] 19

Figure 1.3: Two images with almost similar histograms [7] 23

Figure 1.4: Two distinctly quantized versions of vase in which left one is quantized 24

Figure 2.1: Typical BWCA technique as proposed in [2] 26

Figure 2.2: N x N matrix M of the all circularly shifted data 27

Figure 2.3: Sorted matrix M' 28

Figure 2.4: Only known data for the re-creation of matrix M 30

Figure 2.5: State of Matrix M after recovering the F string 31

Figure 2.6: Matrix M with the transformation vector 31

Figure 2.7: Pseudo-code for obtaining the original index for transforming the data into original format. 32

Figure 2.8: Final step in order to retrieve original data 32

Figure 2.9: (a) Sample image "Lenna" (b) The profile of image from point A to B illustrating the relation of pixels 33

Figure 2.10: Zigzag, Raster and Snake coefficient reordering techniques respectively. 34

Figure 2.11: Pseudo-code for encoding the MTF transform. 36

Figure 2.12: Pseudo-code for decoding the MTF transform. 37

Figure 2.13: (a) Randomness in block of image based data (b) Randomness in the block of text base data. 39

Figure 2.14: Histogram of the MTF transformed data 40

Figure 2.15: Process of source reduction in Huffman [7] 42

Figure 2.16: Code assignment process in Huffman [7] 42

Figure 2.17: Arithmetic coding procedure for sample stream a_1 a_2 a_3 a_3 a_4 43

Figure 3.1: Plot of block-size with respect to time to process using BWT. 47

Figure 3.2: Proposed method of KGST based BWCA [18] 48

Figure 3.3: Plot of outputs of two adjacent blocks of data 49

Figure 3.4: Lenna image with respective BWT output matrix of image 49

Figure 3.5 (a) Histogram of BWT output (b) Histogram of a kernel from BWT matrix 50

Figure 3.6: Histogram of Available gray-levels in kernel 51

Figure 3.7: Comparison of histogram generated by traditional MTF scheme and KMTF scheme 53

Figure 3.8: Pseudo-code for the generation of Canonical Huffman codes.....54

Figure 4.1: Flow chart for the compressing phase of proposed Scheme 57

Figure 4.2: Flow chart for the compressing phase of proposed Scheme 58

Figure 4.3 Standard color image Lenna, with three RGB slices.....59

Figure 4.4: R, G and B slices with their respective BWT matrix. 59

Figure 4.5: Red slice of BWT image, with histogram of MTF transformed data. 60

Figure 4.6: Preview of images used for compression with proposed and original scheme. [18] 61

Figure 4.7: Bar diagram of compression ratios achieved by BWCA and two versions of proposed schemes. 65

List of Tables

Table 2.1: Encoding process of MTF 36

Table 2.2: Decoding process of MTF 37

Table 1.1: Encoding of sample string in column 1, with symbol map shown in column 2 52

Table 1.2: Encoding of sample string in column 1, with reduced symbol map shown in column 2 52

Table 4.1: Compression Ratios of Lenna image.....60

Table 4.2: Results of sample images with different kernels, the best compression ratio achieved by kernel is shown in bold. [18].....62

Table 4.3: Comparison of compression ratios of BWCA with two versions of proposed scheme. [18].....63

Table 4.4: Comparison of compression ratios of BWCA with two versions of proposed scheme [18].....64

Chapter 1

Introduction

1.1 Introduction to Lossless Image Compression

In our day to day life large amount of information is shared, stored and processed. Since from the first time when electric signals ware transmitted the data is transmitted using several coding techniques, whereas sometimes those coding techniques ware used to encrypt the information, or sometimes it was coded in order to transmit the whole information with fewer coded elements. Most of us usually don't realize the fact that we access and process digital data very often, from surfing the internet to some medical diagnosis after analyzing images from CAT scan or CT scan.

Lossless image compression is the technique of compressing the information of the image so that all the data of the image is retrievable when decompressed, unlike the most commonly known image compression techniques such as JPEG (Joint Photography Experts Group) in which the quality of the image and the detail of the image is compromised in order to save more space and bandwidth. Lossless image compression is used widely in sensitive data such as the medical imaging and astronomical imaging.

The detailed information of how the digital image is compressed is coming in detail later, but for short introduction, the image compression is a sub-field of Digital Image Processing, in which the digital image is represented with reduced amounts of data. The basic principle of the reduction process is eliminating the redundant data.

In mathematical viewpoint a statistically uncorrelated data is obtained from transformation of 2-D image's pixel array. The Image data (pixel array) is transformed prior to storing or transmission, in case of transmission the receiver decompress the image by

using inverse transformation of the applied method and convert the compressed data into original image (*lossless image compression*) or the approximation of that original image (*lossy image compression*).

In many organizations related to the field of astronomy, where most of the information comes in the form of digital images, it is not feasible to store all the information as is. Similar dilemma occurs in the databases of medical record keeping, where records of the patients with their respective scans are stored in database such that to investigate and research some particular problems and to determine the new techniques and diagnosis to coupe up with those problems.

The technique which we improved is based on the research of Burrows and Wheeler in 1994 [1] although this technique was designed for text compression, but many variants of this technique can be found which ware then modified for image compression and the scheme which J. Abel proposed was known as BWCA [2].

Fenwick [3] proposed improvements to increase efficiency of Burrows-Wheeler Transform (BWT) based compression and to reduce the complexity of overall scheme. Michael Schindler proposed a variant of BWT which reduces the time taken by BWT by using limited sorting method in [4], where as Sadakane introduced an idea of improving the speed of BWT in [5]. Many improvements on Post-BWT stages are proposed by J. Abel in [2], research by Balkenol, Kurtz and Shtarkov explains the detail analysis of each stage of BWT based compression techniques in [6]. Before we discuss the technique which we proposed and research related to it, fundamentals of the Image compression are discussed very briefly.

1.2 Digital Image and Digital Image Processing

An image can be described as 2-Dimentional function, so it can be elaborated with as, *f* (x, y) where x and y are the spatial coordinates. The amplitude value of *f* at any coordinates x and y is called intensity of the image, which is also referred usually as gray-level of the image. The image having finite, discrete values of intensity and coordinates is called as digital image.

The digital computers can process the digital images; this processing is referred as digital image processing. As described earlier that digital image comprises of finite elements, having particular location and values. These values on the particular locations are referred to as pixels or image elements.

In comparison between the different types of senses, vision is most dominant and advanced. Vision plays vital role in perception for humans, our eyes constantly capturing the images and our brain is interpreting those images into the recognizable and informative details, but humans are only limited to perceive only in only visual band of the Electro-Magnet Spectrum, But machines can operate in various bands of EM Spectrum ranging from very low frequency gamma rays to very high frequency radio waves. The machines can also produce very detailed images of very minute objects using electron microscopy to radio wave images of enormous objects of astronomy such as galaxies.

Digital image processing can be said as the sub-field of digital signal processing because the image itself is 2-D digital signal. There are various fields of image processing such as *image enhancement*, *image restoration* and *image segmentation*, but we will not concentrate on these fields of image processing, it is worth elaborating that throughout this report and research we will be discussing about the field of *image compression*.

1.2.1 Image formation model

Pixels of the digital image contains the gray-levels of the light intensity, when it is gray-scale image then the gray levels describe the illumination of that particular amplitude value of the pixel located at spatial dimensions x and y. Like human eye camera works similarly on the basis of capturing light on receptors, while the amount of the light acquired by them is then interpreted by brain into signals, the same process is used in acquiring the digital images.

The procedure of acquiring digital images can be described as when light is illuminated to any object the reflection of the light is captured by the camera transducers which convert the amount of light on the sensing area to finite and quantized electrical voltages and the resulting voltage array (from multiple receptors) then form digital image.

In the process of image generation, the values of the image are directly proportional to the energy radiation from the target object; here we are generalizing the luminosity with EM waves, because in digital image processing, many types of images can be processed. As a result the function value of f (x, y) must be finite and non-zero that is

$$0 < f(x, y) < \infty. \qquad \text{(Equation: 1.1)}$$

The values of the function f (x, y) can be characterized by two main components

i. The amount of source luminosity on the object at the time the object is viewed by the camera.
ii. The amount of reflected light from object, which depends upon mainly the reflectance of the object and the color of that object.

The above two components are usually referred to as *Illumination* and *Reflectance* component and can be denoted by i(x, y) and r(x, y) and the combinational result of the two of them is the function value of f(x, y) that is,

$$f(x, y) = i(x, y)r(x, y) \qquad \text{(Equation: 1.2)}$$

Where

$$0 < i(x, y) < \infty \qquad \text{(Equation: 1.3)}$$

And

$$0 < r(x, y) < 1. \qquad \text{(Equation: 1.4)}$$

From equation (xyz) it is indicated that luminance on the surface of the object from the source can vary from zero to infinity, where as the amount of light reflected by the object can vary from zero to one, this reflectance of the object can be referred to as multiplying co-efficient which determines the pixel values, the resultant of both factors is the amplitude value of the function *f*(x, y).

Figure 1.1 illustrates the process of image acquisition, where as the reflectance and luminance of the light are also shown, the digital image formed by the image acquisition process produce the amplitude values for the function which are called as pixels, the pixels describe the gray-levels and can be denoted as,

$$\ell = f(x_0, y_0) \qquad \text{(Equation: 1.5)}$$

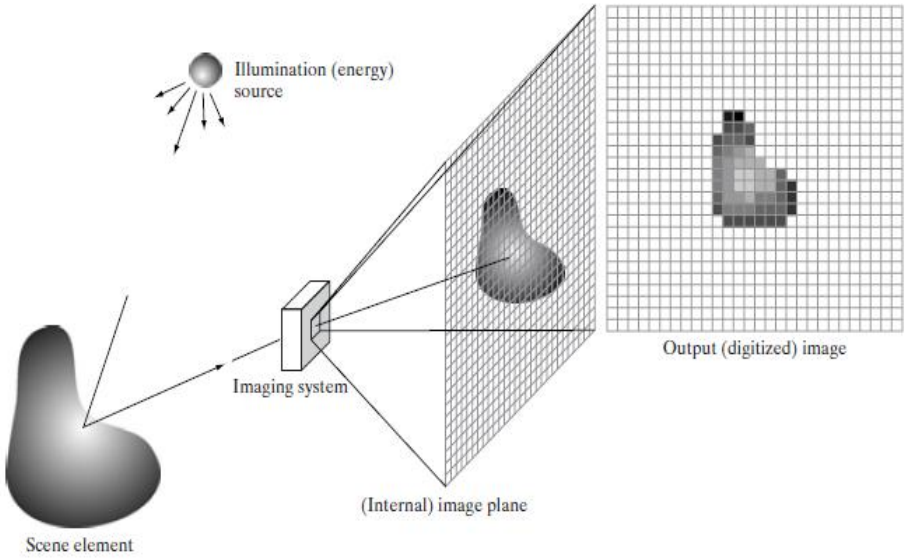

Figure 1.1: Typical Process of image acquisition [7]

The range of the gray-level in digital images depend on the amount of memory elements used to represent the gray-levels, for binary based computers it is usually denoted as 2^n (where n is the number of bits used to represent single element). The value of the *l* lies in the range which is can be represented as,

$$L_{\min} \leq \ell \leq L_{\max} \qquad \text{(Equation: 1.6)}$$

1.2.2 Digitization and Representation of Image

As described in image formation model that gray-levels for the monochrome images (and values of Red, Green and Blue in RGB based color images) the image data lies in the range which is shown by the *equation 1.6*, after all the values of the amplitude of function *f*(x, y) are still continuous and not discrete, so the process of digitization is applied to image. This process is automatically done by the digital image capturing device.

The output of the sensor arrays is usually the voltage waveform which reflects the input of the sensors and is continuous in nature. To create the digital image from the acquired continuous data two processes are done which are sampling and quantization.

The idea if digitization is shown in the Figure 1.2 which shows the continuous image and after that the profile of that image is taken to illustrate the idea of sampling and then quantization is again performed on the same profile of pixels.

The main principle of the sampling is to convert the continuous signal into discrete intervals of time, and take the samples at those intervals, however in digital signal processing point of view the signal (in this case the image) is multiplied with train of impulses and the output of the sampling is discrete in nature, but note that the amplitude of the signal or image is still ranging continuous domain.

To convert the amplitude values of the image the passed through the quantization stage which maps the continuous output of the each sampled value to some finite values, the

process of quantization is not reversible because no information is kept that what changes are made upon the amplitudes.

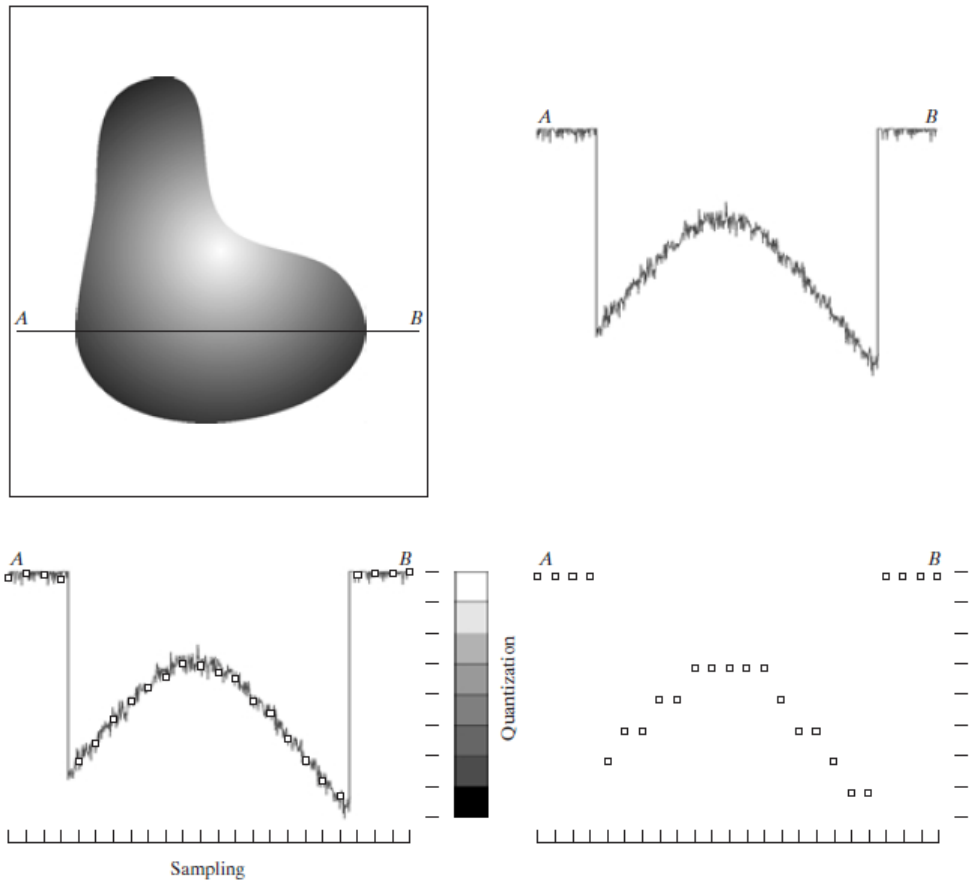

Figure 1.2: Process of sampling and quantization [7]

After successive stages of digitization the image is now finite and discrete values of array which is then stored in 2D matrix order and is digital in nature, hence that digital images can be processed, stored, and shared by digital medium such as internet and mobile phone devices etc.

1.3 Image Compression

Digital Image is 2D signal which is digitized form of analog image, so in general; all the Data Compression methods can be used to compress data in the image, so before we start the actual aspects of digital image compression it would be convenient that some basics of digital images is elaborated.

1.3.1 Image and Data Compression fundamentals

Digital image is a type of digital data which is 2D in nature, as compared to the simple text data which 1D in nature, so the fundamentals of the data compressions are necessary to be discussed briefly before we make the specific type of image compression.

Data Compression refers to the process of reducing the amount of data used to represent the information, noting that information and data are not similar and are not synonymous. Supposing that we have some text information such as "Opening of the hotel XYZ is about to be held tonight at 6pm and the Alice, famous singer is about to present the famous song in the opening ceremony"

From the above sample text the information can be extracted as

- Opening Ceremony of hotel xyz
- Which is about to held at 6pm
- Alice the famous singer is about to present the famous song in the opening ceremony

Now for the data compression consider that the letters in the above text is to be X then if we can re arrange the same information in a way that it can be represented with Y symbols (where X>Y) then we can say that less amount of data is used to describe the information, hence the data compression is achieved. The similar re-arranged text can be "Alice the famous singer performing her hit song on opening ceremony of Hotel XYZ at 6pm"

The amount of data compression achieved is measured by the term compression ratio, which is,

$$C_R = \frac{n_1}{n_2}.$$

(Equation: 1.7)

In above equation the value of n1 and n2 are the values which signify the amount of information carried by two data sets.

And in real life examples the compression ratio of the data depends upon the redundancy of the original data, the more the redundancy the more the compression ratio and the redundancy is the feature which we are targeting to exploit and accomplish the additional data compression, the Redundancy of the data is defined as,

$$R_D = 1 - \frac{1}{C_R}$$

(Equation: 1.7)

Redundancy of the data and the compression ratio both are related terms, because in the data set when there is more redundancy then it means that data can be compressed more so that compression ratio is also increased.

In images the type of redundancies are somewhat greater then simpler redundancies, unlike the data compression the images don't have haphazard entities, and due to the relation of the image pixels with themselves comes with different type redundancies which we will discuss briefly. The main purpose of briefly describing the redundancies is important because the algorithms which we proposed exploits interpixel redundancy and try to increase coding redundancy.

1.3.2 Coding Redundancy

A code can be defined as the system of symbols (for language the symbols are letters, and for digital information symbols are bits) by which the information can represented. Such

as for image the gray-levels of the image stored and represented by the chunks of binary bits called bytes.

The coding redundancy can be described as in any image, the occurrences of gray-levels or individual gray-level is more than others then the amount of data symbols assigned to that high probability gray level can be varied, so that in compressed version of the image the fewer symbols are used. In this report we are not going to describe the mathematical details of the probabilities so we will grasp the idea of coding redundancy with simple example such as in a monochrome digital image of 10x10 consider that gray-levels are in range of 0 – 255 and each gray-level is expressed by the single byte in memory (which is 8-Bits) so average number of representing any gray level is simply 8-Bits.

But variable length coding exploits the probabilities of the gray-levels in order to reduce the average number of bits with respect to the probabilities. Such as in *Huffman Coding* the lowest symbol code is assigned to the gray-level having maximum probability. So simply by varying the amount of data to represent the pixels the overall data can be compressed. Huffman coding technique is a type of *Entropy Coding* technique which exploits coding redundancy and is used in various applications such as in JPEG image compression.

1.3.3 Interpixel Redundancy

Unlike data compression in image compression, coding redundancy is not the only type of redundancy which can occur, as we know that images are 2-D images so that in many images the group of gray-levels can occur in between the image in many locations. That type of redundancy is known as Interpixel redundancy which indicate that some sequence of pixels are behaving as a group and that group can be found repeatedly.

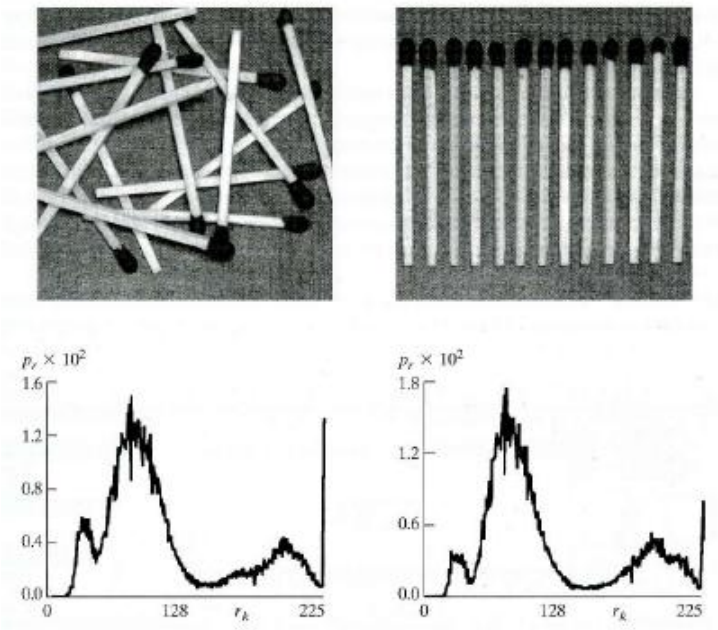

Figure 1.3: Two images with almost similar histograms [7]

The entropy coding technique such as the Huffman coding does not take advantage of such redundancy, because it works on the probabilities and the symbols assigned to them.

In Figure 1.3 two different images are shown in which the histograms shows that the probabilities of the gray levels as shown in the histogram are almost identical, so the entropy coding in the both images would also be almost same. But the compression ratio of the 2nd image can be increased if we can somehow exploit the interpixel redundancy.

There are several data compression techniques which take account of interpixel redundancy, such as *lossless predictive coding* and *Lempel-Ziv* [8]

1.3.4 Psychovisual Redundancy

Psychovisual Redundancy is the type of redundancy which occurs due to the psychological aspects of humans and their perception of vision. The human vision is one of most dominant among senses but human eye can distinguish between different shades of colors on the basis of difference of the light reflected by any object, So most of the times very tiny change in the color is not distinguishable by the human eye. Such as let's take the example of monochrome image with 0-255 gray-levels, the edges in the images are perceived

by human eye due to sudden change of the gray-levels but the surface where shades of the image are changed very smoothly then we cannot distinguish between the amplitude value hold by the neighboring pixels.

Figure 1.4: Two distinctly quantized versions of vase in which left one is quantized

But the image compression techniques which exploit this redundancy are mostly lossy image compression, such as JPEG. In figure 1.4 two versions of images are shown in which one is quantized (Lossy based compressed), but human eye cannot distinguish in between them easily.

1.4 Summary

In this chapter brief introduction regarding fundamentals of image compression are described; the idea of lossless image compression is also mentioned briefly, the basic idea of image data and other types of data are also briefly expressed. The fundamental ideas of compression are discussed and on what factors the compression ratios increase ware also illustrated.

Chapter 2

Literature Review

2.1 The quest for higher compression

There are many techniques available in the field of data compression, in the evolution of the data compression algorithms, these techniques are being modified, and improved, and even novel methods has been proposed and tested by the researchers over the time. In data compression lossy techniques are not are not the favorable methods, but unlike data the image has psycho visual redundancy which we humans don't usually consider and hence it can be exploited, in this research we concentrated on improvements of the lossless image compression, because it is notable that the techniques which we modified and improved can be used on lossless image compression and as well as lossless data compression.

As we have discussed earlier that redundancy is the factor which is very much related to compression ratio of the any type of compression, till this point we have discussed briefly regarding how the redundancies has been exploited with various data compression schemes. But there are some ways by which the redundancies in the data (or image in our case) can be improved, how these all schemes are done is out of the scope of this research phase but we will investigate related techniques which are used to enhance the redundancy of the data.

During the years of development of the data compression algorithms the researchers are trying to find the technique by which the streams of the data can be sorted in any order, because data streams are random sequence of the events which can occur in any order but in sorted data the randomness of any event can easily be deduced. Suppose that in sorted array of data current entity of the data can be represented by value "N" then the value of the upcoming event can be easily deducted that the next event can be equal to or greater then "N" but will never be less, hence the data in sorted sequence can easily be converted into some global format which can be differential or any other statistical form, so the data stream

which is sorted and then passed through that global transformation is most likely to be compressed by the simple entropy coding techniques.

The studies of Data-Structures shows that the process of sorting is not complicated, there are many algorithms which sorts the data, but the problem in finding the sorting transform is that the data which is being sorted must also be able to again de-sort, so that the original information can be retrieved. For the data to be able to de-sort the same amount of the information must have to be kept. In data structure algorithms the information which is used to de-sort the sorted data is known as original indexes. Regrettably the amount of extra information required to store those original indexes is equal to the number of element in the data stream, so for example to sort and de-sort the data of 1000 alphabets, 1000 indexes are necessary to be stored, which is not quiet the solution, but is the problem on its own.

Through time and time for researchers, the idea of finding the Sorting transform was like the finding the Holy Grail. Still that day many attempts has been made to find out the optimal sorting transform which does not require any excess information to de-sort the data.

2.2 Burrows Wheeler Transform

M. Burrows and D. J. Wheeler come up with a new idea of lossless data compression, and in their publication of that data compression algorithm in 1994 [1] based on this algorithm J. Abel named this algorithm as Burrows Wheeler Compression Algorithm (BWCA), block diagram of BWCA is shown in Figure 2.1. The BWCA is based on the reversible sorting transform which was based on the unpublished work which was discovered by D. J. Wheeler in 1984, and the transform was named after the inventors and was known as Burrows Wheeler Transform (BWT).

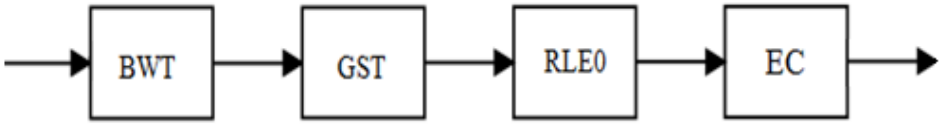

Figure 2.1: Typical BWCA technique as proposed in [2]

New algorithms ware then developed and ware researched, Julian Seward developed a version of data compressor based on BWT and named was named as BZIP2 [9], The traditional data compressor which compresses the files into ZIP format lags behind when compared to BZIP2 because the ZIP compressors uses DEFLATE algorithm [21] which are based on the use of LZ77 [10] and Huffman encoder [11].

Burrows-Wheeler transform is type of block sorting transform which transform the data stream in reversible format which is roughly the sorted version of the data stream, the data which is transformed by the BWT is not equivalent to the sorted version of the data, but the rich factor of the transformed data is that it is totally reversible. The only information which is required to inverse the transformed data is one index which indicates that the first element of the data is available at the following location in the transformed data.

BWT is not a mathematical transform such as Fourier Transform or Discrete Cosine Transform; instead BWT is lexographical transform which maps the data on the basis of the basic sorting e.g. alphabetical sorting.

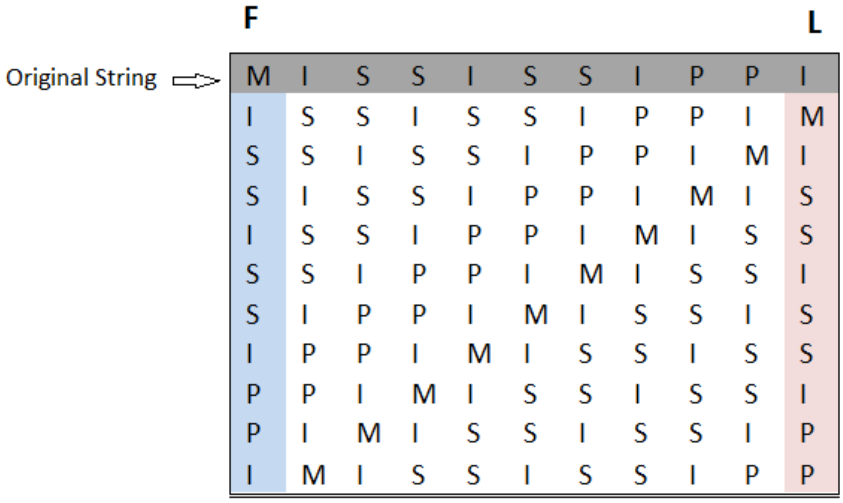

Figure 2.2: N x N matrix M of the all circularly shifted data

The steps involved in the BWT are discussed and illustrated briefly with transforming the sample string data “MISSISSIPPI”. As BWT is block sorting transform, the data in the block is sorted, the size of block for BWT can contain large amount of data, whereas the amount of block-size is related to the overall compression ratio of the whole BWCA. For the

above example let's consider that we are sorting one block of data which contains string shown in figure 2.1.

The first step in BWT of bloc-size N (in our example N=11) to generate N-1 Strings of data block, whereas each individual string from N Strings is the circularly shifted array of the original data, and as it is circularly rotated so each of the N Strings will contain all the available elements as of original block. The Matrix let's say it as M which is of N x N data elements and can be formed from all the N Strings of data which ware generated and is shown in figure 2.2.

The next step in BWT is to sort the strings in the matrix M, the sorting process in this step is not simple string transform, but is sorting of whole row on alphabetical order, in C,C++ and MATLAB this type of sorting can be done by the "sortrows".

F										L
I	M	I	S	S	I	S	S	I	P	P
I	P	P	I	M	I	S	S	I	S	S
I	S	S	I	P	P	I	M	I	S	S
I	S	S	I	S	S	I	P	P	I	M
M	I	S	S	I	S	S	I	P	P	I
P	I	M	I	S	S	I	S	S	I	P
P	P	I	M	I	S	S	I	S	S	I
S	I	P	P	I	M	I	S	S	I	S
S	I	S	S	I	P	P	I	M	I	S
S	S	I	P	P	I	M	I	S	S	I
S	S	I	S	S	I	P	P	I	M	I

Original String ⇨

Figure 2.3: Sorted matrix M'

Figure 2.3 shows the sorted matrix M', the elements in the first column of the sorted matrix M' can be denoted as "F" and the last column as "L". Notable point in the process of

sorting is that the index of the row which contains the original data block must be kept for the de-sorting of the data block; the row containing the original string is highlighted.

The first and last column of the matrix M' plays vital role due to their relation with the original data, the relation in the F and L column with original data block elements is that all the elements in original string are available in the F and L columns, we can analyze that F String which contains "IIIIMPPSSSS" is the sorted version of "MISSISSIPPI", where as L String contains "PSSMIPISSII".

The Last column of the M' is transmitted (or stored in case of data storage) the index value is also sent (or stored), so we can say that the last column and the index value are the only outputs of the BWT. Remarkably the data in the last column is swarming with the consecutive entities, which increase the compression. As described earlier the larger the block the more replicated data entities are found in the BWT output, hence BWT transforms the data which is highly acquiescent for data compression.

2.3 Inverse Burrows Wheeler Transform

The index from BWT output refers to the individual byte in the last column, because as it was described earlier that in the decoding phase the only data available is the last column data. The *Inverse Burrows Wheeler Transform* uses the value of the index to determine the transforming vector, which is the key for the transformation process of BWT.

The transformation process of BWT straightforward for the individual strings, but by analyzing only the output of BWT it seems that it is unattainable for un-sorting the data to its original form, but this analogy is not true, there is certainly a way to un-sort the BWT output of the original data and the key of that inverse lies in the index of the original string.

The index from the output data refers to the individual entity in the last column string, because as it is described earlier that only data available to the decoder is the last string and the index. The most important data in the BWT output data is undoubtedly the index, because the mapping of the last string data to the original data is only achievable by the value of the

index. The index helps the decoding process by jumpstarting the recovery of the correct original string.

The inverse of the burrows wheeler transform is done by finding the transformation vector for the data, the key to finding the transformation vector lies only in the contents of the first column and last column of data. Amazingly if we have the copy of the L then we already have the copy of F.

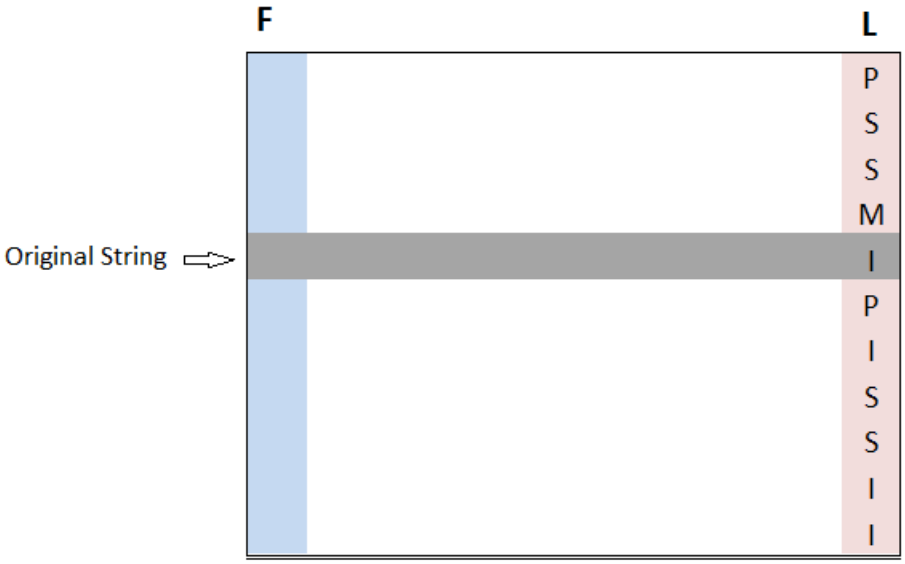

Figure 2.4: Only known data for the re-creation of matrix M

With the copy of L string, the copy of F string can be generated very easily, as we described earlier we mentioned a fact that F and L are related to original data in a way that both F and L contains all available entities presented in the original data, and the F string is just the sorted version of the original data. We use this relation in consideration and by simply sorting process of L string we can conclude the F string.

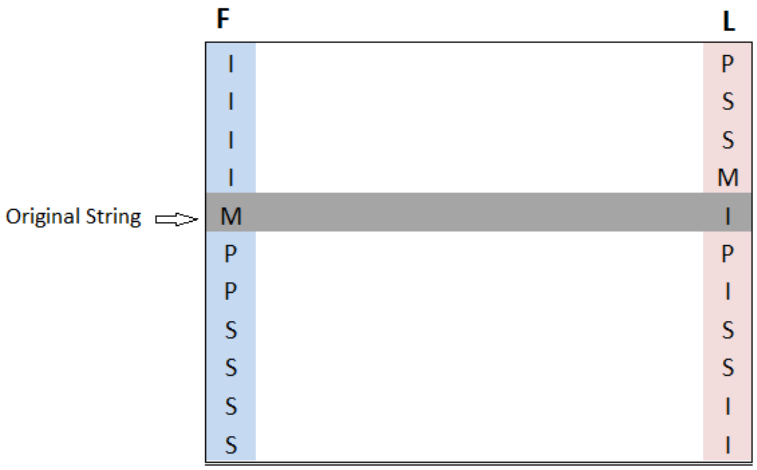

Figure 2.5: State of Matrix M after recovering the F string

Figure 2.5 shows the sorted version of the L string whereas the middle elements of the matrix M are still unknown. As the matter of fact we don't really need all the intermediate values of between F and L, so these values are shaded because these are not used anywhere in the process.

The transformation vector which is used to retrieve the original string is not the last part of the decoding process; the transformation vector is the index values of the L string which describe the target location of the entity in F string.

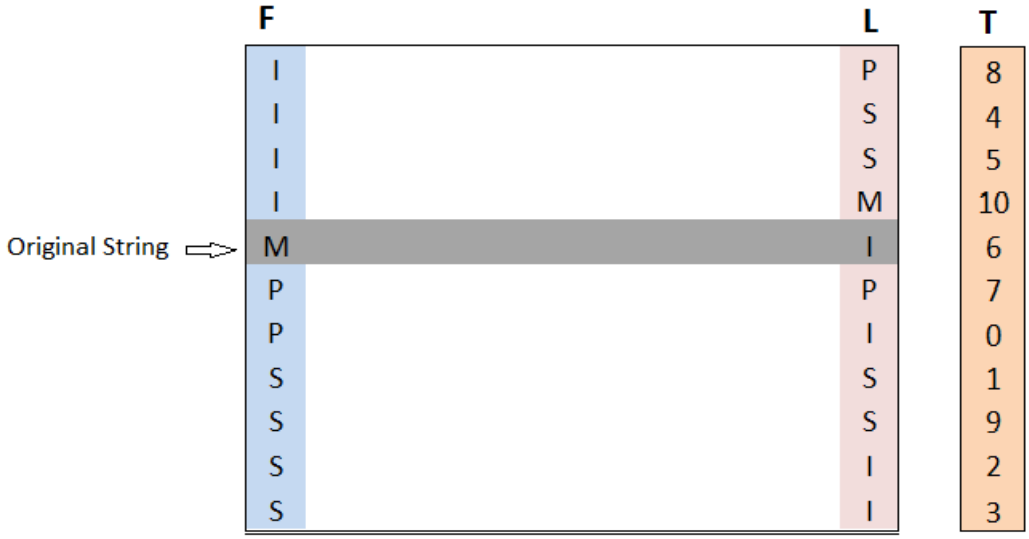

Figure 2.6: Matrix M with the transformation vector

It may seem at first that there are multiple entities of the data with same values so how we are going to know for sure that what "I" belongs to which column? The answer to this question is very simple that in traversing the L string for sorting the first "I" character is assigned to the first location in the F string. This solution does not change anything because the matrix M (in original data) was also sorted so the positions of the entities will not mix-up.

After calculating the values of the transformation vector "T" the original order of the data of L string is determined by the pseudo-code shown in Figure 2.7. Whereas the index

```
void decode()
{
    int index = primary_index;
    for ( int i = 0 ; i <7 ; i++ ) {
        index = T[ index ];
    }
}
```

Figure 2.7: Pseudo-code for obtaining the original index for transforming the data into original format.

The values from index array from the pseudo-code shown in Figure 2.7 will contain the original positions of each entity for the data and the data is finally retrieved without any loss. Figure 2.8 shows the process of retrieving original string from index values.

Index	L	T	Steps	String
4	P	0	1	I
0	P	5	2	PI
5	P	6	3	PPI
6	I	1	4	IPPI
1	S	7	5	SIPPI
7	S	9	6	SSIPPI
9	I	2	7	ISSIPPI
2	S	8	8	SISSIPPI
8	S	10	9	SSISSIPPI
10	I	3	10	ISSISSIPPI
3	M	4	11	MISSISSIPPI

Figure 2.8: Final step in order to retrieve original data

2.4 BWT for Images

BWCA was implemented to work with the compression of text based data, and due to the lexographical sorting techniques it seems that BWT is only suitable for the compression of text, but this is not the case in real world scenario. Several improvements ware proposed to utilize the supremacy of the BWCA for images, there are many variants of the BWCA which are modified to work with the images. There are some advantages of using the BWCA with the image data, such as when working with block of pixels the pixels are closely related to one another than the text data or any other data, such as in smooth images the information in any pixel is having strong relation with adjacent pixels. This relevancy between the pixels can be the potential to increase the compression ratios for lossless image compression.

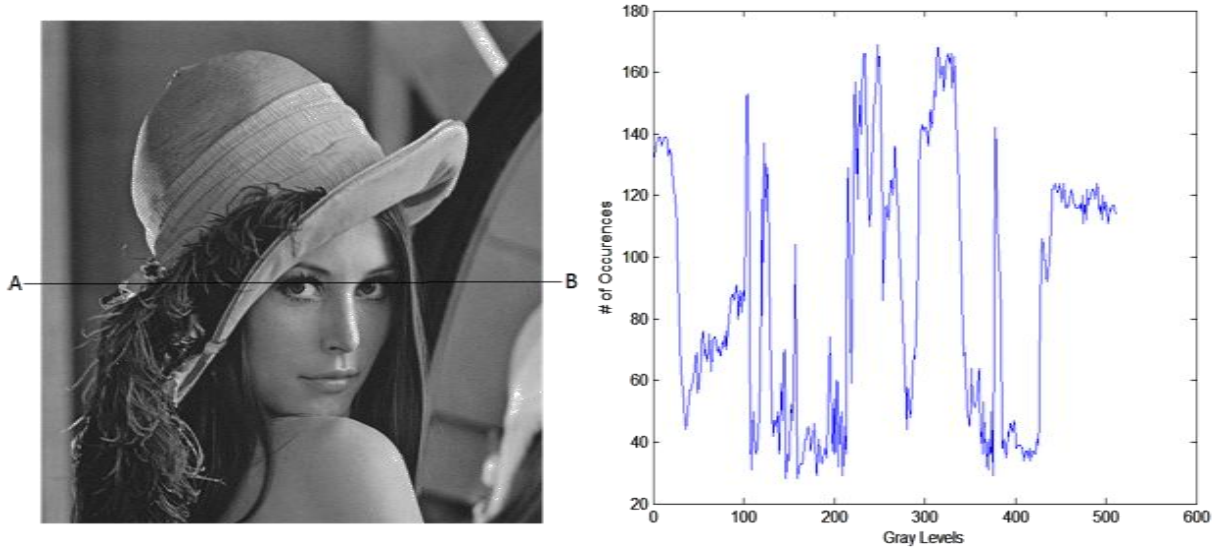

Figure 2.9: (a) Sample image "Lenna" (b) The profile of image from point A to B illustrating the relation of pixels

Figure 2.9 shows the sample image and the profile of image from point A to B which shows that on the normal most of the profile the pixels are smooth and the distortion between the pixels only occurs when there are edges.

Despite the advantages of image data, however there are some complications also exists, the basic complication with the image data is that digital image is inherently 2D matrix of quantities whereas the BWT works on 1D data, so the data is first transformed from

2D to 1D in order to make it compatible to be used by BWT. There are several techniques which are used to perform the task of transformation of the image data into 1D, and these techniques are known as *Coefficient Reordering*.

The typical scheme for the BWCA is slightly modified and the step of coefficient reordering is introduced before selecting the block of data from image. There are many techniques for coefficient reordering which are usually referred as path scanning techniques, because in transformation of the 2D data the elements of the 2D matrix are traversed and are translated to 1D, the path which is used to traverse can be determine by some mathematical formation, or it can be general. Figure 2.10 shows some commonly used path scanning techniques.

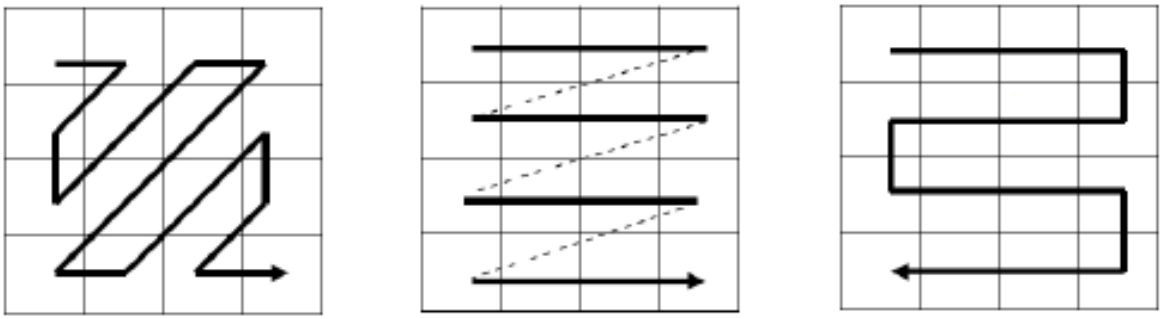

Figure 2.10: Zigzag, Raster and Snake coefficient reordering techniques respectively.

Although the images are transformed into simple stream of data but this transformation comes with a cost, and the cost of this is that if let's consider that raster scan is used for path scanning then the output data stream will only contain the horizontal relation of the pixels, whereas the vertical relation of the pixels is totally compromised. Zigzag scan tends to solve this problem by taking elements in zigzag order so that the output data must contain both relations, but even zigzag technique does not totally solve this problem, due to different types of images variants of path scanning techniques produce variable results.

2.5 Global Structure Transform

The BWT was first developed to compress text based data, and the main characteristic of the BWT is that in the process of transforming the data is sorted lexographically; hence the output of the BWT contains consequent occurrences of similar entities, such as in the example given in section [1.3]. The runs of the similar entities may sometime be haphazard such as runs of character "Z" can come next to character "A" and we already know that the occurrence of the A and Z are on the two opposite ends of the alphabets. So to overcome this transitional behavior M. Burrows and D. J. Wheeler suggested using the Move-to-Front (MTF) in their publication [1]. This transformation scheme was proposed by Bentley, Sleator, Tarjan and Wei (1984) in [12], and was discovered by Peter Elias in [1987]. The brief detail of how the MTF transform works is coming next, but for overview the MTF is the transform, which translates the text based local structure of data (from alphabets) to global context (simple numbers) and hence these type of transforms are known as Global Structure Transform (GST).

The MTF schemes simply works in a way that it translates the input stream, with all entities in string, in this report that table is referred as entity string (in the example of English language all the alphabets) of data in a way that for the transformation of character "X" the index value of that character is kept and the entity of that character is then "moved to front" which is starting of the entity string, at the time of the processing of any entity the index of that particular entity (from entity string) is positioned at 0^{th} location in entity string. Thus the size of the output symbols is equal to the length of the original input stream. This simple yet elegant scheme plays vital role in generating lower index values for the frequent entities.

Figure 2.11 shows the pseudo-code for the encoding process of MTF transformation process. As described earlier that MTF encoder and decoder both shares the entity string for the processing, and hence for text data compression the entity string can be simply generated, the same process of generating the entity table can also be used for any type of data. E.g. for image based data the all possible values for the pixels can be generated similarly.

```
initialize the symbol table;

while ( not end-of-file ) {

   K = get character;

   output K's position(P) in
   the symbol table;

   move K to front of the
   symbol table.

}
```

Figure 2.11: Pseudo-code for encoding the MTF transform.

2.5.1 MTF Encoding

The working of the MTF can be comprehend by taking the example of encoding the example string "abcccbaaa" the encoding process is illustrated in table 2.1, whereas first column shows the symbol being processed, second column shows the code generated (index values) and the third column shows the entities string.

MTF Encoding		
Symbol	Code	List
a	0	a b c d . . . z
b	1	b a c d . . . z
c	2	c b a d . . . z
c	0	c b a d . . . z
c	0	c b a d . . . z
b	1	b c a d . . . z
a	2	a b c d . . . z
a	0	a b c d . . . z
a	0	a b c d . . . z

Table 2.1: Encoding process of MTF

From the output of the MTF encoding we can analyze that output string consists of subsequent runs of zeros, in contrast with the *Run-Length Encoding* in which the runs of the entities are encoded directly.

2.5.2 MTF Decoding

The decoding phase of the MTF algorithm is as uncomplicated as encoding, the output data from the MTF algorithm is processed in similar fashion, in MTF decoding the entity string is also generated and the same "Move to Front" course of action is performed on the data with respect to the entity string. Hence the translation of the data-source is perfectly reconstructed. The pseudo-code of the MTF decoding is shown in Figure 2.12.

```
Initialize the symbol table;
While ( not end-of-file ) {
   P = get position;
   output the symbol(K) in position P;
   move K to front of the symbol table.
}
```

Figure 2.12: Pseudo-code for decoding the MTF transform.

The translation process for the MTF scheme for the string data "012001200" is shown in table 2.2 in which first column is code which was the output of the MTF encoder, second column shows the symbols which are decoded from the entity string shown in third column

MTF Decoding		
Code	Symbol	List
0	a	a b c d . . . z
1	b	b a c d . . . z
2	c	c b a d . . . z
0	c	c b a d . . . z
0	c	c b a d . . . z
1	b	b c a d . . . z
2	a	a b c d . . . z
0	a	a b c d . . . z
0	a	a b c d . . . z

Table 2.2: Decoding process of MTF

2.5.3 Higher Order Processing with MTF

By analyzing the output of the MTF transformed data it may seem that stream of zeros is the only advantage of but it is not the whole scenario, the transformed data contains an significant quality which can be describe as that MTF dose not only maps the runs of distinct data to run single entity of data, but it also maps the sequence of data (with all the characteristics) into a stream of data which also contains the similar pattern of the index codes, to illustrate this procedure consider the stream of data "ABABABCCCCCCCDEDEDEFFFFFFF".

MTF encoder when analyze this streams of data it will eventually start assigning the initial codes, consider that entity string contains all the available symbols used to represent the text based data. At the starting of encoding procedure the first character A and B are assigned 0 and 1 respectively, because the character index in the entity string for A is 0 and B is 1, but the things get interesting afterwards, when the next pair of AB comes in the sequence the encoder will move the character A in front of entity string and will generate the output 1, same output will continue to output from encoder until all the pairs of AB are processed, at the occurrence of first C character the encoder will output 2 and the stream of zeros will be outputted until the first occurrence of D, the process of generating consequent 1s will continue with the pair of DE so the output of the MTF stream will be "011111200000034111115000000".

The interesting factor in the output of the sample stream of data is that it just don't only generates the stream of zeros, but when it comes to encode repeated pattern in the input data then MTF generates a pattern of data and not just increasing the wasting the storage which is the problem of Run-Length Encoding.

There can be many more patterns appearing in MTF encoded data which can be observe and used to enhance the overall compression of the data, there are many second-stage compression algorithms which can be designed to process such output patterns of the MTF those second-stage algorithms can even enhance the output of the MTF before processing it through the entropy coders.

From the above illustration of the patterns in the MTF output it is obvious that MTF transform is more dominant then Run-Length Transform (RLT), since in RLT only the runs of the data are packed, where as the patterns of runs of data are simply ignored.

The RLT is faster encoding technique then MTF but is not able to perform proper exploitation of patterns, in image type data compression the patterns in the pixels is much more then of text based data, so MTF is observed to perform two time better compression ratios than with RLT.

During the testing stage it was observed that output of MTF and RLT radically improves the compression rations for the variety of algorithms such as *Huffman*, *Arithmetic Coding* and *LZW*. The above usage of MTF and RLT was analyzed only when the data is highly redundant, and this type of redundancy is not always found in the patterns of text and binary files which we mostly use. In conclusion it is safe to state that MTF and RLT are incredibly successful with image compression.

Figure 2.13 illustrates the type of data randomness present in the various types of data, as it is obvious from the comparison of the both type of data that randomness in image data is less, and hence the MTF enhance the overall compression performance for the images.

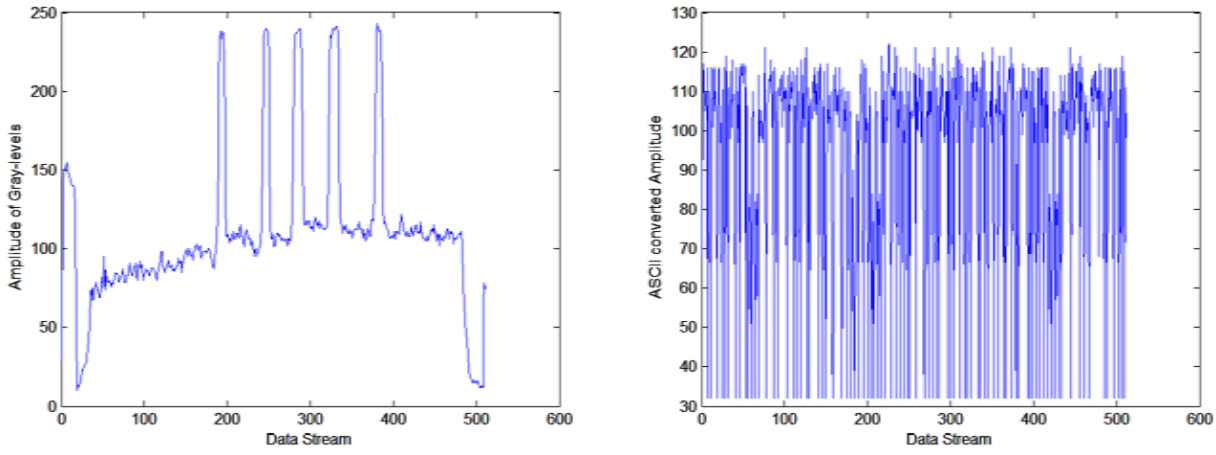

Figure 2.13: (a) Randomness in block of image based data (b) Randomness in the block of text base data.

2.6 Zero Run-Length Encoding (RLE-0)

Typical BWCA scheme consists of four basic steps for data compression in which after GST stage (which is MTF as suggested by M. Burrows and D. J. Wheeler) the *Zero Run-Length encoder* (RLE-0) is also suggested, from all the discussion of the MTF and RLT it seems at first that RLT is the extra step which is not necessary, but this is not the case. MTF is just the encoder of data and the MTF output data is not yet compressed, by analyzing the output of the MTF it is clearly visible that occurrences of zeros are is very frequent, even many times the stream of zeros are present, as shown in Figure 2.14 in which the histogram of the output of the MTF clearly illustrates the high occurrences of zeros.

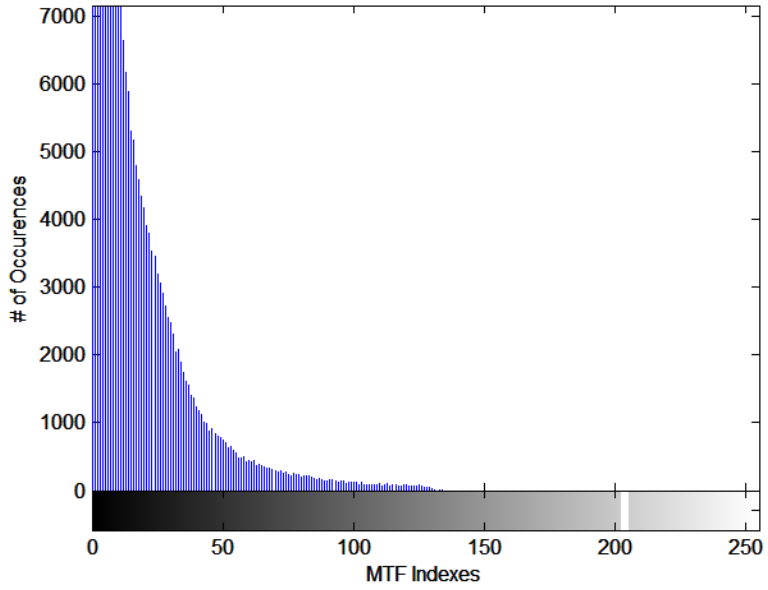

Figure 2.14: Histogram of the MTF transformed data

It is always an option to directly process the MTF output with entropy coders, but the problem with this procedure is that not only the present zeros are in high probabilities but most of the time these zeros are in stream, so one special Run-Length Encoder can be used to exploit this redundancy. Hence the RLE-0 is the run-length encoder which only transforms the zeros so that streams of the zeros can be reduced and the data can be compressed more.

2.7 Entropy Coder

Entropy coder is the final stage of BWCA; this is the stage in which the data is compressed in order to reduce in size to represent the information, as we have described all the stages of BWCA, RLE-0 is the stage in which the stream of 0s are removed, but the zeros are not the only type of redundancy in data, so an actual data compression algorithm is needed to reduce the whole data.

Entropy coders are used to eliminate the coding redundancy of data, and as described earlier coding redundancy is usually present in data, even in the binary codes which are used to represent the values of gray levels. To eliminate coding redundancy a *variable length code* can be used which can represent the data instead of original data.

2.7.1 Huffman coding

When it comes to eliminate coding redundancy then Huffman coding is the most popular and widely used technique, this technique was proposed by Huffman in 1952 [11] and still is used due to the significant amount data reduction. Huffman coding works on the individual symbols of information source and yields the smallest possible code for every symbol; the codes assigned by Huffman coding are variable length codes, as the optimal fixed codes are generated for individual source symbol then the original data can be coded at a time by simple look up table approach.

The brief intro of Huffman encoding is presented here to describe the importance and working of entropy coder stage, the first step in Huffman coding is to generate the list of available symbols with respect to their probabilities, when the list of probabilities are available then the procedure of source reduction is performed, in this process the two lowest probabilities of the information source are added together and the same process is repeated attain until only two sources are remained.

The process of the source reduction is illustrated in Figure 2.15 in which the available symbols are represented in column named as symbols and the probabilities of their respective occurrences are shown next.

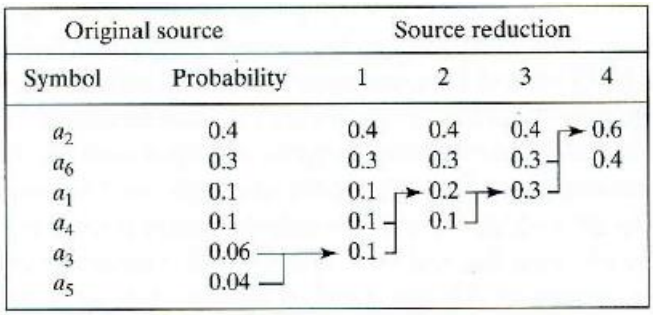

Figure 2.15: Process of source reduction in Huffman [7]

The second step of Huffman coding is to assign short codes to reduced sources, in this step the smallest codes are assigned to the final sources, for the rest of sources the appending of 0s and 1s is used, and the assignment procedure is performed on all the sources and hence all the sources have their own unique codes as shown in Figure 2.16.

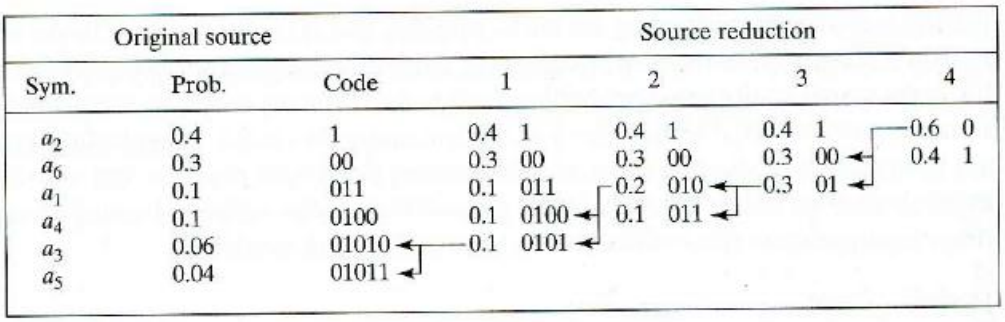

Figure 2.16: Code assignment process in Huffman [7]

Considering that for the above symbols the minimum amount of data bits required to represent all the symbols is 3-bits (since total symbols are 8, i.e. $a_0,a_1,\ldots a_7$ so $2^3=8$) and after Huffman coding the average code length can be calculated by as follows.

$$L_{avg} = (0.41)\,(1) + (0.3)\,(2) + (0.1)\,(3) + (0.1)\,(4) + (0.06)\,(5) + (0.04)\,(5)$$

$$L_{avg} = 2.2 \text{ bits/symbols}$$

Hence the average amount of information with the codes generated by Huffman is lower than original 3 bits/symbols.

2.7.2 Arithmetic Coding

Variable Length Coding is not the only way to reduce coding redundancy, Arithmetic Coding works on the principle of generating nonblock codes, in arithmetic coding the one to one association does not exists, as an alternative in arithmetic coding sequence of symbols are translated into single arithmetic code.

Arithmetic coding also uses the prior information about the occurrences of symbols and with this information the optimal arithmetic code for sequence of symbols can be generated. The main principle of arithmetic coding procedure is illustrated in Figure 2.17 in which the sequence of symbols a_1 a_2 a_3 a_3 a_4 is described. Considering the only available symbols in data are a_1, a_2, a_3, a_3, and a_4 and their respective probabilities is also shown in Figure 2.17

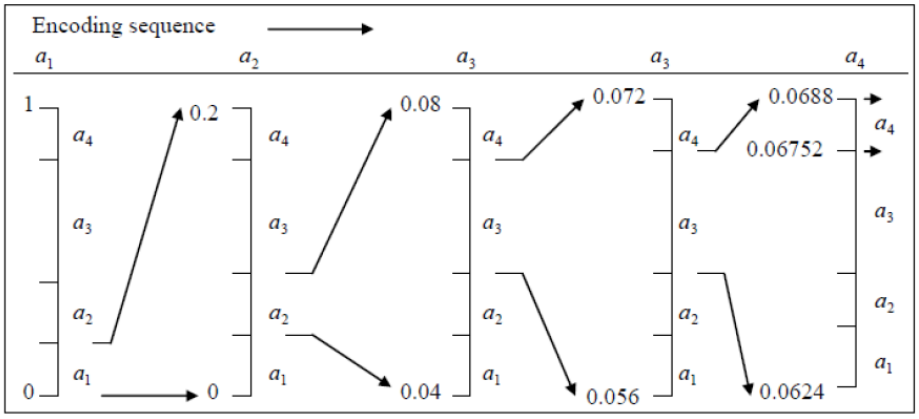

Figure 2.17: Arithmetic coding procedure for sample stream a_1 a_2 a_3 a_3 a_4

Probability of each available symbol in the data provides vital role in coding data using arithmetic coding, the whole probability interval (i.e. 0-1) is divided into sub intervals with respective share of probabilities, on encoding of every symbol the interval is again divided and the process continues until the whole stream of data is coded.

J. Abel in his report [2] proposed the use of Arithmetic coding after using *Incremented frequency count* (IFC) and showed that arithmetic coding provides better results than other entropy coding techniques.

2.8 Improvements and variants of BWT over the time

In the first paper of M. Burrows and D. J. Wheeler they tested and recommended the use of MTF transform as the second-stage processing of BWT data in [1] after the GST stage the entropy coders can take over the data and perform the actual compression. Arithmetic coding performs better in comparison to Huffman coding.

Weighted Frequency Count (WFC) is another type of GST was proposed by Deorowicz in [13] the scheme was proposed to replace MTF at the GST stage on BWT output. The research by Deorowicz showed clearly illustrated that use of WFC with right parameters can perform in similar manner as MTF, i.e. WFC is superset of MTF. He also cited previous two distinct works presented by Balkenhol, Kurtz, and Shtarkov and named as MTF-1 in [6] and other version of MTF by Balkenhol and Shtarkov which is named as MTF-2 in [7]. The working of the MTF-1 is based on the principle that the symbol which is being processed is placed on the 0^{th} index only when the symbol is currently on 1^{st} index, if the symbol lies on higher index then 1^{st} index then the symbol is simply placed at 1^{st} index at first. On other hand MTF-2 the movement of the symbol from 2^{nd} index is only performed when the previous symbol is at 1^{st} index, Both MTF-1 and MTF-2 are superior examples of MTF improvements.

While the MTF-1 and MTF-2 are the second-stage improvements of the BWCA, Chaplin and Tate [15] investigated the improvement of BWT stage itself. Their study of re-ordering the alphabetical characters provides improved compression then usual ordering of characters, the established a special re-ordering in which vowels are placed before consonants, and the sorting of the BWT is done with that particular ordering.

Michael Schindler proposed a new “block sorting” technique in [4] which is derived from BWT which uses the Schindler’s Transform (ST) on very large blocks, the local characteristics of transformed strings are preserved suitably, the program which works on the principles of ST is also available and is known as szip. The compression times of ST based

algorithms is much faster as compared to BWT based techniques, but the compression ratios of BWT is still better. Compression programs such as BWMonstr and Nanozip use the BWT and are at top of benchmark tests.

David Scott has proposed a variant of BWT and named BWTS (Burrows Wheeler Transform Socttified) in [16] which the BWTS does not oblige index of the BWT data for output; on the other hand BWT requires index data. The promising technique of BWTS works on the principle of sharing the load of processing in decoding phase in a way that in BWTS only one permutation will correspond to the recovery of original input string so with this processing and decoding the index is avoidable. It is obvious that BWTS is able to improve the compression without the need of indices, it is reported that by the use of BWTS the *Calgary corpus* is compressed more than best Lempel-Ziv coders.

David Scott has proposed a new version of BWT which does not require index to be stored in [16], while Manfred explains the uses of scottified BWT in [23].

2.9 Summary

In this chapter the original scheme of BWCA is explained in detail, each stage of BWCA with adequate explanation is illustrated with short but related examples. Related works to lossless image compression techniques ware also discussed in brief detail; the other fields of BWT are also discussed momentarily.

Chapter 3

Proposed Scheme

3.1 Overview

In preceding chapters brief introduction of BWCA is described, the main principles and factors which distinguish BWCA with other frequently used algorithms for lossless image compressions such as LZ77 [10], DEFLATE, gzip and rzip are also mentioned briefly. All of the above mentioned techniques use dictionary based compression approach but BWCA works on the principle of block sorting.

As in practical scenarios we can fairly say that all achievement of compression ratio does not come without a cost, so despite the increased compression ratio achieved by BWCA, there are some practical problem faced by BWCA based compression techniques, due to the block sorting structure of BWT the BWT is not scalable for real-time environment because all the data in the block must be present in order to lexographically sort the data. M. Burrows and D. J. Wheeler demonstrated in [1] that compression ratio of BWCA increase with the block-size; the increase in block size inherently comes with two main problems such as Memory utilization and Speed.

3.1.1 Memory Utilization

In section 2.2 the process of BWT is illustrated in detail, and we came to realize that the rotatic permutations of the data are generated and the matrix of all those permutations and the original data is sorted, so in general terms if we want to apply BWT on 1 Kbytes (1 x 1024 bytes) of data, and consider the block size as "*n*" then matrix generated from that block will be equal to *n* x *n* which is equal to 1024 Kbytes. But this case is not the only case used for BWT, in the research presented in [1], [2], [17] to common block size is much larger.

With evolution of memory management techniques it seems at first that processing 1 Mbytes of data (1 Mbytes = 1024 Kbytes) is not problem at all, and this is true indeed, but the main issues in the BWT is large block size, so the problem of higher block size directly relate to high utilization of memory resources.

3.1.2 Speed

The second issue in the large block size of BWT is that the amount of time needed to process the block increase rapidly, the sortrows function of MATLAB and C works on the principle of sorting the 2D matrix based data and with the size issue the complexity and time of the sorting process (which is spirit of BWT) increase.

Many researchers have been taken on improving the sorting time of BWT such as proposed by Sadakane in paper of high throughput BWT [5], implementation of *qsort* (quicksort) is also used instead of using commonly used sortrows, but despite all the efforts the main issue in BWCA scheme is still the sorting of block of data.

Figure 3.1 shows the plot of block size of memory utilization with respect to time it takes to perform BWT on data, as it is vividly elaborated that with the increase of block size the time of BWT increases.

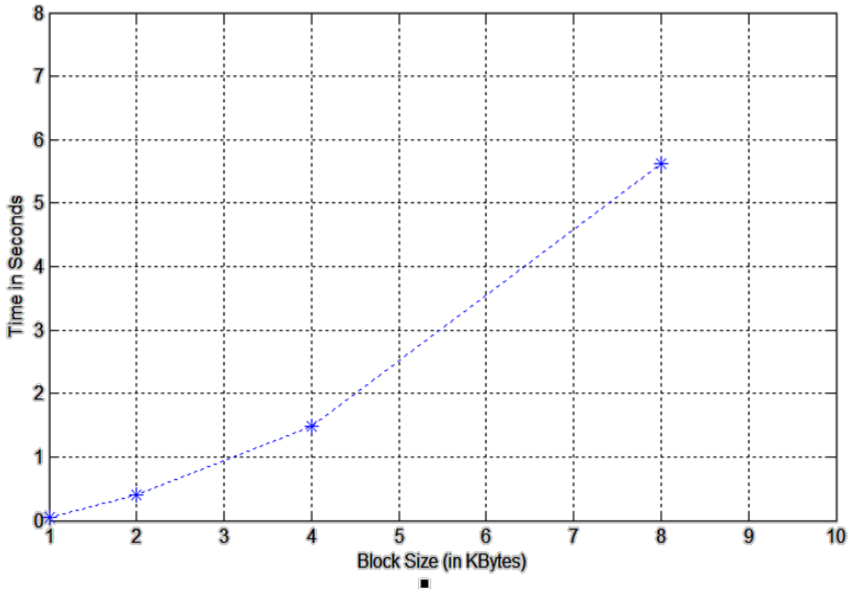

Figure 3.1: Plot of block-size with respect to time to process using BWT.

The proposed model of Kernel based MTF (KMTF) based BWCA is designed to achieve the compression ratios equal to or greater than that of BWCA by using less block size, and this is not the only performance increment achieved by the proposed algorithm.

3.2 KMTF based BWCA

The proposed model is the improved version of the original scheme proposed by M. Burrows and D. J. Wheeler, in which several improvements has been made in order to make the proposed model more suitable for image based data, the redundancies which are discussed in section 1.4 are the main focus of improvements.

The block diagram of proposed scheme for improved BWCA algorithm is shown in Figure 3.2. The proposed scheme maximizes the utilization of all the stages is targeted so that the overall compression of the scheme can perform superior. Our main concern however was to improve the post-BWT stages and the utilization of BWT based data. The improvements on each stage of BWCA are coming in detail in later sections.

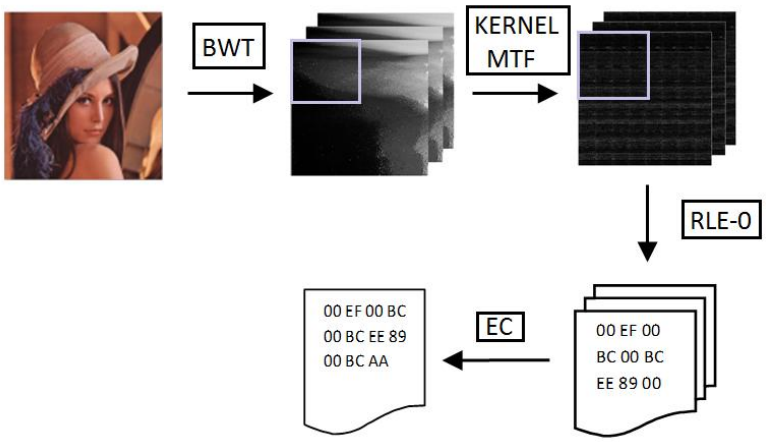

Figure 3.2: Proposed method of KGST based BWCA [18]

3.2.1 Data Characteristics in output of BWT

As described earlier BWT process blocks of data, and hence sorting them individually, in text based data (for which the BWT was originally designed) the binary information is haphazard, but after sorting procedure of BWT the output data contains not only redundancy but the data is sorted in abstract sense.

This sorted behavior of BWT output data plays vital role and in research phase our study of that data revealed that sorted information of image based data is closely related to

output of the nearby blocks. This vital behavior is illustrated in Figure 3.3 in which two adjacent blocks of data are plotted and we can analyze the relevancy between them.

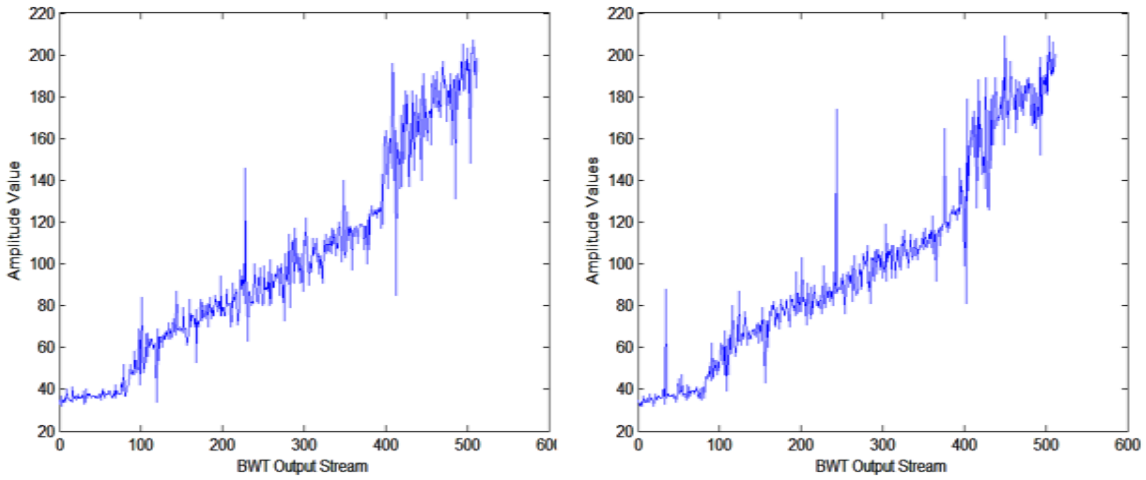

Figure 3.3: Plot of outputs of two adjacent blocks of data

To fully utilize and exploit this relevancy in between the blocks, in proposed scheme we re-arranged the blocks of data in 2D matrix by which the starting and ending of the blocks are placed side by side to each other, with this re-arrangement of BWT data the matrix of BWT output is shown in Figure 3.4, the standard image of "Lenna" is converted into gray and then transformed using BWT.

Figure 3.4: Lenna image with respective BWT output matrix of image

The resolution of standard Lenna image is 512 x 512 and for this illustration we have used the block size of 512 bytes, the BWT matrix and image are scaled down in order to adjust in Figure 3.4.

We can analyze in Figure 3.4 that BWT blocks are arranged in a manner that similar value gray-levels are clustered around and hence building a pattern with neighboring gray-levels.

3.2.2 Kernel MTF Encoding

The rearrangement of BWT data performs vital role in grouping similar type of gray-levels. This important rearrangement of BWT data provided us the freedom of utilizing the group of gray-levels in different regions with respect to their neighboring areas. With this idea, the Kernel based MTF encoding is proposed in the research [18], for the comparison of Kernel based data and simple BWT data (from original BWCA) the Figure 3.5 illustrate histogram of both schemes.

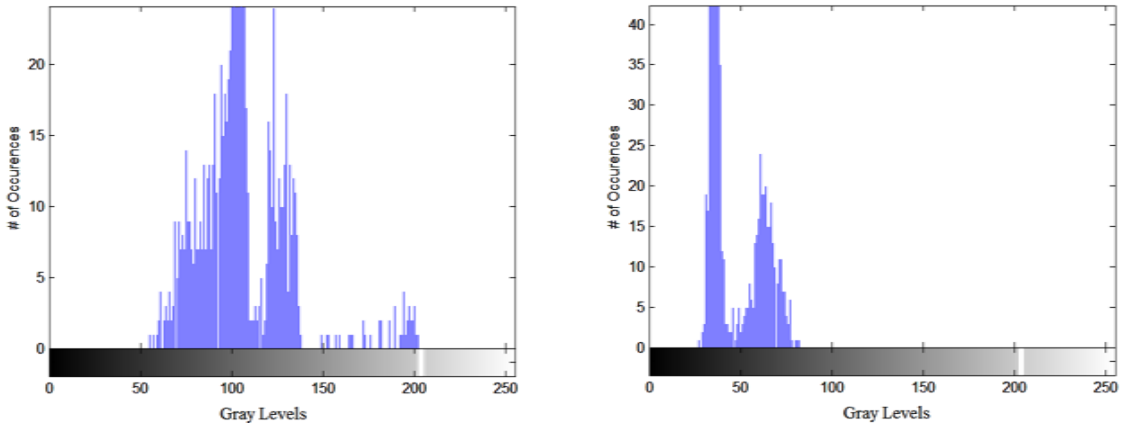

Figure 3.5 (a) Histogram of BWT output (b) Histogram of a kernel from BWT matrix

As described in section 2.5.1 that in MTF encoding is also used to transform 1D data so any coefficient reordering technique can be used which are described in section 2.4, in the proposed scheme we have used Raster scan because it keeps the relations of data in row fashion.

3.2.3 Reduced Encoding Map

MTF encoder translates the input stream of data into the index of the each character present in the symbol map, this process of transformation of data results in output of indexes which are more redundant then original one and contains more patterns in between the data entities which are favorable in data compression.

Figure 3.6 shows the histogram of the available gray-levels in a kernel, which illustrate that not all the gray-levels are available in kernel, so processing MTF with all symbols are redundant.

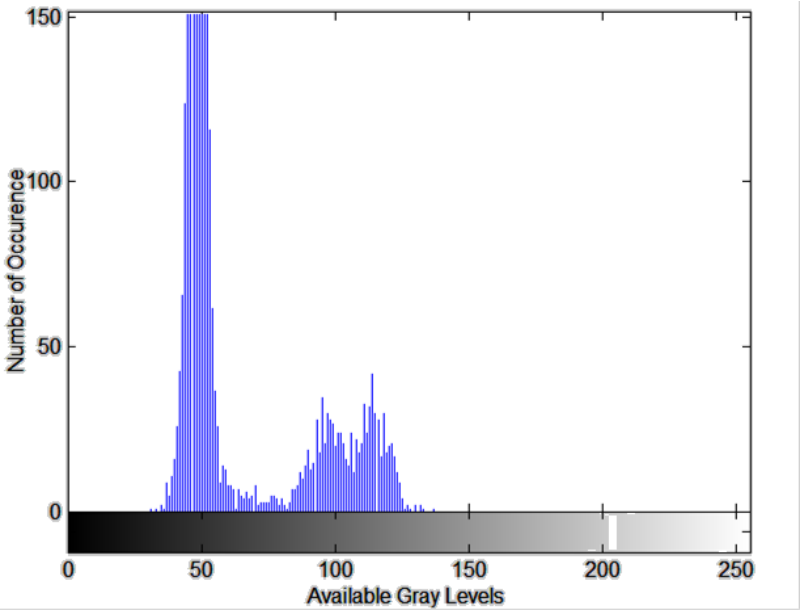

Figure 3.6: Histogram of Available gray-levels in kernel

In Proposed KMTF based BWCA approach, the proposed scheme uses reduced encoding map instead of full encoding map, in traditional MTF encoding as described in section 2.5.1 the MTF encoder generates the symbol map with respect to the type of data being encoded (in case of text all the available symbols and alphabets), but in proposed method only present symbols are used to generate symbol map, this statistical method of symbol map generation is done by calculating the probability of all gray-levels and as kernel of BWT matrix does not contain all the available gray-levels (and it is highly unlikely that a single kernel may contain all gray-levels) so the symbol map is generated locally on KMTF

encoding and the information of available gray-levels is sent along with the output of KMTF in order to decode the data into original form.

The difference of both traditional technique of encoding data with full symbol map is illustrated in Table [1] in which first column shows the string "NNBAAA" which is the BWT form of "BANANA", whereas column two contains a sample symbol map of all available characters.

String	Symbol Map
N,N,N,B,A,A,A,A	AB……..N……Z
13,N,N,B,A,A,A,A	NAB……..……Z
13,0,0,B,A,A,A,A	NAB……..……Z
13,0,0,2,A,A,A,A	BNA……..……Z
13,0,0,2,2,A,A,A	ABN……..……Z
13,0,0,2,2,0,0,0	ABN……..……Z

Table 1.1: Encoding of sample string in column 1, with symbol map shown in column 2

Table 1.2 shows the encoding of same sample string with reduced symbol map; the output of the reduced symbol map contains less index values as compared to the original scheme in the proposed schemes this reduced encoding map is referred as *Encoding Symbols*.

String	Symbol Map
N,N,N,B,A,A,A,A	ABN
2,N,N,B,A,A,A,A	NAB
2,0,0,B,A,A,A,A	NAB
2,0,0,2,A,A,A,A	BNA
2,0,0,2,2,A,A,A	ABN
2,0,0,2,2,0,0,0	ABN

Table 1.2: Encoding of sample string in column 1, with reduced symbol map shown in column 2

In concluding words, we can analyze that the index generated by the traditional schemes contains higher values, but in our proposed scheme the generated values form more

patterns and these patterns are likely to achieve higher compression ratio when entropy coders are applied.

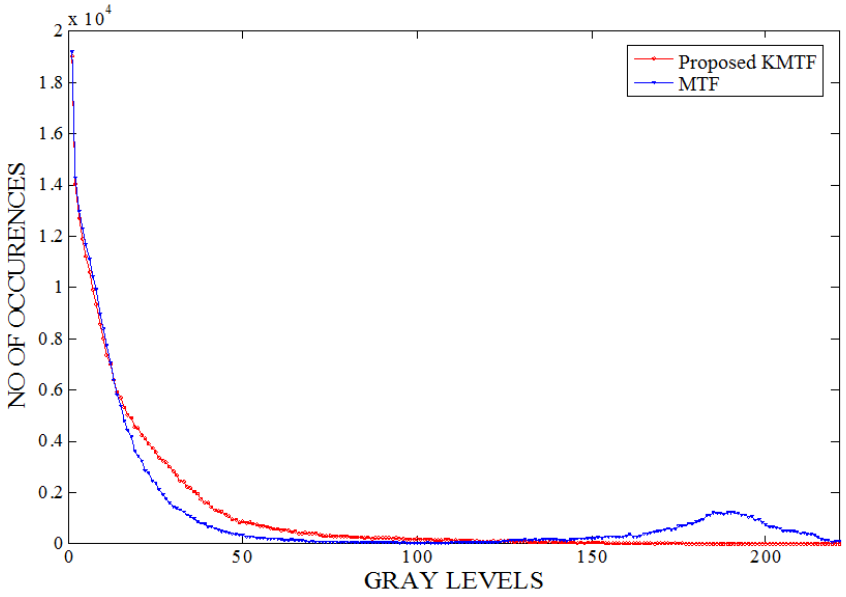

Figure 3.7: Comparison of histogram generated by traditional MTF scheme and KMTF scheme

Figure 3.7 shows the comparison of the outputs of MTF scheme and KMTF scheme in which we can analyze that MTF encoding based output contains even higher level of indexes which range from 150-220 whereas by using KMTF with reduced symbol map the indexes generated are concentrated and are only prominent within range of 0-100, this reduction of indexes increase the performance of entropy coder in the last stage of BWCA.

3.2.4 Improvement in Entropy Coder stage

Entropy coder is the final stage BWCA in which data is compressed into variable length coding, the technique used in the original BWCA scheme uses Huffman coding, the Huffman coding is most popular and very efficient technique and that's why M. Burrows and D. J. Wheeler suggested to use Huffman coding in entropy coding stage of BWCA.

As far as the use of Huffman coding is concerned, it is very efficient technique but the only problem with this technique is that when the data is encoded and stored (or transmitted)

then *dictionary* of symbols is also stored, because this dictionary contains the information regarding the one to one correspondence of data with variable codes generated by Huffman.

The dictionary of Huffman contains variable length code instead of standard 8-bit ASCII codes and our conventional memories are fabricated in a way to store standard data into registers, so storing the dictionary into memory is not a practical solution. This issue of Huffman coding is the problem which was faced by compression algorithms.

Canonical Huffman Encoding is the solution to above issue, in canonical huffman encoding the dictionary of available symbols is not stored, instead only the lengths of the codes assigned is stored, the codes generated by canonical huffman encoders are similarly uniquely distinguishable, but these codes are generated with the algorithm shown in Figure 3.8. After generation of codes by canonical huffman generator, the data is coded with these codes, and the length of the codes are sent along with data, this process of sending data instead of dictionary helps to improve compression ratio.

```
First compute the first code and initialize it's all length time value to 0s
Repeat for value of symbol = 1 to maximum_symbol
        Code (symbol) = Code (symbol-1) + 1;
        while (length(code(symbol)))==Length(original_huffman_code(symbol))
                append(code(symbol),'0');
```

Figure 3.8: Pseudo-code for the generation of Canonical Huffman codes.

With the help of canonical huffman codes the need of dictionary keeping is no longer needed, so using these codes the compression ratio increase even further.

In proposed scheme we have used canonical huffman encoding and arithmetic coding which is discussed in section 2.7.

3.3 Summary

In this chapter the structure of proposed scheme is discussed in satisfactory detail, in which each stage of proposed scheme is defined with the input and output with all the factors which affect the overall efficiency of system. The two versions of proposed scheme ware discussed, the importance and working of kernels in proposed scheme is described, and the comparative analysis of kernel based BWCA and simple BWCA was illustrated with the help of histogram analysis.

Chapter 4

Implementation and Experimental results

4.1 Implementation Overview

In this chapter the complete model of proposed scheme is discussed with the working flow of the system, in the research phase the complete scheme proposed KMTF based BWCA was implemented and tested in the MATLAB.

There are many potential ways to treat the image data for compression, mostly compression softwares read binary data presented in the image, while on other hand some transform the image data using merged RGB data in which the data of pixels are stored in the entities which consists of the values of Red, Green and Blue amplitudes of gray-levels. Besides both of these techniques we adopted the technique of using image base data presented by Alexander Akimove [19] in which the Red, Green and Blue slices of image are separated.

The process of separating slices provide two vital advantages over using merged data which are,

- The values of Red, Green and Blue are consistent with neighboring pixels.
- The transition of sliced values is less as compared to merged data.

These advantages of RGB slices help to increase the inter-relation of pixels and hence by increasing the overall compression of scheme. In merged data for RGB images, the value of a pixel describe the intensity of gray-levels of the pixel, so whenever any of the color component vary in the amplitude then the number representing that pixel changes its value abruptly, this sudden abruption of amplitude value result in poor relation with neighboring pixels, this step of data acquisition for color images may seems miniature but is very vital when the data available in image is smooth, and no edges or color changes are available frequently.

The Flow chart for compression phase of the proposed scheme is shown in Figure 4.1, whereas Figure 4.2 illustrates the decompression part of proposed scheme, in implementation phase many variants of proposed scheme ware formulated and tested, an extensive amount of

sample images ware tested and analyzed, only small portion of the experimentation results are illustrated in this chapter.

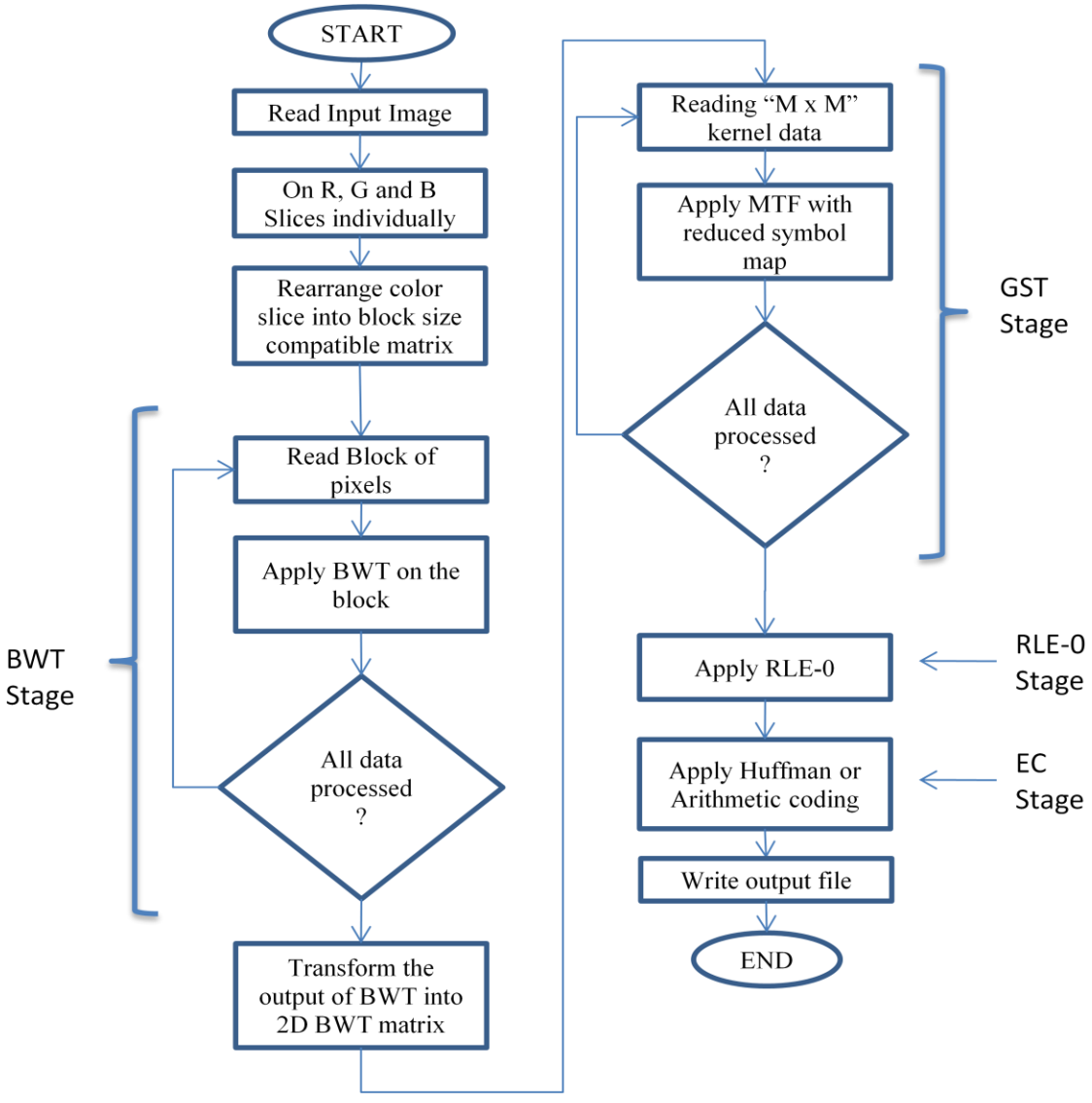

Figure 4.1: Flow chart for the compressing phase of proposed Scheme

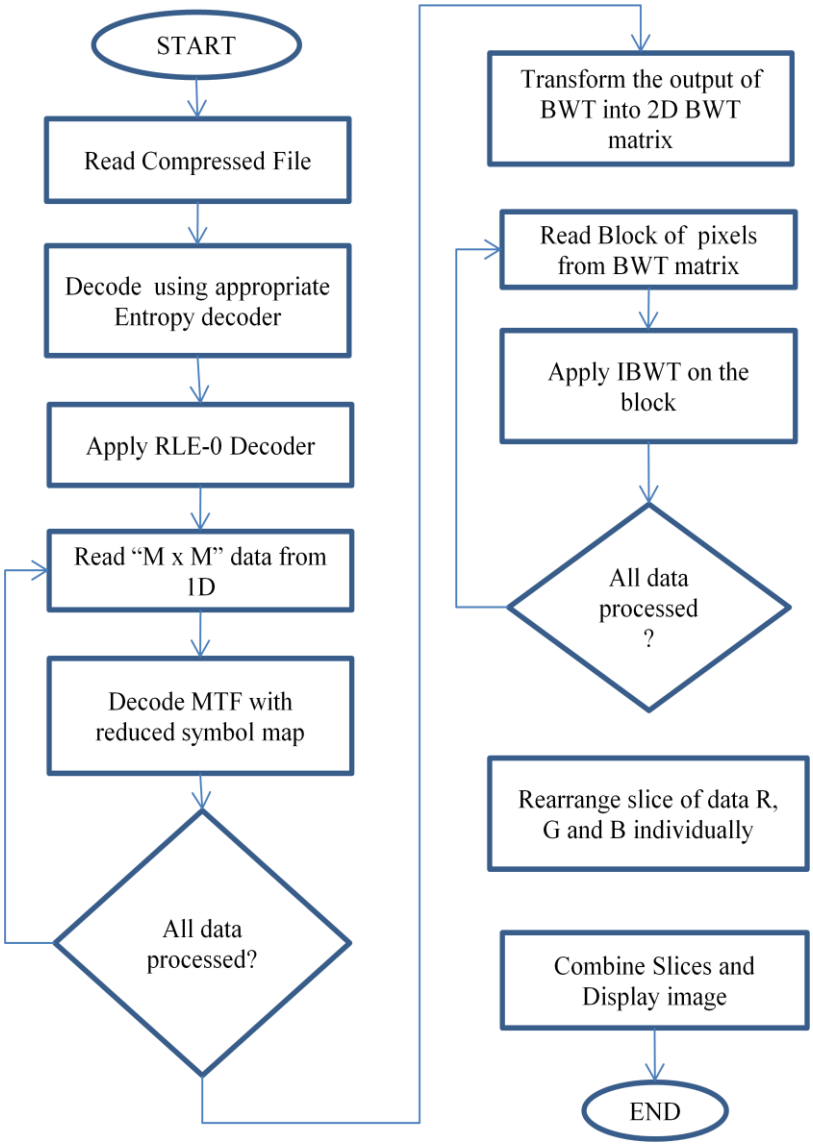

Figure 4.2: Flow chart for the compressing phase of proposed Scheme

Figure 4.1 and [2] shows the flow chart of compression and decompression process, these flow charts does not represent each and every detail of each step, the detail information on each stage is discussed in chapter 3.

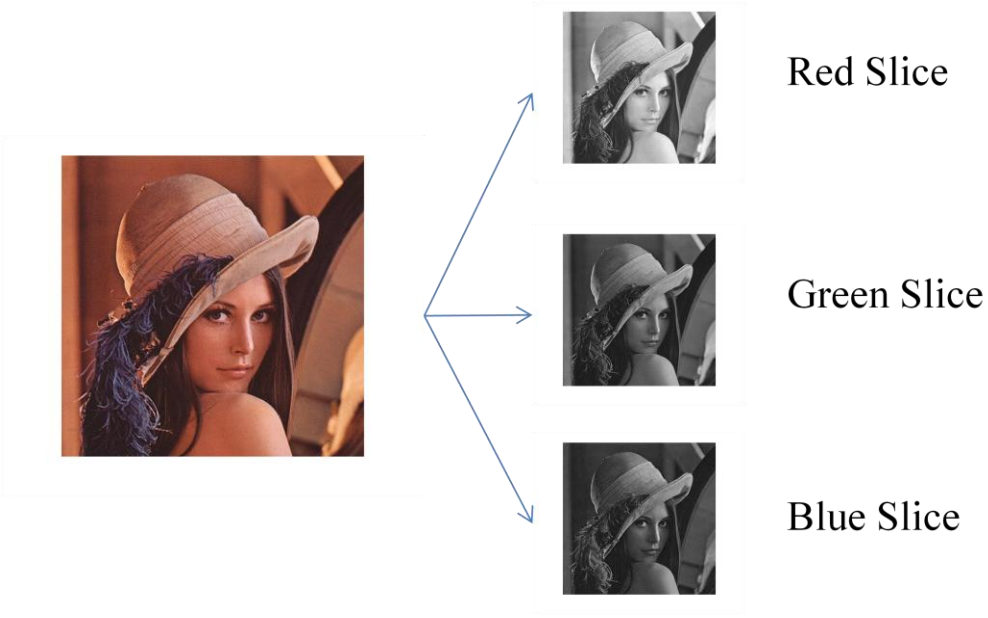

Figure 4.3 Standard color image Lenna, with three RGB slices.

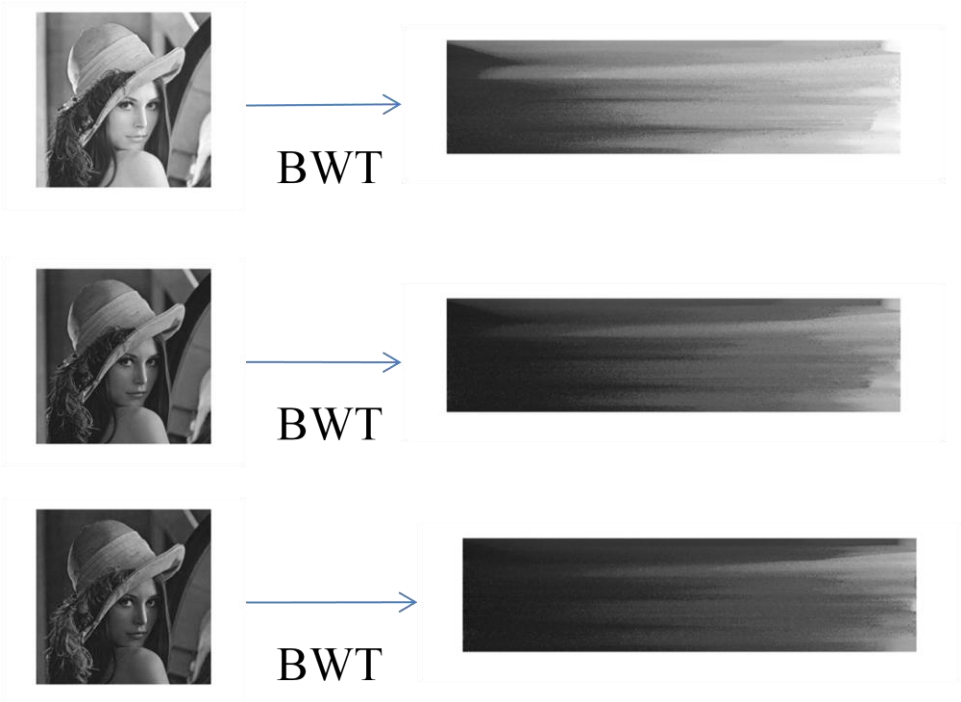

Figure 4.4: R, G and B slices with their respective BWT matrix.

Figure 4.3 shows the slicing of lenna image, and Figure 4.4 shows the transformation of slice of color image to BWT matrix, after GST stage (KMTF) transformation, the data is converted into indexes, so the histogram of Red slice of image with respective histogram is shown in Figure 4.5 the histogram, whereas Table 4.1 results of image compression with BWCA along with compression with proposed methods

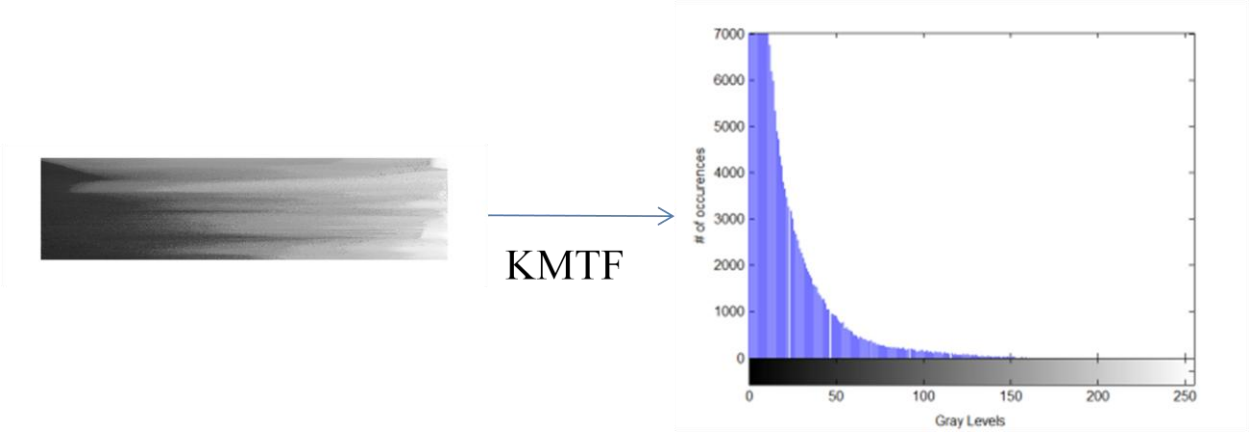

Figure 4.5: Red slice of BWT image, with histogram of MTF transformed data.

Lenna Image	Original Size	Original BWCA	Proposed KMTF based BWCA	
			Canonical Huffman	Arithmetic Coding
R	262144	204340	197111	193151
G	262144	197750	192033	188047
B	262144	196350	191165	187082
Total	786432	598440	580311	568280
CR	--	1.31	1.36	1.38

Table 4.1: Compression Ratios of Lenna image.

During the experimentation phase of KGST based BWCA and BWCA, Extensive images ware selected to test and verify the improvements of proposed scheme, standard TIFF (Tagged Image File Format) extension based images which are used in Digital image processing ware selected and ware used in compiling results, such standard images are Lenna, lake, baboon, Tiffany, peppers and splash (which are numbered as 8, 4, 5, 3, 9 and 13 respectively), these image ware obtained from the online database of University of Southern California [20]. With these standard images, many of random images ware also tested with proposed method, some of them are presented in this thesis. Figure 4.6 shows subset of images which ware used to analyze the performance of system.

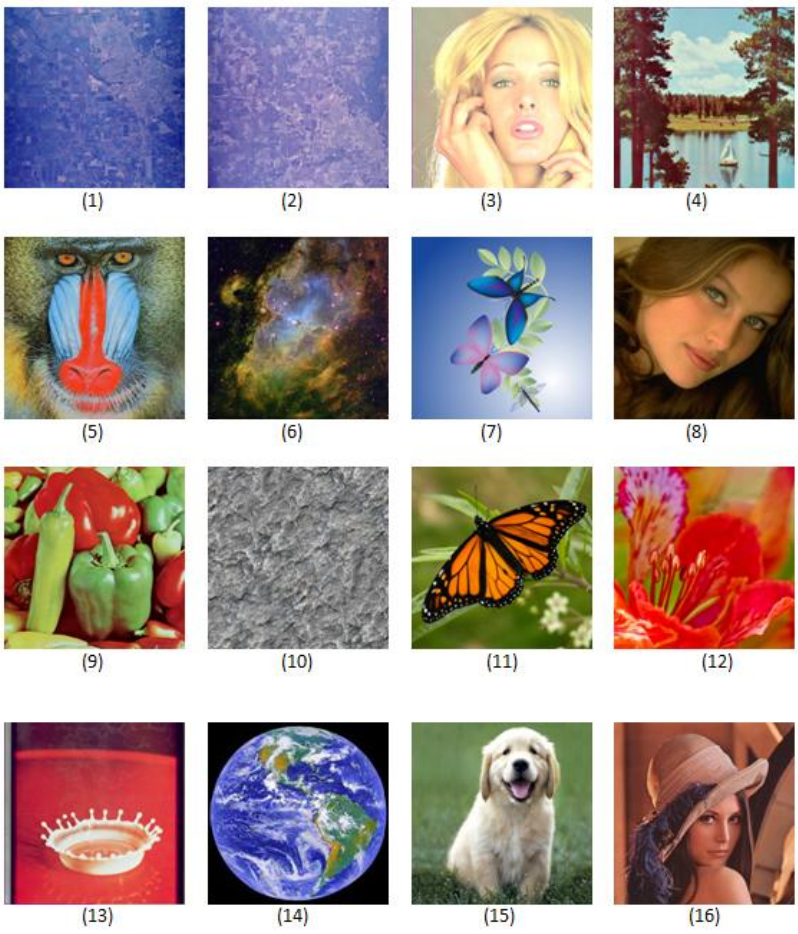

Figure 4.6: Preview of images used for compression with proposed and original scheme. [18]

In evaluation phase it was noticed that different kernels sizes produce different compression ratios, these kernels however found to be related to block size, for higher block size the higher kernel size ware performing better and vice versa, since we have tested proposed scheme with different block sizes, for illustration purpose the results complied with 1k block size are presented in this report.

Image #	Original BWCA Scheme		Kernel MTF Scheme					
			Kernel Size 128x128		*Kernel Size 64x64*		*Kernel Size 32x32*	
	BYTES	*CR*	*BYTES*	*CR*	*BYTES*	*CR*	*BYTES*	*CR*
1	650542	**1.21**	**628844**	**1.25**	629965	**1.25**	639395	**1.23**
2	670258	**1.17**	**648762**	**1.21**	649473	**1.21**	658127	**1.19**
3	549166	1.43	532156	1.48	**530653**	1.48	536224	1.47
4	652472	1.21	628233	1.25	**627947**	1.25	640616	1.23
5	723781	**1.09**	**686452**	**1.15**	687860	**1.14**	698967	**1.13**
6	415177	1.89	399459	1.97	**395949**	1.99	396468	1.98
7	641768	1.23	618311	1.27	**616103**	1.28	625716	1.26
8	503860	1.56	487706	1.61	**478056**	1.65	477431	1.65
9	623977	1.26	602724	1.30	**595570**	1.32	602768	1.30
10	756982	**1.04**	**730707**	**1.08**	734363	**1.07**	744760	**1.06**
11	559798	1.40	543863	1.45	**539008**	1.46	544248	1.44
12	510997	1.54	491322	1.60	**485622**	1.62	488295	1.61
13	509005	1.55	492575	1.60	486611	1.62	**486121**	1.62
14	600167	1.31	572302	1.37	**570381**	1.38	577871	1.36
15	520852	1.51	497955	1.58	**492453**	1.60	496689	1.58
16	606460	1.30	586168	1.34	**580312**	1.36	584345	1.35
AVERAGE	593454	1.36	571721	1.41	568770	1.42	574877	1.40

Table 4.2: Results of sample images with different kernels, the best compression ratio achieved by kernel is shown in bold. [18]

With 1k block size the kernel size of 64 x 64 achieved higher compression ratio with respect to 32 x 32 and 128 x 128, the reason behind this relation is that when small kernels are selected then size of encoding symbols generated by reduced encoding maps are higher, and when kernel size is chosen large, then gray-levels in the kernel vary in high range, and hence the compression ratio is affected with this phenomena.

Table 4.3 shows the results of proposed scheme with optimal kernel for 1k bytes block size, in which the compression ratio of BWCA is compared with two versions of proposed method.

Image #	Original BWCA Scheme		Kernel MTF Scheme			
	Huffman Encoder	*CR*	*Canonical Huffman Encoder*	*CR*	*Arithmetic Encoder*	*CR*
1	650542	1.21	632365	1.25	623484	1.26
2	670258	1.17	651873	1.21	642760	1.23
3	549166	1.43	533053	1.48	519627	1.52
4	652472	1.21	630347	1.25	618936	1.27
5	723781	1.09	690260	1.14	680919	1.16
6	415177	1.89	398348	1.99	385415	2.05
7	641768	1.23	618503	1.28	608832	1.30
8	503860	1.56	480456	1.65	463579	1.70
9	623977	1.26	597970	1.32	587850	1.34
10	756982	1.04	736763	1.07	732223	1.08
11	559798	1.40	541408	1.46	527234	1.50
12	510997	1.54	488022	1.62	473722	1.67
13	509005	1.55	489011	1.62	497455	1.59
14	600167	1.31	572781	1.38	561994	1.40
15	520852	1.51	494853	1.60	479128	1.65
16	606460	1.30	582712	1.36	569817	1.38
AVERAGE	593454	1.36	571170	1.42	560811	1.44

Table 4.3: Comparison of compression ratios of BWCA with two versions of proposed scheme. [18]

In experimental results we discovered that Arithmetic coding at entropy coding stage optimize the output of algorithm further.

Even with small and large kernels, the output of proposed scheme produce higher compression ratios, but to achieve best results with kernel based GST then proper kernel exploits the redundancy of available gray-levels in efficient manner.

Table 4.4 shows the time taken by each image at GST stage, the time shown below (in seconds) is calculated using MATLAB 2008a, the machine used to calculate experimental results has specifications as follows: 1GB RAM, Windows 7 (32-bit) OS and 926 MB Virtual Memory on Intel Pentium D CPU 3.4 GHz

Image #	KMTF (Encoding)	Proposed Scheme (Encoding)	KMTF (Decoding)	Proposed Scheme (Decoding)
1	44.60	44.86	11.97	11.89
2	43.65	39.38	12.24	10.45
3	38.22	37.10	10.93	9.03
4	39.29	39.38	12.09	9.93
5	43.41	39.19	12.44	10.43
6	44.47	31.66	8.05	7.10
7	42.47	39.14	12.25	10.20
8	42.18	33.59	8.24	8.40
9	43.40	35.31	9.04	9.12
10	40.46	37.04	9.70	9.79
11	45.75	34.62	8.67	8.81
12	39.30	32.24	7.71	7.81
13	39.34	37.69	8.45	9.02
14	41.12	34.61	8.20	8.56
15	40.67	33.41	7.27	7.88
16	38.62	37.23	8.71	9.62
AVERAGE	41.69	36.65	9.75	9.25

Table 4.4: Comparison of compression ratios of BWCA with two versions of proposed scheme [18]

Figure 4.7 illustrates bar diagram of comparison, in which Red and Green bars represent proposed schemes, and it is clearly analyzable that in each image, the compression ratios achieved by proposed schemes are higher.

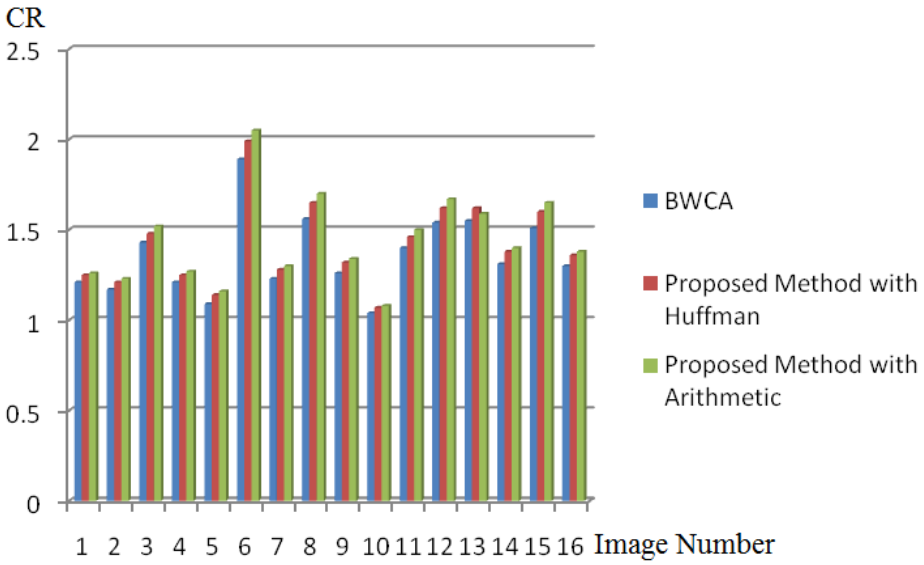

Figure 4.7: Bar diagram of compression ratios achieved by BWCA and two versions of proposed schemes.

4.2 Summary

In this chapter the implementation phase of the proposed scheme is presented, in which result of image-set are presented and the analysis of results calculated from proposed scheme and with original schemes are discussed.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis a novel approach of lossless image compression; Kernel based Global Structure Transform (GST) for Burrows Wheeler Compression Algorithm (BWCA) is presented. Original BWCA scheme was designed at first for the text based compression, but later on the efficiency of the algorithm proved its potential in various fields, BWCA based compression schemes achieves top compression in the field of data compression, the proposed scheme is the modified and improved version of BWCA, in which the improvements has been made in order to use the scheme to achieve even higher compression ratios for lossless image compression.

The proposed KGST based BWCA scheme inherits the block-sorting technique from original scheme and due to that inherited technique, BWCA achieved higher compression among other compression techniques, the issues of BWCA are targeted in the proposed method so that to increase the efficiency and reducing the problems faced by BWCA.

KGST based BWCA targeted main issues of BWCA, which are Memory utilization and Speed. In proposed scheme not only issues of BWCA ware targeted but improvements ware also implemented and tested, results and comparative analysis of both techniques proved that these improvements and the modification achieved the higher compression with less memory requirements and achieved higher speed.

The performance of the BWCA scheme and performance of proposed scheme are compared and presented. Experimental results illustrate that the achieved compression ratio of proposed scheme is higher on the average of multiple sample images.

Standard images as well as random images are chosen to ensure the improvements of proposed method. In modified GST (KMTF) stage multiple kernel sizes ware used to analyze the relation of kernel size with compression ratios, and results suggest the use of kernel of 64 x 64 pixels, the two other kernel sizes (32 x 32 and 128 x 128) provided improved results then original BWCA scheme but kernel size of 64 x 64 provided optimal results.

5.2 Recommendations and Future work

Kernel GST based BWCA is the improved technique which is presented in this thesis report, the need of data compression as well as image compression is always striving fields of digital communication and digital signal processing. The propose scheme is efficient in compression ratio and is fast as compared to BWCA, in this thesis report we have concentrated on implementing the modified scheme for lossless image compression but the features added to enhance the output of proposed algorithm can be inherited in many applications, such as in data compression.

Nowadays speed is the important asset and need of many organizations and application softwares. The speed achieved by the proposed scheme is faster than original with increased compression ratios, yet more improvement in the schemes can be implemented by the use of parallel algorithms and the use of advance Hardware support with these parallel algorithms.

The proposed scheme can be implemented on Hardware technology to reduce the processing time and as well as to generalize the type of data on which the scheme is working in this research. Field Programmable Gate Array (FPGA) based implementation of proposed method can result in even more faster and robust scheme.

The application of proposed algorithm can be implemented with the lossy version of image compression, these and many other improvements and applications can be implemented and used in the day to day use in order to enhance the efficient use of storage and communication channels.

References

[1] M. Burrows and D.J. Wheeler, “A Block-sorting lossless data compression”, SRC Research Report 124, Digital systems research center, Palo Alto, 1994.

[2] J. Abel, “Improvements to the Burrows-Wheeler compression algorithm: after BWT stages”, ACM Trans. Computer Systems, submitted for publication, 2003.

[3] P. Fenwick, "Block sorting text compression–final report", Technical reports 130, University of Auckland, New Zealand, Department of Computer Science. 1996.

[4] M. Schindler, “A Fast Block-sorting Algorithm for lossless Data Compression”. In Proceedings of the IEEE Data Compression Conference 1997, Snowbird, Utah, STORER, J.A. AND COHN, M. Eds. 469.

[5] K. Sadakane, “A fast algorithm for making suffix arrays and for Burrows-Wheeler transformation,” in *Proc. Data Compression Conf.* 1998, pp. 129–138.

[6] B. Balkenhol, S. Kurtz, and Y.M. Shtarkov. Modi cation of the Burrows and WheelerData Compression Algorithm. In accepted for publication at the IEEE Data Compression Conference, Snowbird, Utah. IEEE Computer Society Press, 1999

[7] Rafeal C. Gonzalez and Richard E. Woods “Digital Image Processing” 3rd Edition, SBN:013168728X

[8] Jacob Ziv and Abraham Lempel; *Compression of Individual Sequences Via Variable-Rate Coding*, IEEE Transactions on Information Theory, September 1978

[9] “Bzip2: Home”, Retrieved on 24 March, 2011.Website: http://bzip.org/

[10] Jacob Ziv and Abraham Lempel; *A Universal Algorithm for Sequential Data Compression*, IEEE Transactions on Information Theory, 23(3), pp. 337–343, May 1977.

[11] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102.

[12] J. Bentley, D. Sleator, R. Tarjan, and V. Wei, "A locally adaptive data compression scheme," CACM 29, no. 4, pp. 320-330, Apr. 1986

[13] S. Deorowicz, "Second step algorithms in the Burrows-Wheeler compression algorithm". Software-Practice and Experience 32(2), 2002, pp. 99-111.

[14] B. Balkenhol and Y. Shtarkov, "One attempt of a compression algorithm using the BWT", SFB343: Discrete Structures in Math., Faculty of Math., Univ. of Bielefeld, Germany, 1999.

[15] B. Chapin and S. R. Tate, "Higher compression from the Burrows Wheeler Transform by modified sorting," in *Proc. Data Compression Conf,*Snowbird, UT, Mar. 1998, p. 532.

[16] "SECOND BWT COMPRESSION PAGE" Retrived on 24 March 2011, Website: http://bijective.dogma.net/compresa.htm

[17] Elfitrin Syahrul, Julien Dubois, Vincent Vajnovszki, Taoufik Saidani and Mohamed Atri, "Lossless Image Compression using Burrows Wheeler Transform (Methods and Techniques)", IEEE International Conference on Signal Image Technology and Internet Based Systems, 2008.

[18] M. Asif Ali, Aftab Khan, M. Y. Javed, Aasia Khanum, “Lossless image compression using kernel based Global Structure Transform (GST)” ICET 10.1109/ICET.2010.5638494

[19] A. Akimov, A. Kolesnikov and P. Fränti, “Lossless compression of color map images by context tree modeling”, *IEEE Trans. on Image Processing*, vol. 16 (1), pp. 114-120, 2007.

[20] “SIPI Image Database” Retrived on 24 March 2011 Website: http://sipi.usc.edu/database/database.php?volume=misc&image=15#top

[21] “RFC 1951 – DEFLATE Compressed Data Formate Specification version 1.3” Retrived on 24 March 2011 Website: http://tools.ietf.org/html/rfc1951

[22] Haitao Guo and C. Sidney Burrus, "Waveform and Image Compression with the Burrows Wheeler Transform and the Wavelet Transform." Proceedings of the IEEE International Conference on Image Processing (Oct. 1997) : 65-68.

[23] Manfred Kufleitner, “On Bijective Variants of the Burrows-Wheeler Transform”, unpublished.

[24] S. Deorowicz, “Improvements to Burrows-Wheeler Compression Algorithm. Software - Practice and Experience”, 2000, 30(13), 1465-1483.