2010

Empirical analysis of BWT-based lossless image compression

Kalyan Varma Bhupathiraju
West Virginia University

Empirical Analysis of BWT- Based Lossless Image Compression

Kalyan Varma Bhupathiraju

Thesis submitted to the

College of Engineering and Mineral Resources

at West Virginia University

in partial fulfillment of the requirements

for the degree of

Master of Science

in

Electrical Engineering

Donald A. Adjeroh, Ph.D, Associate Professor (chair)

Feruz Ganikhanov, Ph.D, Assistant Professor

Arun A. Ross, Ph.D, Associate Professor

Xin. Li, Ph.D, Associate Professor

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2010

Keywords: BWT, MTF, Image, Compression, lossless Context partitions

ABSTRACT

Empirical Analysis of BWT- Based Image Compression

Kalyan Varma Bhupathiraju

The Burrows-Wheeler Transformation (BWT) is a text transformation algorithm originally designed to improve the coherence in text data. This coherence can be exploited by compression algorithms such as run-length encoding or arithmetic coding. However, there is still a debate on its performance on images. Motivated by a theoretical analysis of the performance of BWT and MTF, we perform a detailed empirical study on the role of MTF in compressing images with the BWT. This research studies the compression performance of BWT on digital images using different predictors and context partitions. The major interest of the research is in finding efficient ways to make BWT suitable for lossless image compression.

This research studied three different approaches to improve the compression of image data by BWT. First, the idea of preprocessing the image data before sending it to the BWT compression scheme is studied by using different mapping and prediction schemes. Second, different variations of MTF were investigated to see which one works best for Image compression with BWT. Third, the concept of context partitioning for BWT output before it is forwarded to the next stage in the compression scheme.

For lossless image compression, this thesis proposes the removal of the MTF stage from the BWT compression pipeline and the usage of context partitioning method. The compression performance is further improved by using MED predictor on the image data along with the 8-bit mapping of the prediction residuals before it is processed by BWT.

This thesis proposes two schemes for BWT-based image coding, namely BLIC and BLICx, the later being based on the context-ordering property of the BWT. Our methods outperformed other text compression algorithms such as PPM, GZIP, direct BWT, and WinZip in compressing images. Final results showed that our methods performed better than the state of the art lossless image compression algorithms, such as JPEG-LS, JPEG2000, CALIC, EDP and PPAM on the natural images.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to Dr. Donald Adjeroh for his support, advice, and motivation during my research and study at West Virginia University. Without his constant guidance, it would not have been possible to complete this thesis.

I take this opportunity to express my heartfelt thanks to Dr. Feruz Ganikhanov, who has guided and supported me over the past three years. Thank you very much for everything.

I would like to thank Dr. Arun Ross and Dr. Li Xin for agreeing to be on my thesis committee and for their valuable time. I want to thank all members of my research group for their valuable suggestions.

Finally, I would like to thank my family and all my friends for being there for me all the time.

Table of Contents

1. Introduction

1.1 Introduction

Digital image compression is playing an important role despite the rapid progress in digital communications and mass storage devices in the recent years. The efficient storage, manipulation and transmission of digitized pictures still remain a major challenge. In applications like video streaming, satellite imaging, medical imaging and high quality photography the size of data to be transferred is incredibly large when compared to the bandwidth available in the communication channel.

Digital image compressions algorithms can reduce the size of digital images, thus reducing the storage and transmission cost. Image compression is often more economical than increasing bandwidth or storage capacity.

Image compression is a process in which the image data is transformed into a different form which can be represented by less numbers of bits by taking advantage of the redundancy in the image data and at the same time is able to decode back to the original image. Lossless image compression schemes either treat the image as a 1-D text sequence, or make use of the 2-D contexts to improve the coding performance. Using the Burrows Wheeler Transform (BWT) [1,2,3,4] on images involves an initial stage of 2-D to 1-D conversion, and a text compression algorithm for compressing the 1-D sequence. We use spatial prediction techniques to take advantage of the 2-D context in an image before sending the data to the BWT.

They are two types of image compression schemes, namely lossy and lossless compression. In lossless image compression the exact image is reproduced from the compressed data, no information is lost in the compression process. This report describes lossless image compression using the BWT which has been very popular for text compression. Linearized image data has been used to make it suitable for compression using the BWT.

The Move - To - Front (MTF) stage is an important stage in the BWT compression pipeline [5,6,7,21].

The main focus of this report was to observe the influence of prediction, mapping, variations of MTF and context partitioning technique on the image compression capability of the BWT. Different types of predictors such as DIFF, MEAN, MED and GAP were used. The main objective of the predictors is to predict the next pixel by using the neighboring pixel values by taking advantage of the redundancy in the image data. Different variations of MTF such as MTF, TRANS, MHD, MTFW(w,r) etc have been used to see their impact on BWT-based image compression. This work also proposed a new method for BWT image compression using context partitions. In this method the BWT output is broken down into partitions and each of these individual partitions is passed over to MTF, RLE and ARI. Results were compared with various state-of-the-art lossless image compression algorithms, such as EDP[8], PPAM[9], JPEG 2000, JPEG-LS[11], CALIC[12,13,18], SPIHT[14] etc.

1.2 Major Contributions of the work

This report presents the performance evaluation of the BWT on lossless image compression. These results are based on a set of images containing natural, medical and rendered images. The following are the major contributions of this report.

1. A comprehensive performance analysis of different BWT variants on image compression.

2. The key observation that, the MTF stage in the BWT compression pipeline should be eliminated when the objective is improved performance in lossless image compression.

3. Development of a context partition based BWT image compression scheme for improved performance on the image data.

1.3 Thesis Organization

Chapter 2 provides an introduction to the field of image compression, different coding techniques and briefly describes the Burrows-Wheelers Transformation (BWT), Run Length Encoding (RLE) and Arithmetic coding (ARI).

In Chapter 3, gives introduction to different ideas such as BWT based image compression on predicted and non predicted data, compression with different variations of MTF and compression with and without MTF. Also explains the scheme of context partitioning that is developed for further improvement of image compression results.

Chapter 4, presents results, and discusses some implementation. Various results have been analyzed to measure the performance of the researched method.

In Chapter 5, summarizes the results of the study and provide suggestions for future research.

2. Background

2.1 Image Compression

2.1.1 Digital Image

A digital image can be considered as a matrix composed of pixels each of which holds the information about the intensity at that point in the image. A digital image is obtained by digitalizing an analog image. So a digital image can be considered as a matrix of integers where each integer corresponds to the intensity level at a particular point in the image. The number of intensity levels depends on the number of bits used to represent a particular intensity. For example in this case 8-bit gray scale images are used, so there would be $2^8 = 256$ distinct intensity levels, i.e. from 0-255. Digital imaging has many advantages like transmission of images across the networks and also post processing is made easier with digital images.

2.1.2 Image Compression

Image compression is the application of data compression on digital images. The main objective of an image compression scheme is to produce a system which can encode an image to a format which occupies less space by taking advantage of the redundancy in the image data, and at the same time is able to decode back to the original image. There are different types of image compression scheme like JPEG, JPEG 2000, TIFF, PNG, PGF etc.

2.1.3 Need for Image Compression

Now a day's access of multimedia data through the Internet is growing enormously and a large amount of data is also transferred across the telecommunication networks in which,

images occupy a considerably large amount of space. The invention of a digital camera also made it easy to save the digital memories which are tending to occupy a large portion of our personal storage media like CDs, DVDs, and Hard Drives etc. By compressing the data and representing it in a more concise way, can cut storage and transmission costs by a significant factor. Thus development of efficient image compression techniques continues to be an important challenge both academically and industrially. For example a 10.1 megapixel camera can save around 969 JPEG compressed images on a 4GB memory card when compared to a 250 RAW images. This shows a compression factor of about four.

2.1.4 Types of Image Compression

Image compression can be of two types-lossy and lossless. In lossless Image compression the exact original image is reconstructed from the compressed image at the decoder. This is mainly used in applications like medical Images. Examples of lossless image compression methods are JPEG-LS, GIF, TIFF and PNG etc. In lossy image compression the reconstructed image from the compressed image is degrading in quality when compared to the original image only by an acceptable value so that it is useful in some way. This is used in applications where the compression is the major issue than the quality such as streaming media and internet telephony. Examples of lossy image compression methods are JPEG[15], JPEG2000[10]. There are different image compression schemes, but each of them tends to differ from one another in complexity, implementation, speed and performance.

2.1.5 Lossless Compression

Lossless compression is achieved mainly by reducing the redundancies in the image data and typically concentrates on more efficient ways of encoding the image data. This technique is used in applications where information loss is intolerable. The advantage is that the compressed file

will decompress to an exact duplicate of the original file, having the same quality. As no data is lost the compression ratio is not all that high when compared to lossy compression.

2.2 Compression Ratio

The performance of lossless image compression schemes can be specified in terms of compression efficiency. Compression efficiency is measured by the compression ratio or by the bit rate. Compression ratio is the ratio of the size of original image to the size of the compressed image. The bit rate is the number of bits required to represent each pixel in the compressed image. For example, a 512×*512* pixel image with a bit depth of 8 requires 512×512×8 bits = 2,62,144×8 bits = 2,62,144 bytes when stored in uncompressed form. If the size of the compressed image is 65536 bytes, then the compression ratio is 262144/65536 = 4.0. Since the image has 512×*512* = 262144 pixels, the compressed file needs 65536×8/262144 = 2 bits per pixel, on average. Hence the bit rate is 2.

The compression ratio (CR) and bit rate (BR) are related. Let *b* be the number of bits per pixel (*bit depth*) of the uncompressed image. The compression ratio is given by

$$CR = \frac{b}{BR}$$

2.3 Entropy and Image Compression

Entropy is the measure of the amount of uncertainty or information in the data. The larger the uncertainty of a random variable the larger is the entropy. Let *I* represent the self information of a random variable a_k, whose $P(a_k)$ is the probability of a_k. Then

$$I(a_k) = -\log P(a_k)$$

If $P(a_k) = 1$ then $I = 0$, i.e., if it is certain that an event is going to happen then the information in that event is zero.

$$H = -\sum_{k=1}^{K} P(a_k) \log P(a_k)$$

Where $P(a_k)$ is the probability that the symbol a_k in *S* will occur.

From the above equation we can observe that the entropy is going to be high if all the symbols are uniformly distributed. The maximum entropy is obtained when all the symbols have equal probability i.e. $P(a_k) = \frac{1}{k}$, where *k* is the number of symbols.

For example, in an image with uniform distribution of gray-level intensity, i.e. p_i = 1/256, then the number of bits needed to code each gray level is 8 bits. The entropy of this image is 8 bits.

Thus an image with smaller entropy can be compressed more than an image with higher entropy. The effectiveness of a lossless compressor is measured by determining how closely its bit rate approximates the entropy of the image calculated from the probability distribution. Therefore, if the entropy of an image is 4 bits/pixel and the bit-rate of the lossless compressor is 4 bits/pixel, then the lossless compressor did the best job possible.

2.4 BWT Compression Pipeline

The BWT compression pipeline is show in Figure 2.1 and consists of the following four algorithms:

1. Burrows-Wheeler Transformation (*BWT*)
2. Move-To-Front coding (*MTF*)
3. Run-Length Encoding (*RLE*)
4. Arithmetic encoder (*ARI*).

Encoding

The basic encoding pipeline is as follows

< *inputfile* | BWT | MTF | RLE | ARI > *outputfile*

This pipeline forms the transformation and the encoding phase. The algorithms in the pipeline scheme have been used in different combinations to study their influence on the overall compression performance.

Decoding

This pipeline forms the transformation and the decoding phase.

< *compressed-file* | UNARI | UNRLE | UNMTF | UNBWT > *raw-file*

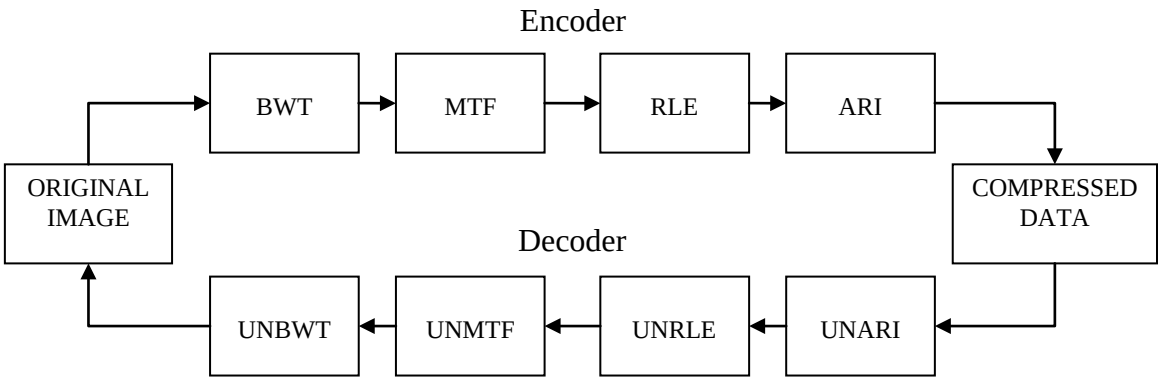


Figure2.1: BWT compression pipeline.

2.4.1 Burrows Wheeler Transform

Since its publication in 1994 by Michael Burrows and David Wheeler [1,4], the Burrows Wheeler transform has been employed in many different compression programs[23,24,25,26]. BWT is a block sorting compression algorithm used in data compression like Bzip2. Unlike most of the lossless compression algorithms which operate in streaming mode i.e. one byte at a time, BWT transform breaks the data into blocks, and compresses each block independent of the other.

Ideally larger the chunks of data available to operate the better it is, but it's limited by the amount of memory that is available. This transformation tends to group symbols together so that the probability of finding a symbol close to another instance of the same symbol is increased substantially. Text of this kind can easily be compressed with fast locally-adaptive algorithms, like move-to-front coding in combination with Huffman or arithmetic coding. The BWT has been very popular in text compression; here different variations of the BWT are tried out to see its performance in image compression.

How does it work?

BWT transformation permutes the order of the symbols. If the original string had several substrings that occurred often, then the transformed string will have several places where a single symbol is repeated multiple times in a row. Let's consider a small example dataset.

M	I	S	S	I	S	S	I	P	P	I

Figure 2.2: Sample data set.

Sample data set shown in figure 2.2 contains ten symbols. Let N be the length of the data set. To perform BWT, first make N-1 rotated copies of the input data set. Represent each of the rotated copy with an index as shown in figure 2.3.

S0	M	I	S	S	I	S	S	I	P	P	I
S1	I	S	S	I	S	S	I	P	P	I	M
S2	S	S	I	S	S	I	P	P	I	M	I
S3	S	I	S	S	I	P	P	I	M	I	S
S4	I	S	S	I	P	P	I	M	I	S	S
S5	S	S	I	P	P	I	M	I	S	S	I
S6	S	I	P	P	I	M	I	S	S	I	S
S7	I	P	P	I	M	I	S	S	I	S	S
S8	P	P	I	M	I	S	S	I	S	S	I
S9	P	I	M	I	S	S	I	S	S	I	P
S10	I	M	I	S	S	I	S	S	I	P	P

Figure 2.3: N-1 rotated copies of the data set

These N-1 copies are rearranged in the lexicographic order as in figure 2.4. There are two important points to note here. First, the strings have been sorted, but we've kept track of which string occupied which position in the original set. So, we know that the string S0, the original unsorted string, has now moved down to row 5 in the array.

S10	I	M	I	S	S	I	S	S	I	P	P
S7	I	P	P	I	M	I	S	S	I	S	S
S4	I	S	S	I	P	P	I	M	I	S	S
S1	I	S	S	I	S	S	I	P	P	I	M
S0	M	I	S	S	I	S	S	I	P	P	I
S9	P	I	M	I	S	S	I	S	S	I	P
S8	P	P	I	M	I	S	S	I	S	S	I
S6	S	I	P	P	I	M	I	S	S	I	S
S3	S	I	S	S	I	P	P	I	M	I	S
S5	S	S	I	P	P	I	M	I	S	S	I
S2	S	S	I	S	S	I	P	P	I	M	I

Figure 2.4: Lexicographic order.

Second, the first column contains all the characters in the original string in sorted order. So our original string "MISSISSIPPI" is represented in the first column as "IIIIMPPSSSS". The characters in the last column don't appear to be in any particular order, but in fact they have an interesting property. Each of the characters in the last column is the *prefix character* to the string that starts in the same row in the first column.

The output of the BWT consists of two things: a copy of the last column, and the *primary index*, an integer indicating which row contains the original first character of the buffer B. So performing the BWT on our original string generates the output string in the last column which contains "PSSMIPISSII", and a primary index of 3.

The integer 3 is found easily enough since the original first character of the buffer will always be found in last column in the row that contains S1. Since S1 is simply S0 rotated left by a single character position, the very first character of the buffer is rotated into the last column of the matrix. Therefore, locating S1 is equivalent to locating the buffer's first character position in the last column.

To get the original sequence, it must be possible to reconstruct the full table of lexicographically ordered cyclic shifts using only the last column of the table i.e. the BWT output. The key that makes this possible is that you can recreate the transformation vector from the last column and the first column of the matrix. First column can be determined by simply sorting the last column. So all you need is the BWT output and the primary index (3 in this case). As each row is a cyclic shift of every other row, the last and first columns together provide a list of all consecutive pairs of symbols.

0	P	I
1	S	I
2	S	I
3	M	I
4	I	M
5	P	P
6	I	P
7	S	S
8	S	S
9	I	S
10	I	S

Figure 2.5: Last (L) and first (F) columns.

Since by definition the strings in first column must appear in sorted order, it means that all the strings that start with a common character in last column appear in the same order in first column, although not necessarily in the same rows. Because of this, first 'S' encountered will be

followed by ‘I’, the next ‘S” encountered will also be followed by ‘I’, the next ‘S’ encountered will also be followed by ‘S’, the next ‘S’ encountered will also be followed by ‘S’. Similarly the first 'I' encountered will be followed by ‘M’, the next ‘I’ encountered will also be followed by ‘P’, the next ‘I’ encountered will also be followed by ‘S’, the next ‘I’ encountered will also be followed by ‘S’.

As the primary index is 3 we can obtain the decoded sequence as follows.

M	I									
M	I	S								
M	I	S	S							
M	I	S	S	I						
M	I	S	S	I	S					
M	I	S	S	I	S	S				
M	I	S	S	I	S	S	I			
M	I	S	S	I	S	S	I	P		
M	I	S	S	I	S	S	I	P	P	
M	I	S	S	I	S	S	I	P	P	I

Figure 2.6: Decoding sequence.

2.4.2 Run-Length Encoding (RLE)

Run length encoding is one of the oldest compression methods. This mainly takes advantage of the repetitiveness of symbols in a stream of data. Its performance depends mostly on the input data. It is most effective when a single character is repeated multiple times in a sequence.

For example, the string “*gggggddddpppeeee*” can be represented with run length encoding as “*g6*d5*p4*e4**”. That saves us 9 symbols. This code contains a flag character, a count byte, and the repeated characters. We can also see that it doesn’t make any sense to code any symbols which are repeated less than four times. Therefore a sequence “*aagggggghhhttttt*” can be encoded as “*aag6*hhht6**”.

Due to the randomness in the natural images RLE may not be very effective on its own. But it's observed to be effective when applied on the output sequence of the BWT.

2.4.3 Arithmetic coding (ARI)

Arithmetic coding [16] is one of the methods for lossless compression that encodes data by creating a code string which represents a fractional value on the number line between 0 and 1. This is a variable length code as different symbols are represented by different number of bits. The main objective of arithmetic coding is to use the minimum number of bits to represent a stream of symbol. The main logic is to use fewer bits for representing frequently occurring symbols and more bits to represent less frequently occurring symbols resulting in fewer bits in total.

Although arithmetic coding is more powerful than Huffman coding in compression ratio, arithmetic coding requires more computational power.

2.5 Preprocessing Stage

2.5.1 Predictive Coding

Predictive coding has been extensively used in image compression. The correlation between the adjacent pixels is well exploited by the use of predictive image coding algorithms. They predict the value of a given pixel based on the values of the surrounding pixels. The use of a predictor can reduce the amount of information bits needed to represent an image, due to the correlation among the adjacent pixels. Images are considered to be a sequence of pixels in row major order in lossless image compression. As each pixel is handled, all pixels preceding it are available to the decoder and hence to the encoder too, and these already-known values provide useful information to suitably bias the prediction as to the next pixel value.

The processing of each pixel consists of two separate operations. The first step forms a prediction as to the numeric value of the next pixel. Typical predictors use a linear combination of neighboring pixel values. In the second step, the difference between the predicted pixel value and the actual intensity of the next pixel is coded using an entropy coder.

We tested the BWT approach using four different prediction methods. Figure 2.7 shows a general schematic diagram of the prediction contexts used by the prediction algorithms. The symbols *x, a, b, c, d, e, f, g* in the following discussion refer to the figure.

		e	**d**	
	c	**b**	**f**	
g	**a**	**x**		

Figure 2.7: Prediction context used by various prediction schemes.

Prediction using Previous Pixel (DIFF)

In this predictor the predicted value is assumed as the value of the adjacent left pixel. The difference between the predicted value and the original value is transmitted as the output value.

$$x = a;$$

Prediction using the mean (MEAN)

In this predictor the predicted value is assumed as the mean of the adjacent left pixel and the top pixel. The difference between the predicted value and the original value is transmitted as the output value.

$$x = \frac{(a + b)}{2};$$

Median Edge Detection (MED)[17]

In this predictor the predicted value is assumed as the median of min(a, b), max(a, b), and $a + b - c$. The difference between the predicted value and the original value is transmitted as the output value. MED is the spatial predictor used in the JPEG-LS[11] standard.

$$x = \begin{cases} \min(a,b) & if\ \ c >= \max(a,b) \\ \max(a,b) & if\ \ c <= \min(a,b) \\ a + b - c & otherwise \end{cases}$$

Gradient-Adjusted Prediction (GAP)[18]

The GAP predictor works by taking into account, the gradient variations of seven neighboring pixels of the current pixel. GAP is the predictor used in CALIC [12,13,18], one of the best performing lossless image compression schemes. The GAP algorithm is given below.

$dh = |a - g| + |b - c| + |b - f|$
$dv = |a - c| + |b - e| + |f - d|$

IF $(dv - dh > 80)$
Sharp horizontal edge $x = a$

ELSE IF $(dv - dh < -80)$
Sharp vertical edge $x = b$

ELSE
$x = (a + b)/2 + (f + c)/4$

IF $(dv - dh > 32)$
Horizontal edge $x = (x + a)/2$

ELSE IF $(dv - dh > 8)$
Weak horizontal edge $x = (3x + a)/4$

ELSE IF $(dv - dh < -32)$
Vertical edge $x = (x + b)/2$

ELSE IF $(dv - dh < -8)$
Weak vertical edge $x = (3x + b)/4$

2.5.2 Mapping

The 2-dimensional image data is converted to a linear sequence. After the image is linearized the sequence is further processed before being sent to the next stage – Transformation & encoding stage.

The linear data has been subjected through several alterations to increase the suitability of the data for better processing by the transformation and compression algorithms. These are the different methods used.

Method of differences As most of the nearby pixels in an image tend to be similar, their differences will produce more runs of zeros or values close to zero, which will require fewer bits to represent.

In a modified sequence *B* of an original sequence *A* is given by,

$B_1 = A_1;$

$B_i = A_{i-1} - A_i;\ i = 2, 3, 4, \dots, \dots, \dots$

Check the example in the following scan and the resulting scan after the differences method was applied.

Scan: 128, 128, 128, 128, 128, 50, 50, 50, 50, 217, 217, 217, 216, 216, 216, 216

Result: 128, 0, 0, 0, 0, 78, 0, 0, 0, -167, 0, 0, 1, 0, 0, 0

We know that 8 bits are required to represent the 256 different grey levels. From the example shown above we can see the introduction of negative signs in the sequence, which require an extra bit to represent the signed numbers. The experimental analysis proved that preprocessing the data with the difference method resulted in a better performance than sending the linear data

unaltered to the transformation stage. The following methods have been implemented and tested in consideration of the extra bit required to represent the negative sign.

8-Bit Mapping Method: This method allows the representation of the linear sequence, by eliminating the need for the extra sign bit. It is based on a very interesting mathematical observation on the linear sequence. The range of possible error values (differences) varies based on the predicted value $\hat{I}$ and cannot be more than $2^b - 1$ distinct values. The possible range of error for e ($e = I - \hat{I}$) is $-2^b + 1 \leq e \leq 2^b - 1$, where b = # of bits. Based on the above observation, given $\hat{I}$, the predicted value at the current position, the value of e must be within the following range:

$$-\hat{I} \leq e \leq 2^b - 1 - \hat{I}$$

The index of the possible prediction errors mapped to the order $0, +1, -1, .., .., .., .., .., .. +\hat{I}, -\hat{I}, ..,$ $2^b - 1 - \hat{I}$, is calculated by the following equations.

Case 1: $\hat{I} < 2^{b-1}$ (128)

$$index = \begin{cases} 2|e|; & e \leq 0 \\ 2|e|-1; & 0 < e \leq \hat{I} \\ e + \hat{I}; & e > \hat{I} \end{cases}$$

Case 2: $\hat{I} \geq 2^{b-1}$ (128)

$$index = \begin{cases} 2|e|; & e \leq 0, |e| \leq 255 - \hat{I} \\ 2|e|-1; & 0 < e \\ 255 - \hat{I} + |e|; & |e| > 255 - \hat{I} \end{cases}$$

3. The MTF and Context Partitions in BWT-based Image Compression

3.1 Analysis of MTF in BWT-based Image Compression

3.1.1 The Move-To-Front algorithm

The Move-To-Front (MTF) algorithm is used to improve the performance of entropy encoding techniques of compression. The MTF [5,6,7] algorithm was originally proposed in [Benteley & Torjan 1986]. A Move-To-Front coder is used for preprocessing the input before it is fed to the actual compressor. Encoding works as follows: The coder maintains a list *L* containing an ordered list of all the 256 characters that can appear in the input. Whenever it receives an input character *c* it looks up the position *i* of *c* in the current ordered list of symbols *L*, outputs *i* and moves *c* to the front of *L*. Let's see how MTF works

Considering an input sequence 4136006042, the MTF coding is done as explained in the Table 3.1. The encoder accepts the symbol and is translated into the index using the list. Then the list is updated by moving the symbol to the front of the list. This updated list is used as the lookup list for the next symbol.

Table 3.1: MTF encoding.

NUMBER	SEQUENCE	LIST
4136006042	4	{0,1,2,3,4,5,6,7}
4**1**36006042	42	{4,0,1,2,3,5,6,7}
41**3**6006042	424	{1,4,0,2,3,5,6,7}
413**6**006042	4246	{3,1,4,0,2,5,6,7}
4136**0**06042	42464	{6,3,1,4,0,2,5,7}
41360**0**6042	42464	{0,6,3,1,4,2,5,7}
413600**6**042	424640	{0,6,3,1,4,2,5,7}
4136006**0**42	4246401	{6,0,3,1,4,2,5,7}
41360060**4**2	424640114	{0,6,3,1,4,2,5,7}
413600604**2**	4246401145	{4,0,6,3,1,2,5,7}
Final	4246401145	{2,4,0,6,3,1,5,7}

Let's see how the decoding works now, here the input sequence is 4246401145. The decoder accepts the index in the list which is translated into the correct output symbol using the list. Then the list is updated by moving the symbol to the front. This updated list is used as the lookup list for the next index. Note that the same initial symbol list is used for both encoding and decoding, allowing perfect reconstruction of the data source.

Table 3.2: MTF decoding

NUMBER	SEQUENCE	LIST
4246401145	4	{0,1,2,3,4,5,6,7}
4**2**46401145	41	{4,0,1,2,3,5,6,7}
42**4**6401145	413	{1,4,0,2,3,5,6,7}
424**6**401145	4136	{3,1,4,0,2,5,6,7}
4246**4**01145	41360	{6,3,1,4,0,2,5,7}
42464**0**1145	413600	{0,6,3,1,4,2,5,7}
424640**1**145	4136006	{0,6,3,1,4,2,5,7}
4246401**1**45	41360060	{6,0,3,1,4,2,5,7}
42464011**4**5	41360060	{0,6,3,1,4,2,5,7}
424640114**5**	413600604	{4,0,6,3,1,2,5,7}
Final	4136006042	{2,4,0,6,3,1,5,7}

3.1.2 Variations of MTF.

1. MTF

Upon an access for an item *x* move *x* to the front [5,6,7,22].

2. Transpose (TRANS)

Upon an access to an item *x* transpose *x* with the immediately preceding item [7,22].

3. MHD(k)

Upon a request for an item *x*, move *x* forward by *k* positions and if there are no *k* preceding items, move it forward to the first position.

4. Move-One-From-Front (MTFF)

If R(*x*) > 2 move *x* to the 2nd position, else move *x* to the front. R(*x*) is the position of *x* in the sequence [20,21].

5. Move-One-From-Front 2 (MTFF2)

If R(x) > 2 and the symbol at the front was requested at most 2 requests ago, move x to the 2nd position, else move x to the front. R(x) is the position of x in the sequence.[21]

3.1.3. Proposed Variants of MTF

1. Transpose (k) (TRANS (k))

Upon an access to an item x interchange x with the k th preceding item, if there are no k preceding items, interchange it with the first one. We experimented with different values of k, namely k=4,8,32,64,128.

2. Modified Transpose (k) (MTRANS(k))

Upon an access to an item x interchange x with the k th preceding item and if there are no k th preceding items don't change it.

3. Modified_MHD(k) (MMHD(k))

Upon a request for an item x, move x forward by k positions and if there are no kth preceding items don't change it [19,22].

4. Windowed MTF (MTFW(w,r))

This variant of MTF was designed to capture the spatial correlation in natural images. Let R(x) is the position of x in the list L and μ_x the average of the values in a window of size w (i.e. using the preceding w symbols in the input sequence. Upon access to an item x move it to the front if R(x) < 2 or | x- μ_x | <= r else move x to the second position in the list. We observed MTFW for different values of w = 2, 3, 5, 10 and r = 1, 3, 5, 10. Here w is the window size and r is the range.

5. Windowed MTF2 (MTFW2(w,r))

Upon access to an item x move it to the front if R(x) < 2 or | x- μ_x | <= r else don't do anything. We observed MTFW for different values of w = 2, 3, 5, 10 and r = 1, 3, 5, 10. Here w is the window size and r is the range.

Table 3.3 shows the Final O/P sequence and the Final list obtained when different variations of MTF's have been used on the input sequence is 4246401145.

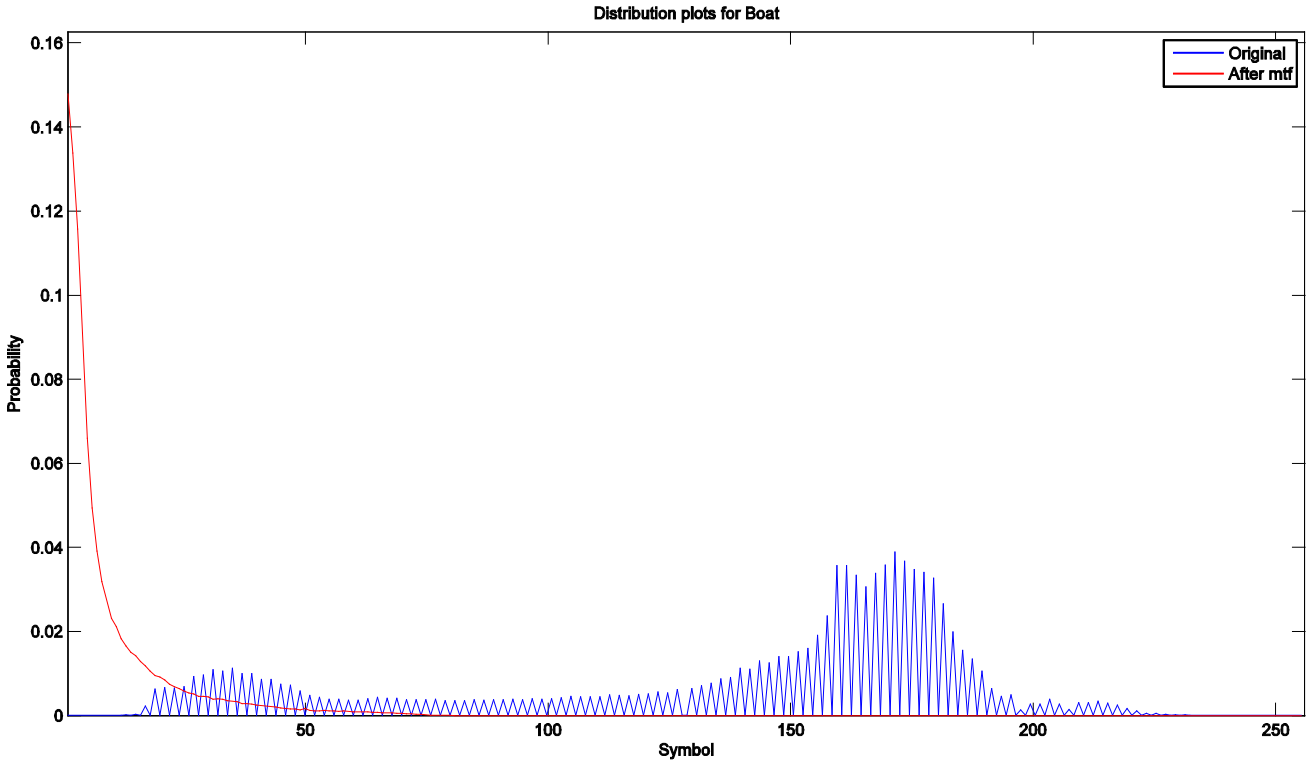
Table 3.3: O/P sequence and Final List for different MTF's when I/P sequence are 4246401145.

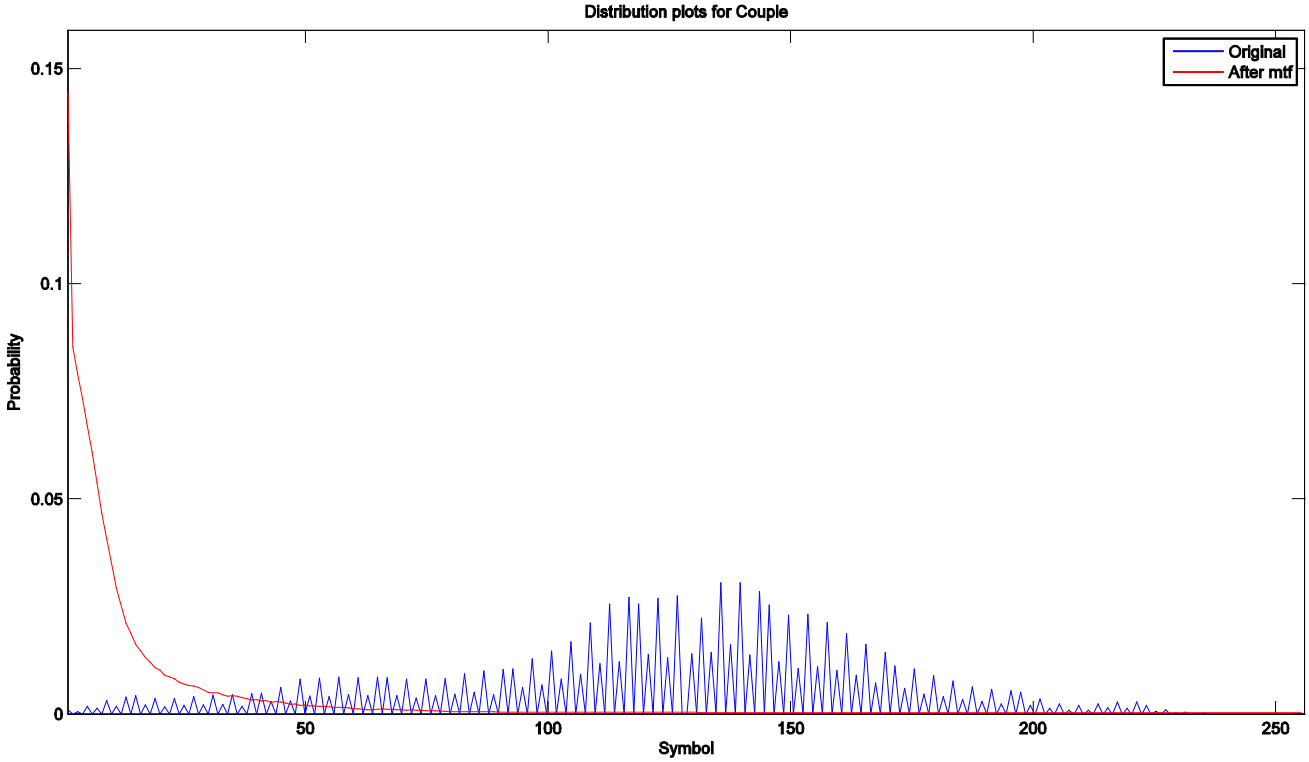
	Final O/P Sequence	Final list
MTF	4136006042	{2,4,0,6,3,1,5,7}
TRANS	4236203226	{0,4,1,2,3,5,6,7}
TRANS(2)	4446221025	{4,0,1,5,6,3,2,7}
MHD(2)	4336113126	{4,1,0,2,5,6,3,7}
MTRAN(2)	4446221125	{4,1,0,5,6,3,2,7}
MMHD(2)	4336103126	{4,0,1,2,5,6,3,7}
MTFF	4326204126	{1,5,4,0,6,2,3,7}
MTFF2	4306034116	{4,5,1,0,6,2,3,7}
MTFW(3,1)	4316033016	{4,5,1,0,6,2,3,7}
MTFW(3,1)	4316023015	{4,1,2,0,3,5,6,7}

3.1.4 Characterization of MTF outputs

1. **For Images.**

Figure 3.1 shows the empirical distribution of the original image and the MTF output. The effect on the first distribution plots for three images BOAT, COUPLE and PARROT has been observed using MTF. It also shows how MTF helps to improve the distribution.


Distribution plots for Boat
Original
After mtf
Probability
Symbol
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
50
100
150
200
250


Distribution plots for Couple
Original
After mtf
Probability
Symbol
0.15
0.1
0.05
0
50
100
150
200
250

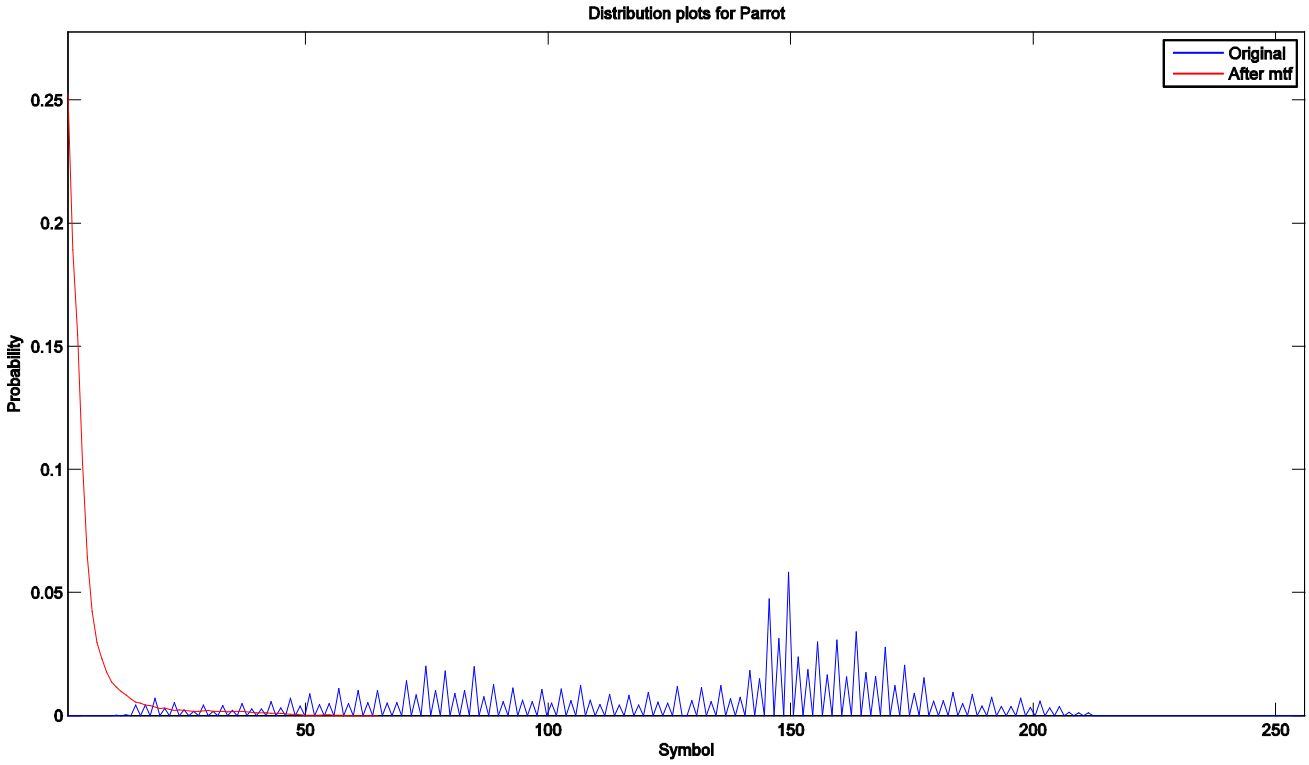


Figure 3.1: First order distribution plots of original and after MTF for boat, couple, parrot.

Figures 3.2 to 3.4 show the empirical distribution of the MTF outputs for different predictive schemes and for different images. The effect on the first and second order distribution plots for three images BOAT, COUPLE and PARROT have been observed using MTF and the three different predictors DIFF, MED and GAP. It is observed that the distributions are narrow when a predictor is used. We can notice that when the MTF is used after a predictor it actually broadens the distribution, which means that this may degrade later compression using an entropy encoder. The MED predictor seems to be the best among the three predictors which can be observed from the distribution plots. The figures also show the first order and second order symbol distributions. In both distributions first the probability of each symbol is calculated and then sorted in decreasing order of their probability.

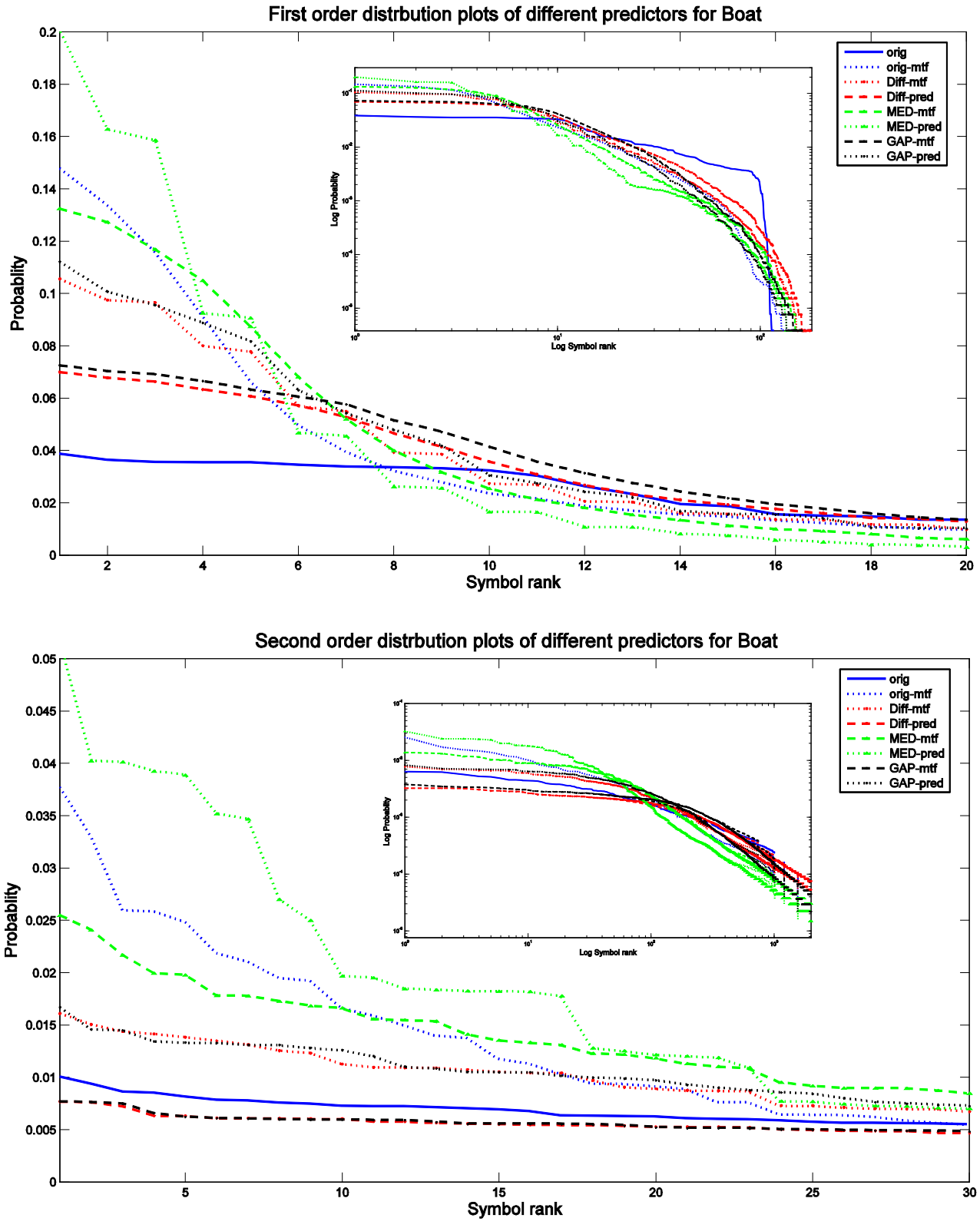


Figure 3.2: First and second order distribution plots of different predictors for boat.

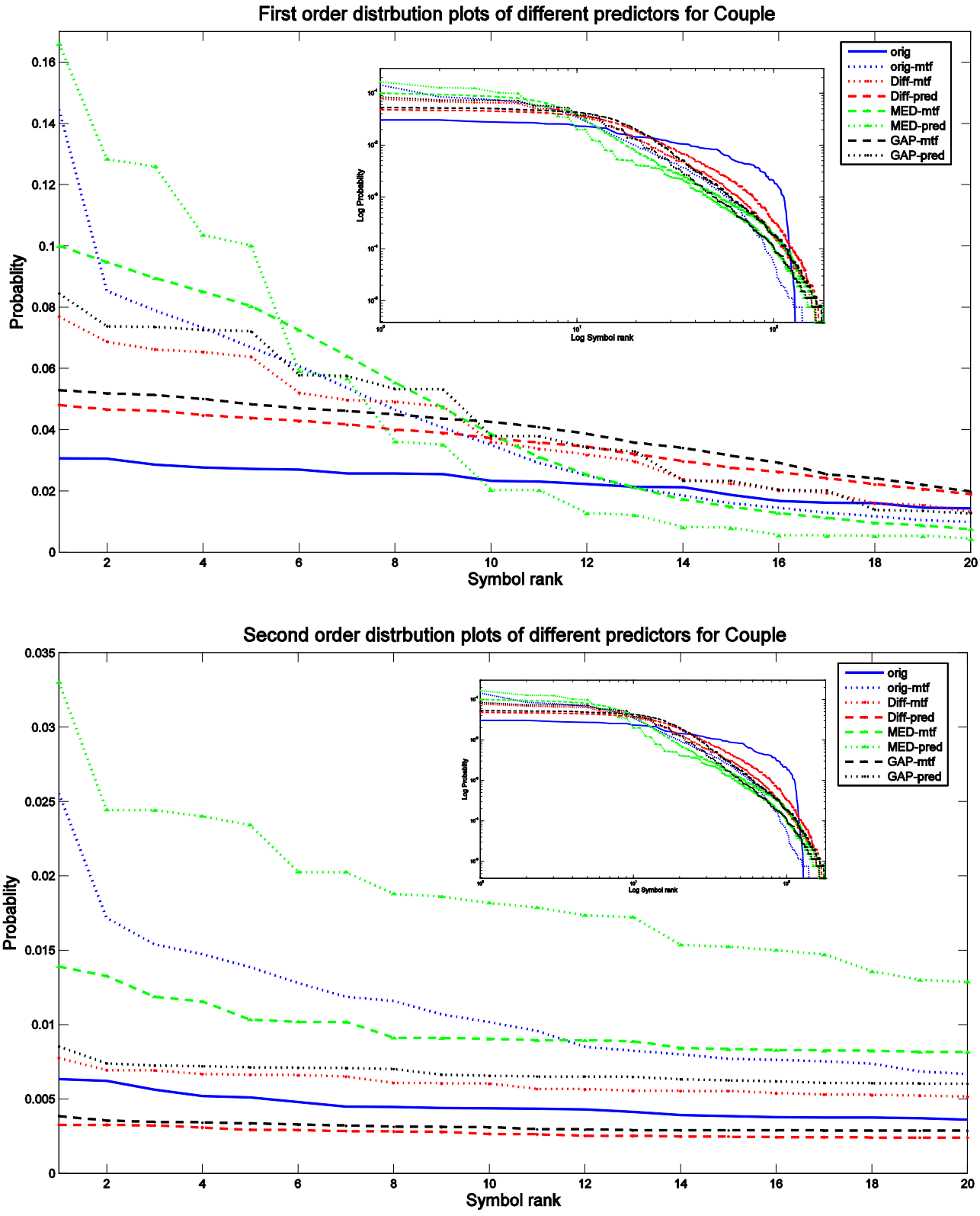


Figure 3.3: First and second order distribution plots of different predictors for Couple.

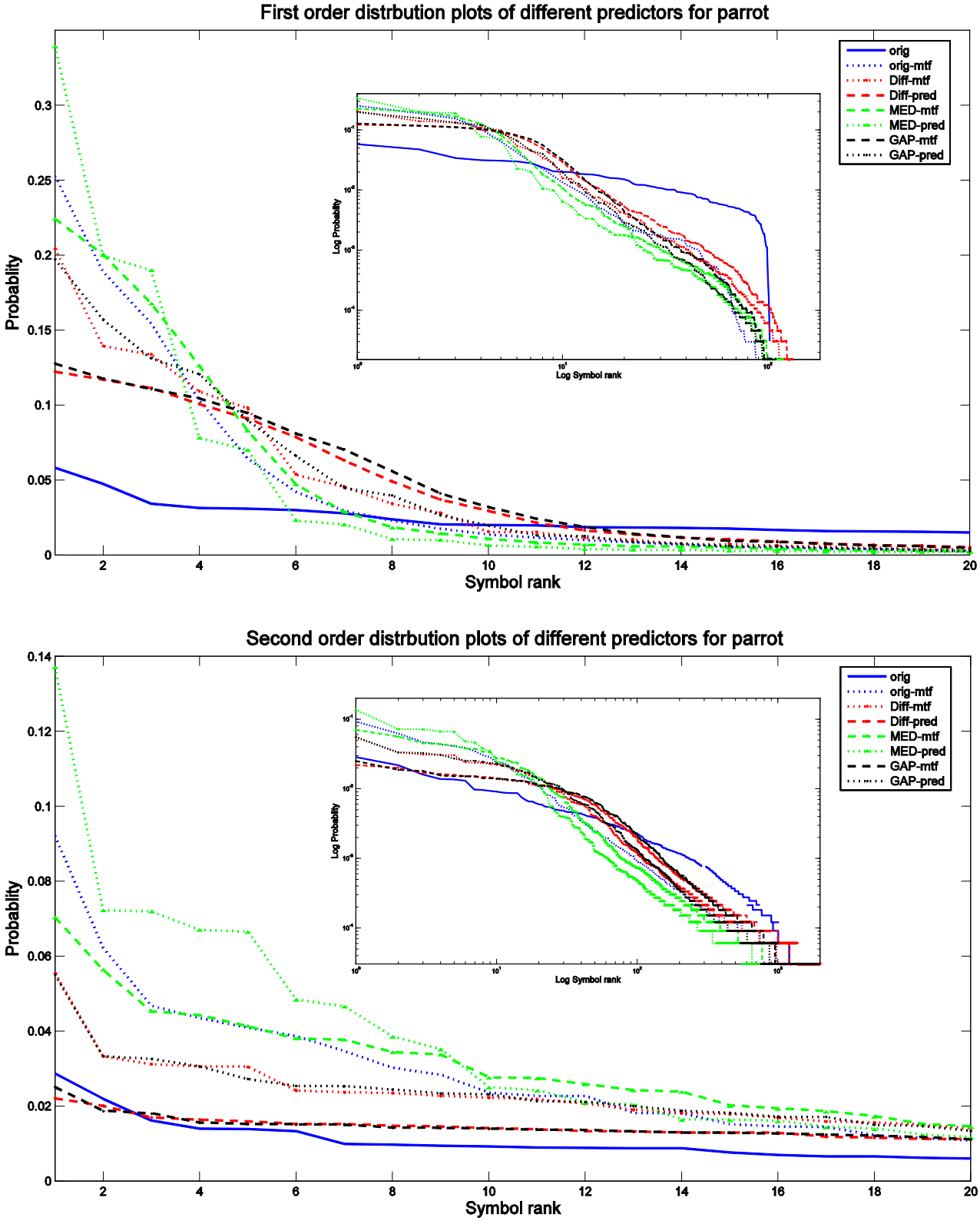


Figure 3.4: First and second order distribution plots of different predictors for Parrot.

There are 256 symbols in first order where as they are 256^2 symbols in the second order. The second order distribution is achieved by consider a pair of symbols as a single symbol. The scales on each of the plots have been adjusted in such a way to show the characteristics of the distribution plots for symbols with the highest probability. Only the first 20-30 symbols are shown in each plot. The log plots have also been included for better illustration.

2. For text files.

The distribution plots for text data have also been plotted to observe the reason why MTF does not help for BWT based Image compression. As expected the distribution is improve by MTF in the case of text data. We plotted the first and second order distribution plots for BIBLE.TXT, PROGL, TRANS and WORLD192.TXT which are seen in the figures 3.5, 3.6, 3.7 and 3.8 respectively. The text files have been obtained from the Canterbury corpus.

3. For Laplace distribution sequence.

A data sequence with laplace distribution having mean 128 and standard deviation 14 has been generated. The first and second order distribution plots for the original and MTF outputs have been plotted using this sequence in figure 3.9. It is observed that the MTF broadens the distribution of the input if it is laplacian, which doesn't help for the compression. As known the predictive errors generally tend to have a laplacian distribution. Thus the improvement in the compression ratio with the removal of MTF from the BWT Pipeline makes sense.

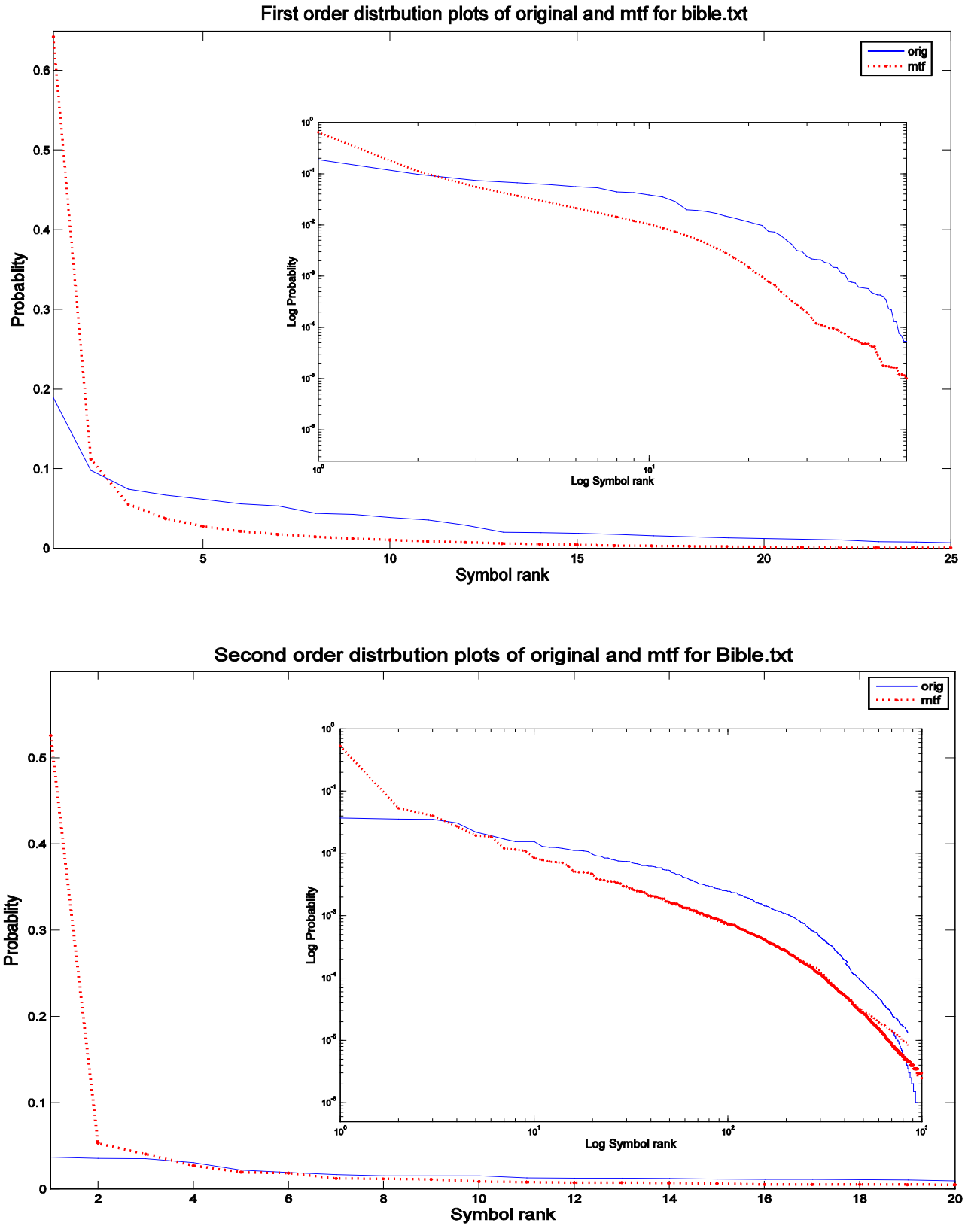


Figure 3.5: First and second order distribution plots of different predictors for Bible.txt.

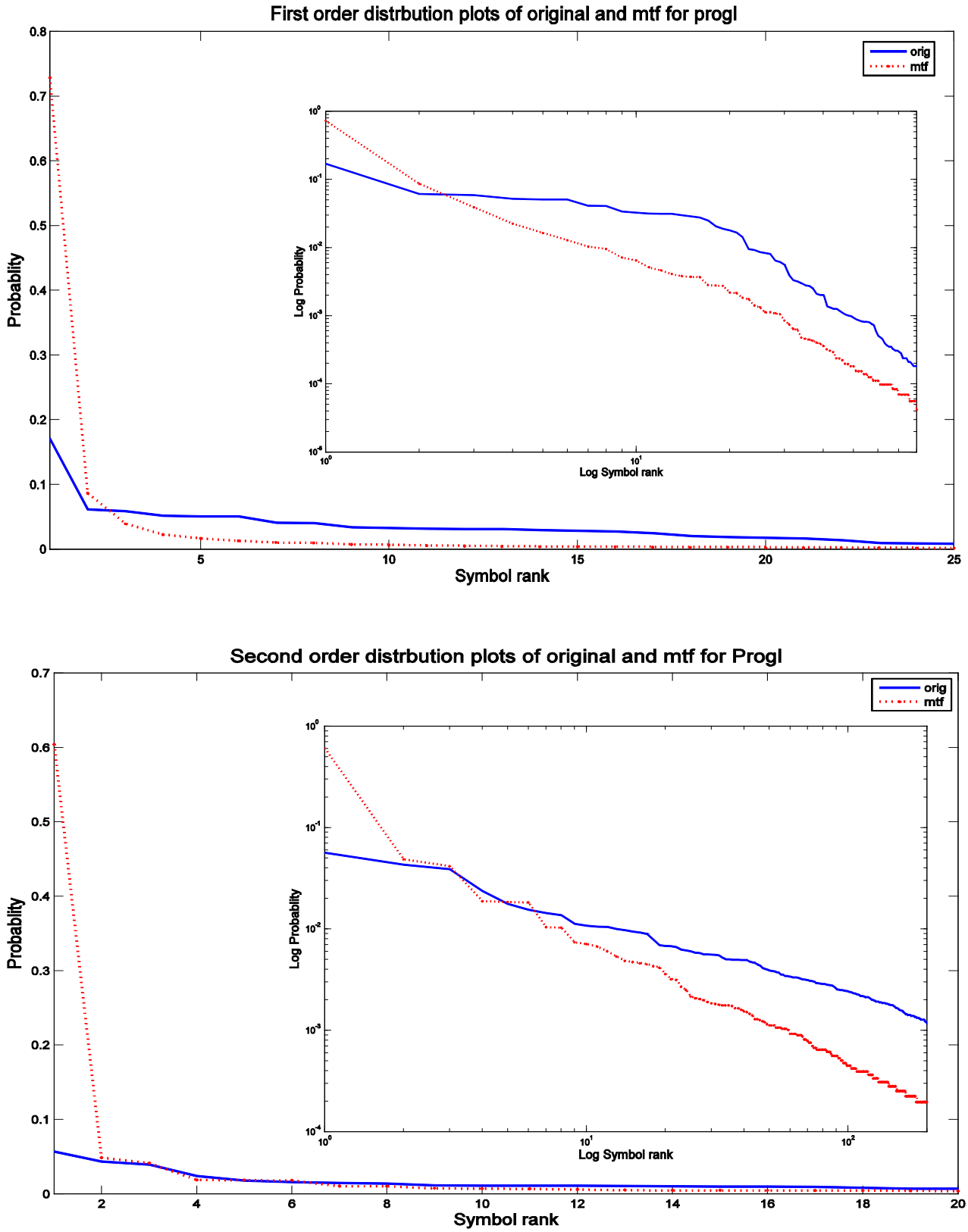


Figure 3.6: First and second order distribution plots of different predictors for Progl.

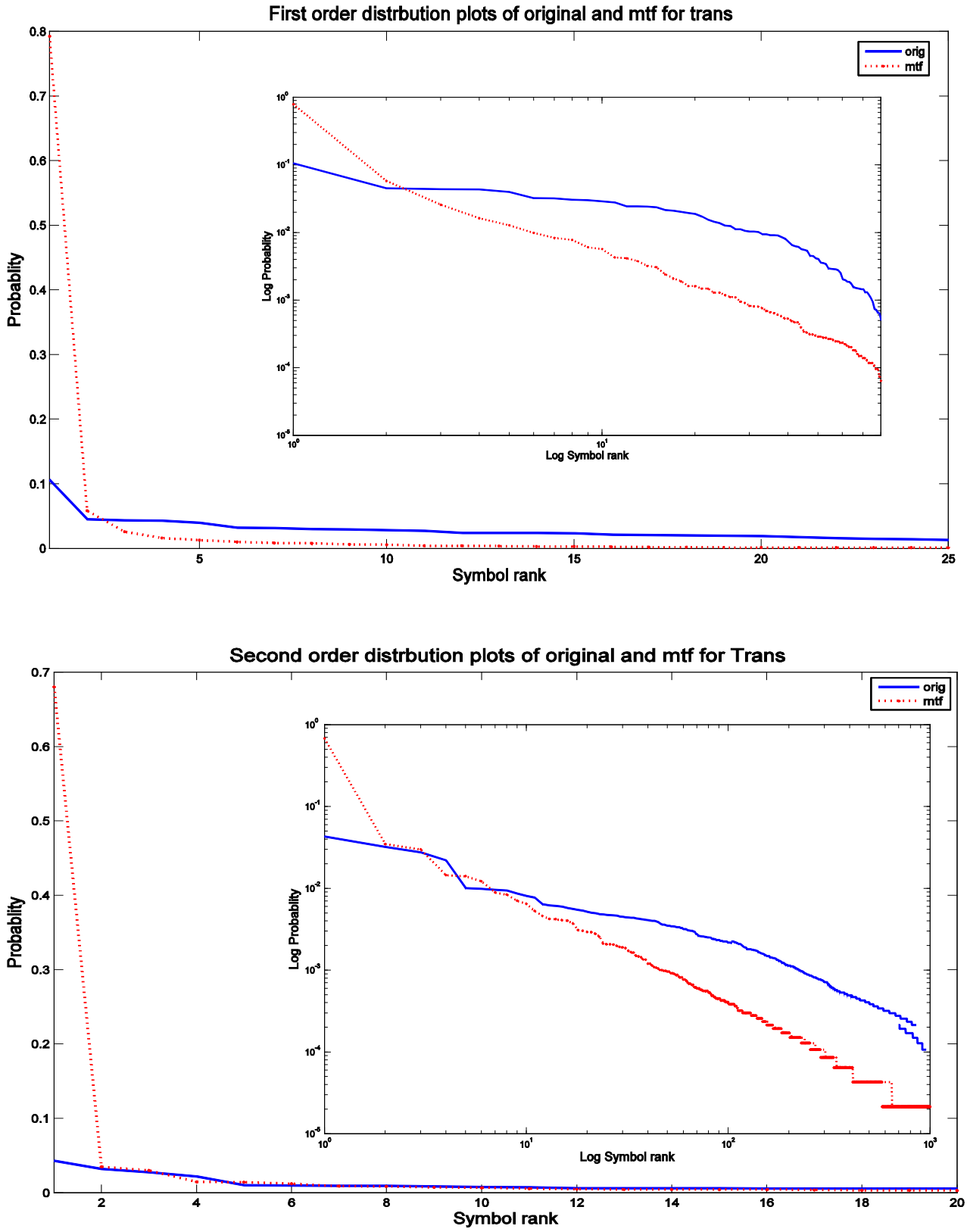


Figure 3.7: First and second order distribution plots of different predictors for Trans.

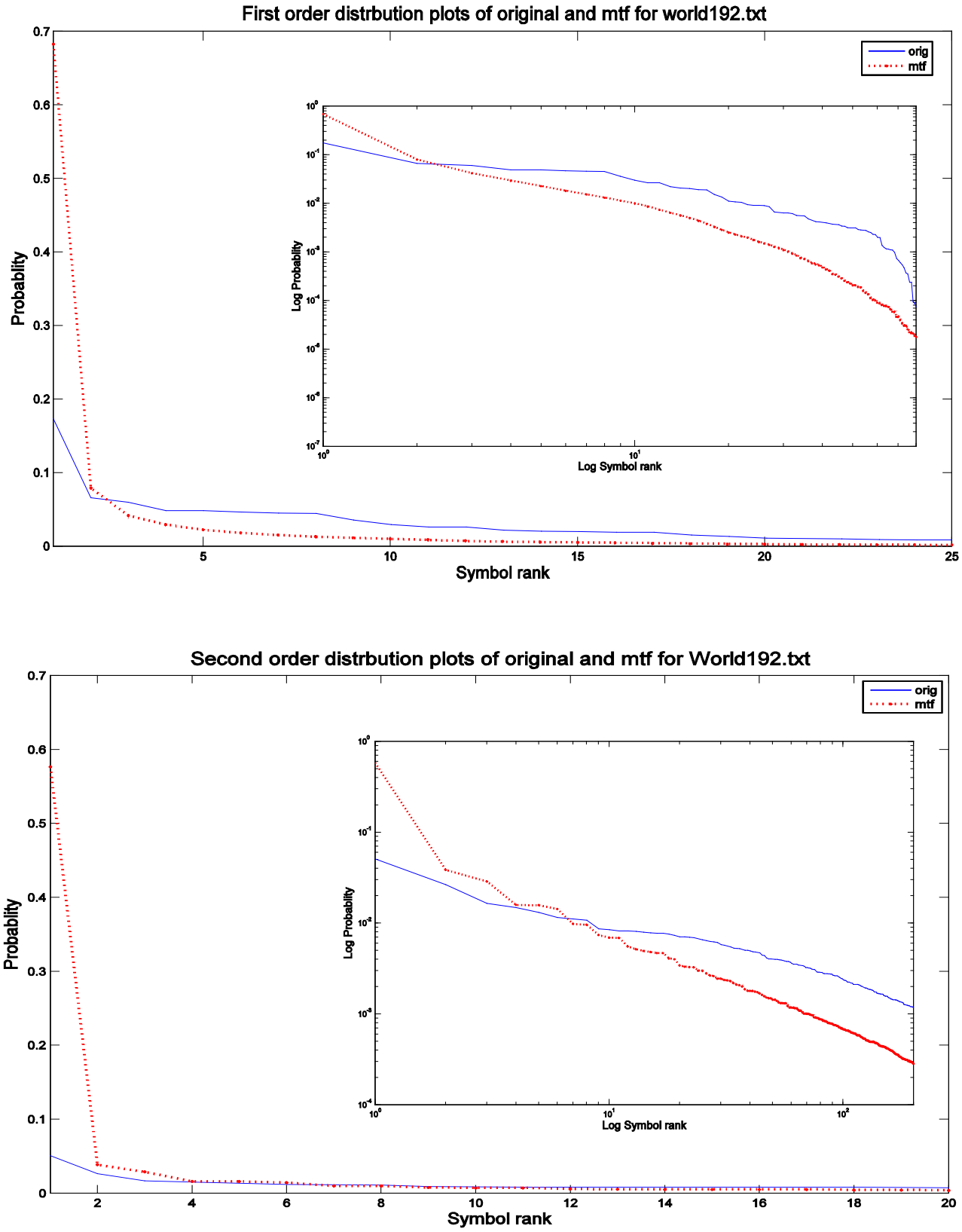


Figure 3.8: First and second order distribution plots of different predictors for World192.txt.

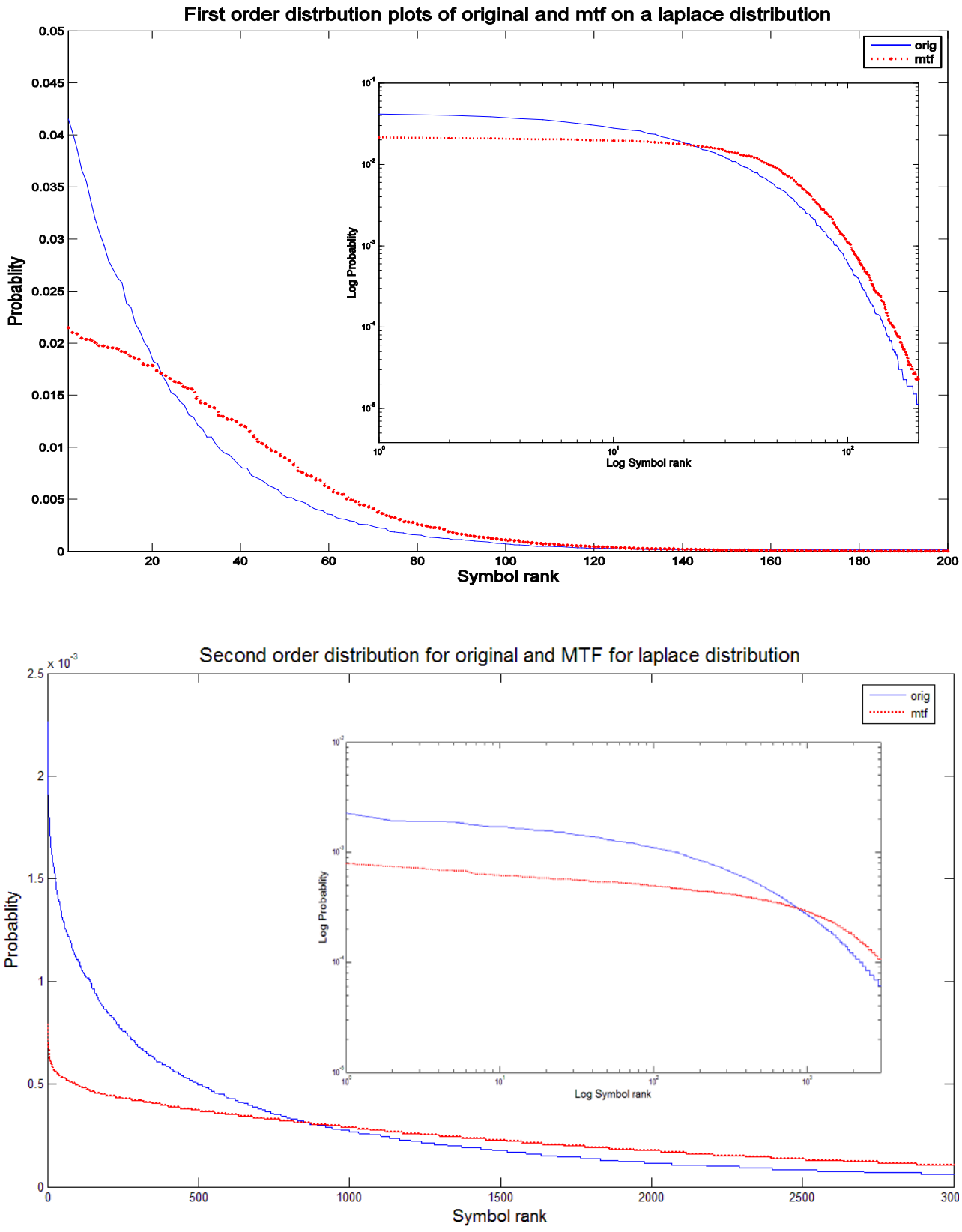


Figure 3.9: First and second order distribution plots of original and MTF for a Laplacian Distribution.

3.1.5 Characterizing the Effect of MTF on Image Compression

3.1.5.1 Effect of Range (*k*) in MTF on Image Compression.

The value of *k* has been varied from 1-255 for MHD, TRANS, Modified MHD and Modified TRANS, the average of the compression ratios for 9 images (i.e. BOAT, CAR, COUPLE, HOUSE, MAN, PARROT, TREE, WATERWHEEL and ZELDA) have been calculated and plotted as shown in figure 3.10. We can also observe the slight improvement in compression provided by the proposed modifications. As the value of the range parameter *k* increases towards the maximum of 256, it means that we are increasingly delaying the application of MTF stage. At *k* = 256, this effectively means that the MTF stage is no longer being applied. Therefore, it is very instructive to observe that each of the four variations produced their best result at *k* = 256, i.e effectively without MTF. We carry this observation even further, by investigating the performance of BWT based image compression with or without MTF, and with or without RLE for each case.

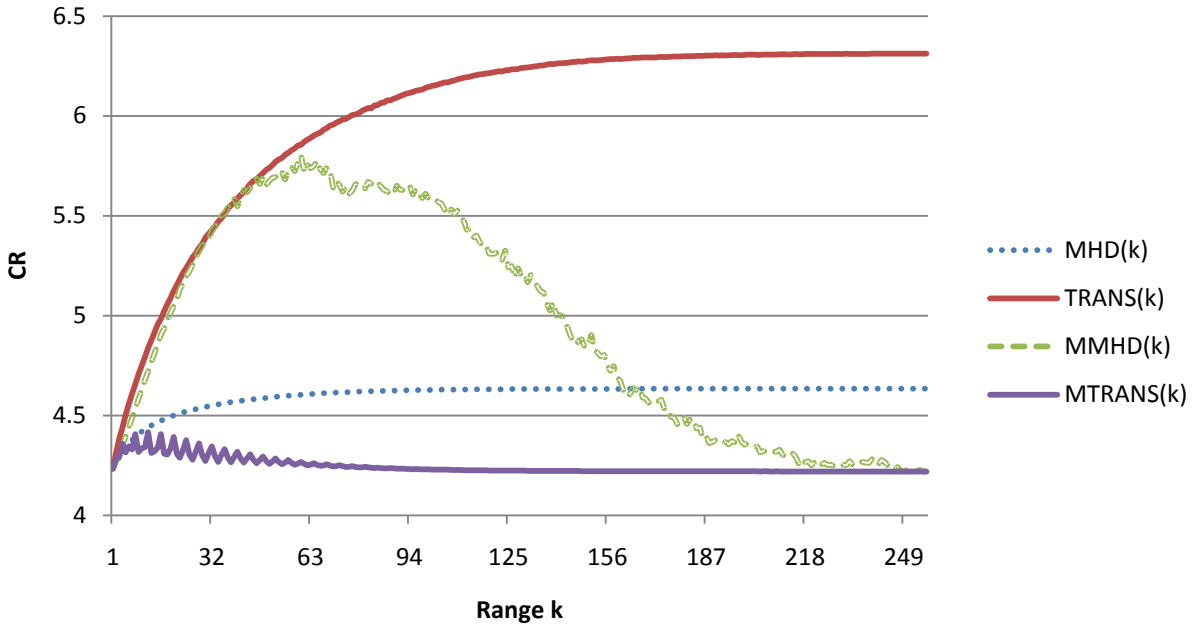


Figure 3.10: Variation of CR for MHD, TRANS, MMHD and MTRANS with range *k*.

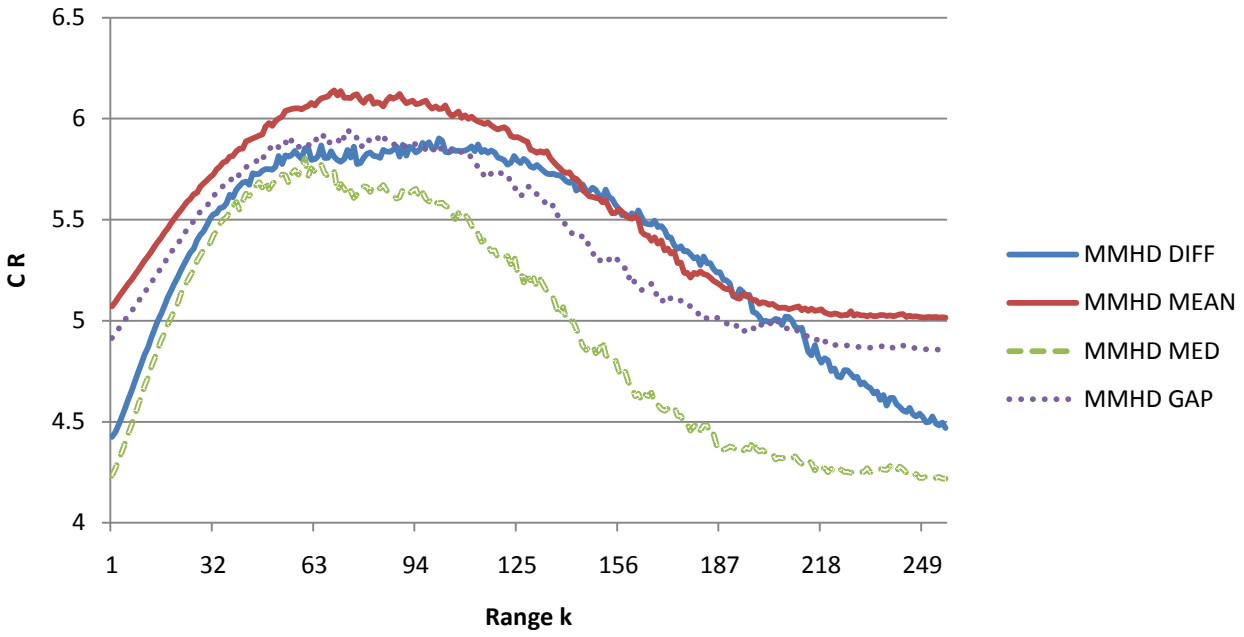


Figure 3.11: Variation of CR for MMHD with range *k* for different predictors.

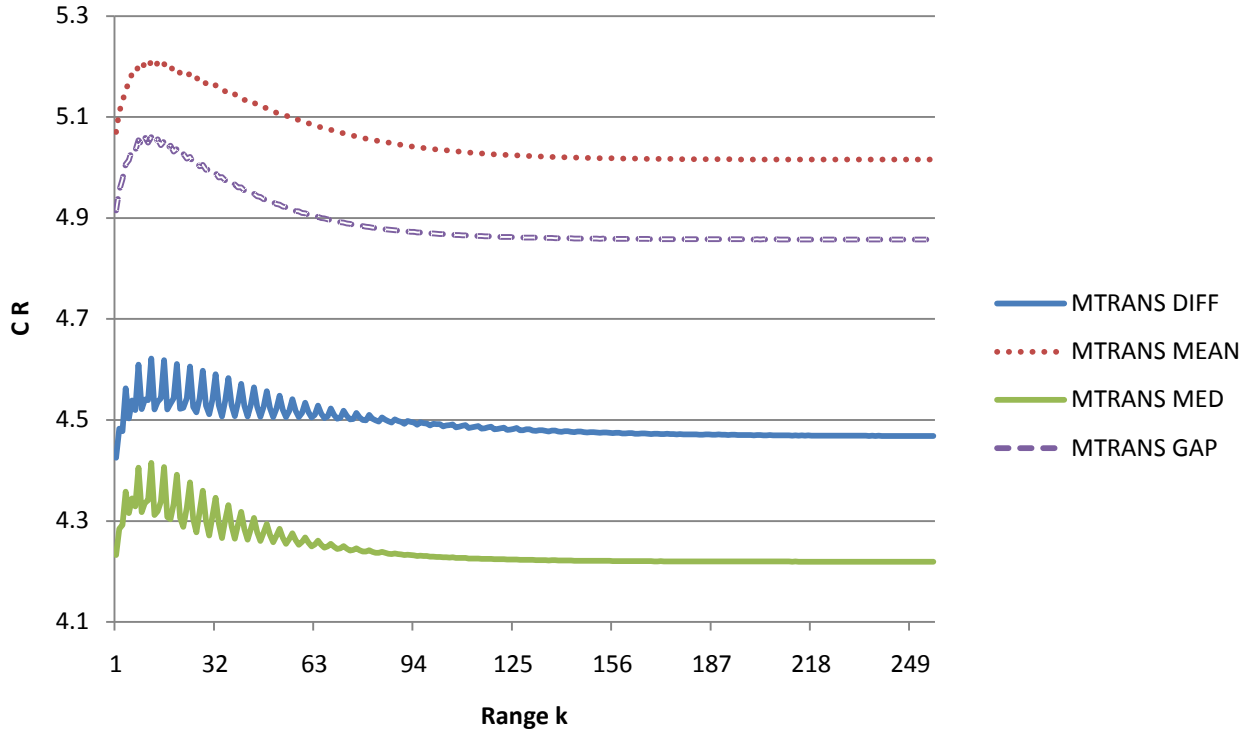


Figure 3.12: Variation of CR for MTRANS with range *k* for different predictors.

3.1.6 Effect of range (*r*) and window size (*w*) in MTFW on Image Compression

Figure 3.13 shows how the compression ratio is affected both by range *r* and the window size *w* in MTFW. We can observe the periodic curves with an interval of 30, for each value of range *r* the window size *w* was varied from 1-30. There are 16 such curves as the range *r* value is varied from 1-16. The compression ratio improves as the value of range *r* is decreased and the value of window size *w* is increased. The variation of compression ratio with window size *w* also reduces as the value of range *r* increases. The CR calculated here is the average of the nine images referred in the previous section and also MED predictor has been used.

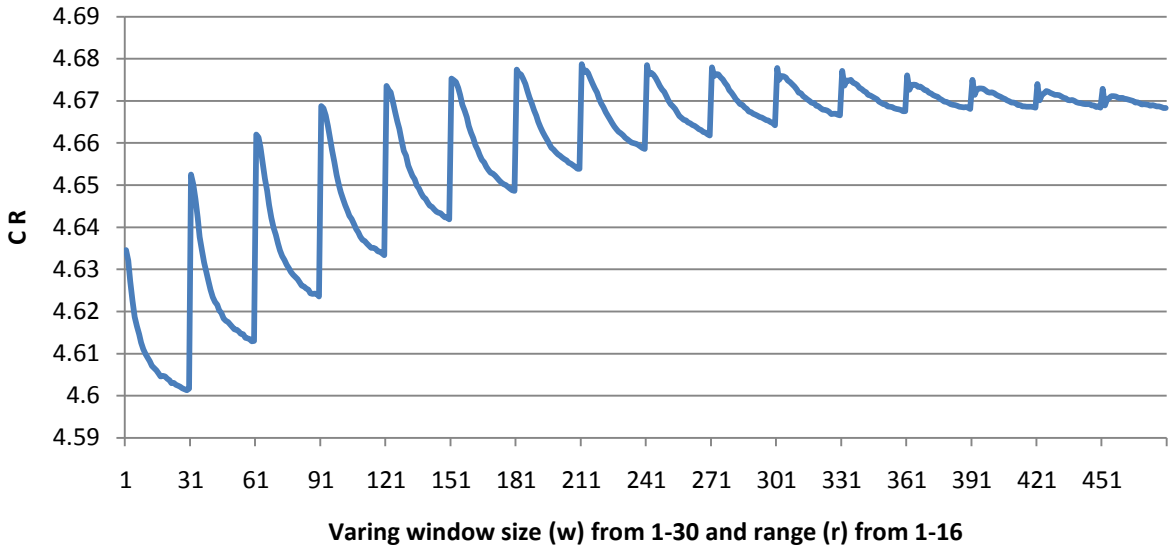


Figure 3.13: Variation of CR for MTFW(*w*,*r*) with window size *w* =1-30 and *r* =1-16.

Figure 3.14 shows the variation of CR in MTFW(*w*,*r*) for three values of range *r* = 1,2,3 and window size *w* varying from 1-100. As seen from the figure the compression ratio tends to be small as *w* increases and *r* decreases.

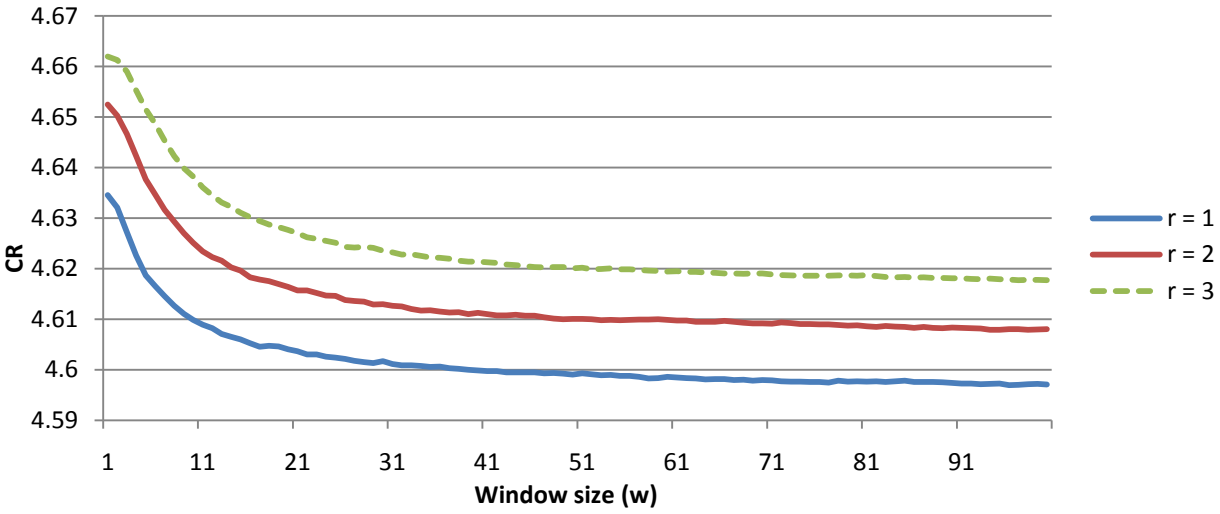


Figure 3.14: Variation of CR for MTFW(*w*,*r*) with window size *w* =1-100 and *r* =1,2,3.

Surface plot for MTFW

The plot above is redrawn using a surface plot, to give a 3-dimentional perspective. Figure 3.15 shows how CR (average from 9 images) is affected by both the range *r* and the window size *w* in MTFW. The CR improves with decreasing *r*, or increasing *w*.

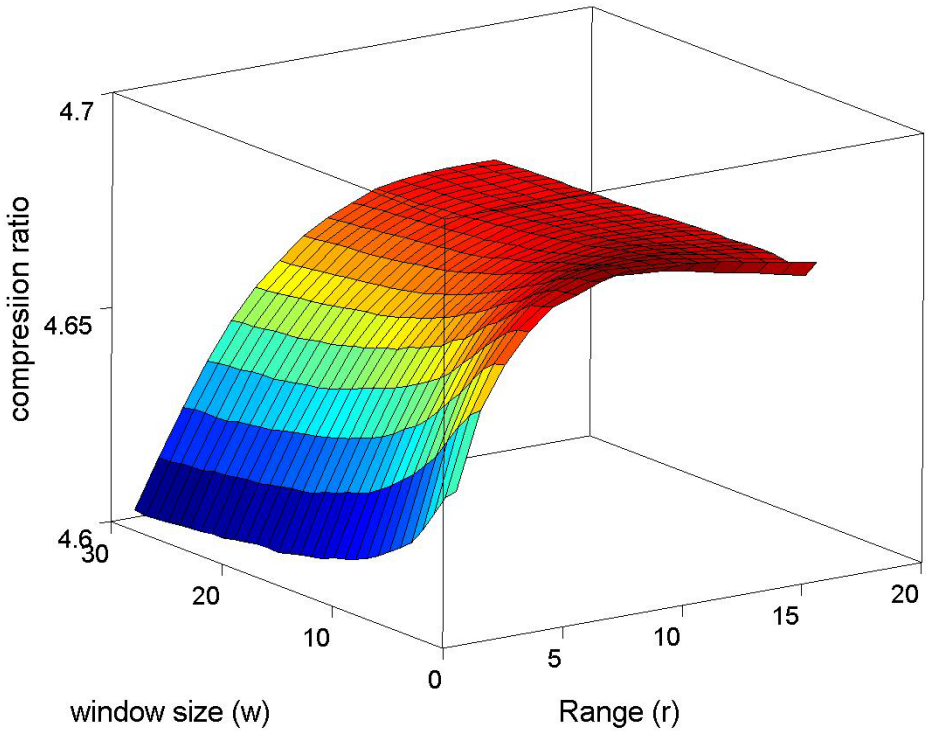


Figure 3.15: 3D plot for Variation of CR for MTFW(*w*,*r*) with window size *w* =1-30 and *r* =1-16.

3.1.7 Effect of range (*r*) and window size (*w*) in MTFW2 on Image Compression

Figure 3.16 shows the variation of CR in MTFW2(*w*,*r*) for three values of range *r* = 1,2,3 and window size *w* varying from 1-100. As seen from the figure the compression ratio tends to be small as w increases and *r* decreases but with some oscillations. The best compression is observed for *r* = 1 and *w* = 91.

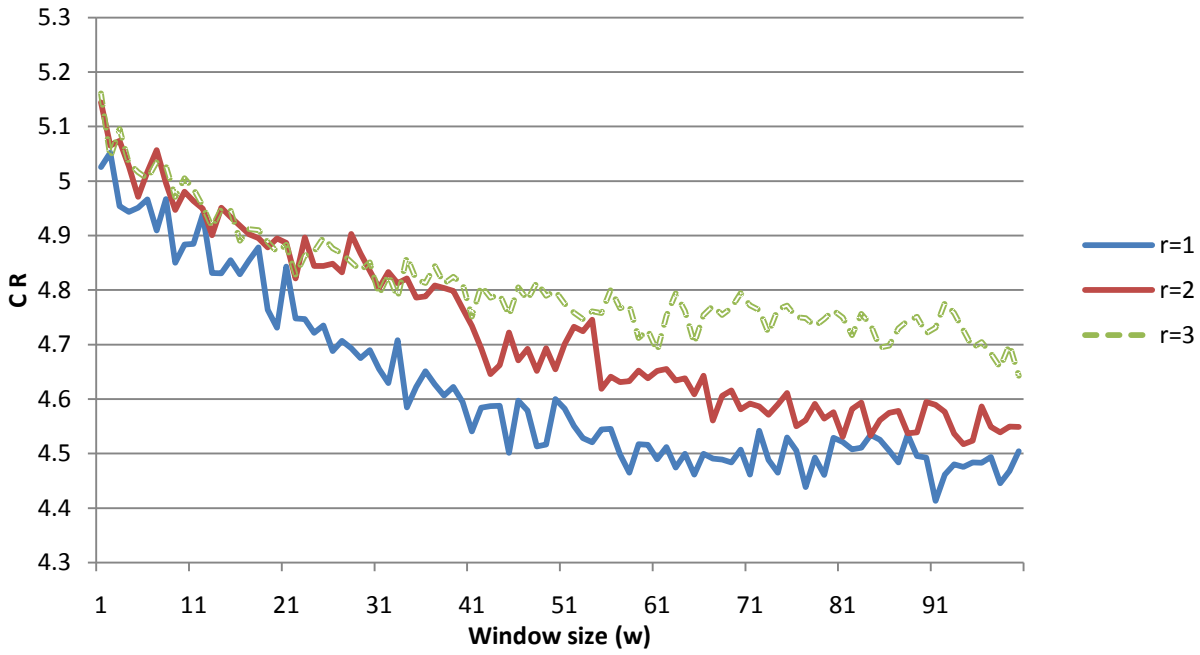


Figure 3.16 Variation of CR for MTFW2(w,r) with window size w =1-100 and r =1,2,3.

3.2 Context Partitions for BWT Image Compression

Context-based lossless image compression algorithms exploit the two-dimensional spatial redundancy in natural images. They represent the most successful among lossless image coders. Examples algorithms in this group are CALIC [13], JPEG-LS[11] and PPAM[9]. For this class of algorithms, an initial step of spatial prediction is used to remove the spatial redundancy in the image. The prediction depends on a chosen context selection strategy for each given position in the image. Context modeling is then applied to estimate the conditional probability distribution of the prediction residues given their contexts. Finally, an entropy coder uses the estimated conditional probabilities to encode the prediction residues. The different lossless image coders vary in the details of one or more of the above basic steps.

Consider an image represented by a sequence $T = t_1 t_2 \cdots t_n$, with symbols taken from a fixed alphabet $\Sigma = \sigma_1, \sigma_2, \cdots, \sigma_{|\Sigma|}$, where $n = |T|$ is the image size. Here, Σ is typically the set of distinct pixel gray levels in the image, or the set of distinct prediction errors, after prediction. Let the corresponding probability of the symbols in the image be $p(\sigma_i), i = 1,...,|\Sigma|$, $\sum_i p(\sigma_i) = 1$. The entropy $H(T)$ gives the minimum number of bits per symbol required to encode the image without context modeling:

$$H(T) = -\sum_{\sigma_i \in \Sigma} p(\sigma_i) \log_2 p(\sigma_i).$$

Now, consider the contexts for each symbol t_i in the image. Let S'_j be the set of symbols with the context C_j in *T*. Suppose we know the conditional probability distribution $p(t_i \mid C_j), i = 1,...,|S'_j|$. In this case, $H(T \mid C)$ the conditional entropy gives the minimum number of bits per symbol needed to encode the image:

$$H(T \mid C) = -\sum_{j=1}^{K} \left(p(C_j) \sum_{s_i \in S_j'} p(t_i \mid C_j) \log_2 \left(p(t_i \mid C_j) \right) \right) = -\sum_j \sum_{s_i} p(t_i, C_j) \log_2 \left(p(t_i \mid C_j) \right)$$

Where K is the total number of contexts. Since $H(T \mid C) \leq H(T)$, (see [29]), the average code length (per symbol) needed to describe the image is also reduced using context modeling. This is significant, as it provides an important connection between image compression and the contexts induced by the BWT. While traditional image compression schemes use preceding (reverse) pixel contexts, the BWT uses succeeding (forward) contexts. We will exploit the sorted contexts induced by the BWT in our approach to image compression.

An important property of the BWT is the introduction of sorted order on nearby contexts in the output string. Given the similarity of nearby symbols in the image (even after spatial prediction), we tried to use context partitions on the image sequence to see it this could lead to improved results. The general procedure is shown in Figure 3.17.

The BWT output string is divided into blocks which are determined by sorting the output string in alphabetic order. These individual blocks are processed by MTF, RLE and ARI individually and the outputs are combined to form the final compressed data. All these individual files are reused while decoding the original image. Let's see how this works with an example, if the output BWT sequence is ASDFSADSFSASFGHFDS, the sorted sequence will be AAADDDFFFFGHSSSSSS, so the output BWT sequence is broken into blocks ASD, FSA, DSFS, A, S, FGHFDS. These blocks are then sent individually to MTF, RLE and ARI blocks.

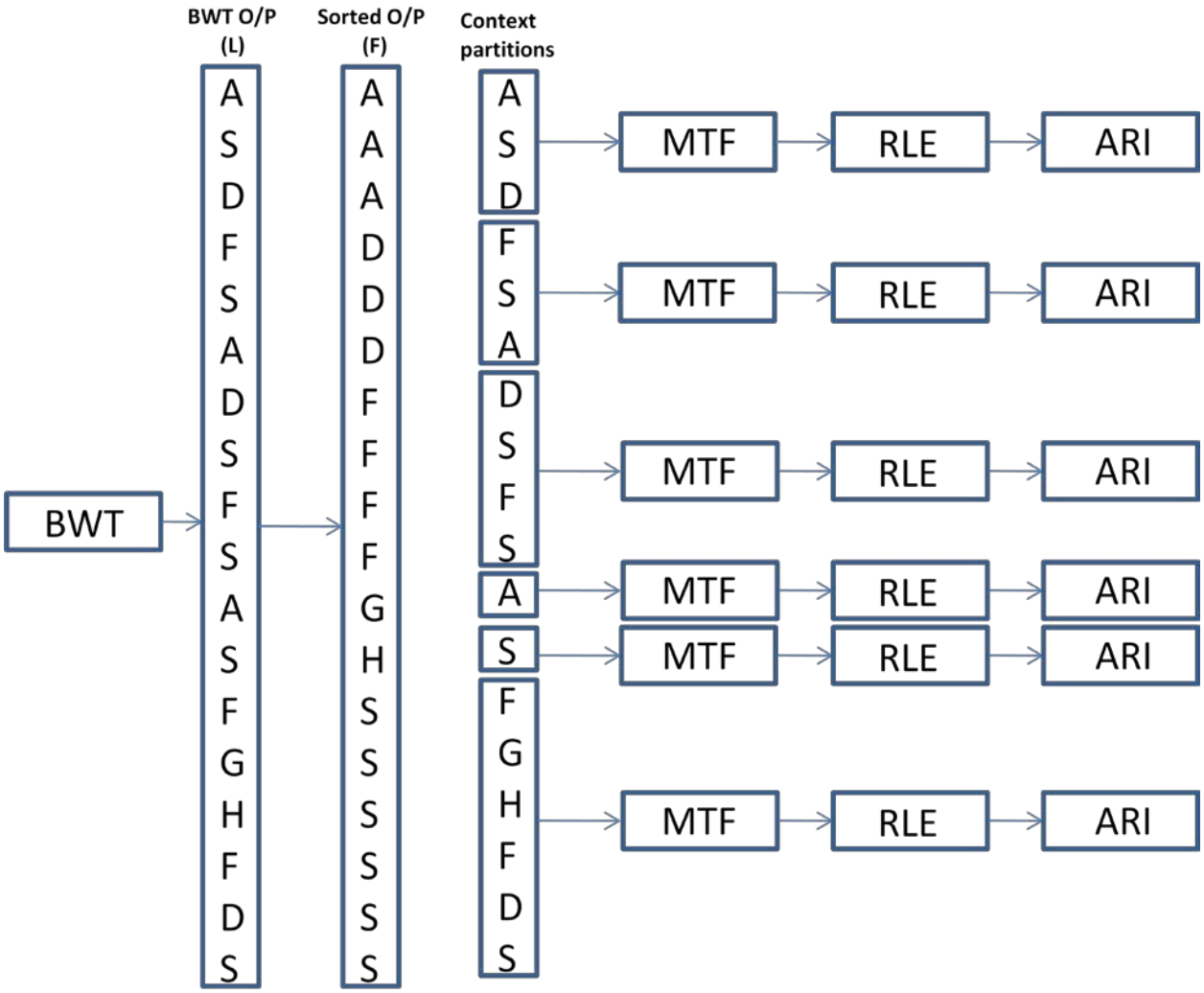


Figure 3.17: BWT compression with context partitions.

Guided by the foregoing, we obtain two variations for the proposed BWT-based image compression scheme. We call the first variation **BLIC** (BWT-based lossless image coder), which corresponds to the BWT compression pipeline but without MTF and without RLE. The second method is **BLICX** (BWT-based lossless image coder with context partitions). Thus, **BLICX** corresponds to BWT compression using context partitions, but without MTF and RLE.

4. Experimental results

4.1 Experimental Data and Experimental Environment

4.1.1 Data Sets

The following images sets have been used

- Standard images- SET-1

  1.BRIDGE, 2.AIRPLANE, 3.BABOON, 4.BALLON, 5.BARB, 6.CAMERA, 7.COUPLE, 8.GOLDHILL, 9.LENA, 10.PEPPERS, 11.SHAPES are used as first image set.

- Standard images- SET-2

  1.BOAT, 2.CAR, 3.COUPLE, 4.HOUSE, 5.MAN, 6.PARROT, 7.PLANT, 8.TREE, 9.WATERWHEEL, 10.ZELDA is the second image set.

- NEURITE SET

  1.NEURITE01, 2. NEURITE02, 3. NEURITE04, 4. NEURITE05, 5. NEURITE06, 6. NEURITE07, 7. NEURITE09, 8. NEURITE10, 9. NEURITE11, 10. NEURITE14 is the third image set.

- SLICE SET

  1.SLICE00, 2. SLICE01, 3. SLICE02, 4. SLICE03, 5. SLICE04, 6. SLICE05, 7. SLICE06, 8. SLICE07, 9. SLICE08, 10. SLICE09 is the fourth image set.

- RETINAL SET

  1.RETINAL00, 2.RETINAL03, 3.RETINAL04, 4.RETINAL09, 5.RETINAL11, 6.RETINAL12, 7.RETINAL14, 8.RETINAL15, 9.RETINAL17, 10.RETINAL18 is the fifth image set.

These sets have been used in the recently published work on lossless image compression [Zhang [9]]. The images are shown for reference in appendix A.

4.1.2 Experimental Environment

The experiments were carried out using MATLAB 7 on a system having Pentium core 2 duo processor, running at 1.83 GHz with 3GB RAM. We used MATLAB for image analysis and spatial prediction. The source codes for BWT the compression algorithms are based on Mark Nelson's codes [2], and are compiled using VC++ in Windows XP.

4.2 Results for different variations of MTF.

4.2.1 BWT Without mapping and without prediction

Table 4.1 shows the compression ratios for different variations of the MTF without the use of any mapping or any prediction. These results are obtained by applying the basic BWT which is famous for text compression. As debated it doesn't have the best performance in image compression. Observing this we have tried different variations like mapping and prediction which comes under preprocessing the image data to see whether it might improve the image compression capability of the BWT algorithm.

Table 4.1: BWT compression with different variations of MTF (without any mapping or prediction).

	MTF	TRANS	MTFF	MTFF2	MTRANS (8)	MTRANS (128)	MTFW (2,1)	MTFW (10,1)	MTFW2 (84,1)	MTFW2 (87,1)
BRIDGE	6.899	6.445	6.831	6.890	6.783	6.678	6.901	6.881	5.612	5.677
AIRPLANE	5.871	5.499	5.823	5.866	5.561	5.396	5.882	5.869	5.553	5.553
BABOON	7.847	7.461	7.833	7.846	7.592	7.443	7.848	7.834	6.683	6.683
BALLON	4.475	4.287	4.434	4.472	4.197	4.148	4.496	4.497	5.642	5.642
BARB	6.068	5.854	6.044	6.068	5.899	5.781	6.087	6.072	6.576	6.576
CAMERA	6.419	5.989	6.225	6.248	6.030	5.887	6.427	6.423	6.625	6.628
COUPLE	5.791	5.406	6.225	6.248	5.474	5.281	5.804	5.793	5.720	5.685
GOLDHILL	6.166	5.842	6.225	6.248	6.023	5.798	6.180	6.159	5.942	5.942
LENA	6.198	5.878	6.225	6.248	6.046	5.820	6.223	6.198	6.374	6.374
PEPPERS	6.250	5.903	6.225	6.248	6.067	5.840	6.268	6.247	6.249	6.249
SHAPES	1.943	1.878	1.898	1.910	1.910	1.937	1.914	1.923	2.295	2.295
AVERAGE	**5.812**	**5.495**	**5.817**	**5.845**	**5.598**	**5.455**	**5.821**	**5.809**	**5.752**	**5.755**

4.2.2 Performance Analysis for Different Predictors

Prediction using the neighboring pixels had a significant effect on the performance of BWT based image compression algorithm. We used different prediction schemes such as DIFF, GAP, MED and MEAN. The value underlined in each row indicates the lowest for a particular image. In each of these cases we have used the 8-bit mapping.

4.2.2.1 DIFF Predictor

Table 4.3 shows the compression ratios for different variations of MTF using 8-bit mapping and DIFF predictor. Here 8-bit mapping method which allows for the representation of the linear sequence, by eliminating the need for the extra sign bit has been used. This resulted in a 1-bit gain for each symbol. That's the reason why we observe a better compression when compared to the above other two techniques.

Table 4.2: BWT compression with different variations of MTF using 8-bit mapping and DIFF predictor.

	MTF	TRANS	MTFF	MTFF2	MTRANS (8)	MTRANS (128)	MTFW (2,1)	MTFW (10,1)	MTFW2 (84,1)	MTFW2 (87,1)
BRIDGE	5.977	5.704	5.947	5.974	5.684	5.618	5.974	5.964	7.632	7.768
AIRPLANE	5.149	4.765	5.115	5.148	4.889	4.711	5.149	5.130	6.563	6.585
BABOON	7.053	6.686	7.046	7.054	6.771	6.790	7.052	7.045	8.033	7.616
BALLON	3.796	3.566	3.760	3.794	3.624	3.469	3.810	3.796	4.604	4.604
BARB	5.451	5.299	5.438	5.451	5.432	5.271	5.460	5.446	6.695	6.651
CAMERA	5.626	5.177	5.589	5.617	5.326	5.177	5.621	5.605	6.363	6.822
COUPLE	4.993	4.596	4.948	4.983	4.705	4.516	4.987	4.971	5.978	6.252
GOLDHILL	5.565	5.219	5.551	5.564	5.364	5.185	5.566	5.554	6.709	6.686
LENA	5.478	5.138	5.463	5.478	5.307	5.121	5.483	5.469	6.434	6.326
PEPPERS	5.494	5.138	5.481	5.492	5.331	5.167	5.498	5.488	6.462	6.577
SHAPES	1.457	1.423	1.408	1.421	1.566	1.655	1.421	1.427	2.103	2.170
AVERAGE	**5.094**	**4.792**	**5.068**	**5.089**	**4.909**	**4.789**	**5.093**	**5.081**	**6.143**	**6.187**

4.2.2.2 Mean Predictor

Table 4.4 shows the compression ratios for different variations of MTF using 8-bit mapping and mean predictor. Here the mean value of the preceding pixel and the above pixel has been used as the predicted value and the difference between the predicted and original pixel value has been mapped and transmitted.

Table 4.3: BWT compression with different variations of MTF using 8-bit mapping and MEAN predictor.

	MTF	TRANS	MTFF	MTFF2	MTRANS (8)	MTRANS (128)	MTFW (2,1)	MTFW (10,1)	MTFW2 (84,1)	MTFW2 (87,1)
BRIDGE	6.719	6.255	6.698	6.719	6.324	6.253	6.713	6.701	6.220	6.220
AIRPLANE	4.896	4.507	4.860	4.896	4.616	4.415	4.897	4.880	4.883	4.765
BABOON	6.678	6.319	6.669	6.678	6.421	6.353	6.676	6.669	6.300	6.300
BALLON	3.731	3.505	3.698	3.729	3.548	3.413	3.749	3.736	3.872	4.002
BARB	5.702	5.408	5.688	5.703	5.534	5.375	5.709	5.697	5.813	5.732
CAMERA	5.513	5.047	5.476	5.508	5.163	5.018	5.506	5.495	5.510	5.509
COUPLE	4.934	4.586	4.899	4.928	4.701	4.509	4.934	4.919	5.129	4.949
GOLDHILL	5.294	4.956	5.277	5.294	5.108	4.905	5.298	5.283	5.381	5.067
LENA	5.402	5.054	5.386	5.403	5.229	5.033	5.407	5.395	5.122	5.141
PEPPERS	5.325	4.974	5.308	5.324	5.189	5.002	5.330	5.318	4.996	4.996
SHAPES	1.702	1.707	1.649	1.667	1.796	1.866	1.666	1.672	2.156	2.075
AVERAGE	**5.081**	**4.756**	**5.055**	**5.077**	**4.875**	**4.740**	**5.080**	**5.069**	**5.035**	**4.978**

4.2.2.3 MED Predictor

Table 4.5 shows the compression ratios for different variations of MTF using 8-bit mapping and MED predictor. The median predictor achieved the best overall compression ratio when compared with the other predictors tested. MED is the spatial predictor used in the JPEG-LS standard.

Table 4.4: BWT compression with different variations of MTF using 8-bit mapping and MED predictor.

	MTF	TRANS	MTFF	MTFF2	MTRANS (8)	MTRANS (128)	MTFW (2,1)	MTFW (10,1)	MTFW2 (84,1)	MTFW2 (87,1)
BRIDGE	5.927	5.487	5.878	5.923	5.490	5.413	5.923	5.898	5.391	5.391
AIRPLANE	4.775	4.399	4.741	4.771	4.526	4.310	4.778	4.760	4.766	4.794
BABOON	6.710	6.339	6.704	6.709	6.441	6.384	6.710	6.705	6.323	6.323
BALLON	3.678	3.424	3.642	3.678	3.467	3.309	3.696	3.680	3.699	3.726
BARB	5.541	5.214	5.522	5.540	5.353	5.172	5.544	5.532	5.291	5.570
CAMERA	5.417	4.971	5.381	5.417	5.106	4.933	5.414	5.398	5.408	5.094
COUPLE	4.612	4.257	4.570	4.606	4.372	4.159	4.610	4.591	4.479	4.661
GOLDHILL	5.188	4.854	5.172	5.187	5.020	4.809	5.192	5.178	4.901	4.821
LENA	5.402	5.069	5.387	5.402	5.237	5.039	5.407	5.393	5.459	5.212
PEPPERS	5.364	5.070	5.353	5.363	5.280	5.112	5.373	5.363	5.110	5.110
SHAPES	1.448	1.363	1.388	1.403	1.351	1.358	1.415	1.423	1.619	1.681
AVERAGE	**4.915**	**4.586**	**4.885**	**4.909**	**4.695**	**4.545**	**4.915**	**4.902**	**4.768**	**4.762**

4.2.2.4 GAP Predictor

Table 4.5 shows the compression ratios for different variations of MTF using 8-bit mapping and GAP predictor. Gradient-Adjusted Prediction is the prediction algorithm which is used in CALIC. The GAP predictor was better than the MEAN predictor, but was not as effective as the MED predictor.

Table 4.5: BWT compression with different variations of MTF using 8-bit mapping and GAP predictor.

	MTF	TRANS	MTFF	MTFF2	MTRANS (8)	MTRANS (128)	MTFW (2,1)	MTFW (10,1)	MTFW2 (84,1)	MTFW2 (87,1)
BRIDGE	6.943	6.543	6.932	6.944	6.636	6.588	6.938	6.932	6.524	6.524
AIRPLANE	4.746	4.375	4.710	4.744	4.488	4.278	4.748	4.731	4.901	4.931
BABOON	6.670	6.288	6.661	6.670	6.390	6.329	6.670	6.661	6.267	6.267
BALLON	3.585	3.346	3.546	3.582	3.361	3.229	3.601	3.586	3.719	3.598
BARB	5.416	5.126	5.399	5.415	5.263	5.090	5.422	5.410	5.450	5.371
CAMERA	5.380	4.933	5.342	5.376	5.044	4.871	5.372	5.352	5.340	5.320
COUPLE	4.656	4.308	4.616	4.650	4.428	4.226	4.661	4.637	4.719	4.740
GOLDHILL	5.173	4.837	5.156	5.172	4.997	4.788	5.175	5.160	5.034	5.058
LENA	5.265	4.940	5.251	5.264	5.099	4.893	5.272	5.256	5.058	5.058
PEPPERS	5.274	4.954	5.259	5.275	5.149	4.954	5.281	5.269	4.952	4.952
SHAPES	1.725	1.636	1.660	1.685	1.743	1.749	1.686	1.697	1.923	1.897
AVERAGE	**4.985**	**4.662**	**4.957**	**4.980**	**4.782**	**4.636**	**4.984**	**4.972**	**4.899**	**4.883**

Comments

Overall the MTRANS and its variants provided the best results. The overall best of the MTF variants was MTRANS(*k*) with *k*=128. Similar results were observed with higher values of *k*. See figure 3.11. The other proposed variants were generally better than the traditional MTF (Column 2), but, in general, did not do better than MTRANS(128). With respect to spatial predictors, the MED seemed to provide the best results when used as the preprocessor before the BWT compression pipeline.

Figures 3.10 and 3.11, shows the variation of the performance for various values of *k*, and for different prediction schemes. Once again, these results show that, when the objective is image compression using the BWT, better performance could be produced by eliminating the stage of MTF. Following these results, we also investigated the impact of the RLE stage (without MTF) in BWT-based compression.

4.3 Compression with and without MTF/RLE

The results in Table 4.6 show the effect of removal of both MTF and RLE from the BWT pipeline scheme in lossless image compression for different types of predictors. It has been observed that the use of MED predictor and the removal of both the MTF and RLE have resulted in the best CR. We have proposed a new compression scheme BLIC which corresponds to BWT compression pipeline without MTF, RLE or context partitions.

Table 4.6: Compression with and without MTF/RLE using different predictors.

IMAGE	Without predictor			MEAN			MED			GAP		
	BWT	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**	**BWT**	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**	**BWT**	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**	**BWT**	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**
AIRPLANE	5.533	5.552	4.874	4.415	4.411	4.254	4.310	4.308	4.149	4.278	4.278	4.118
BABOON	6.589	6.683	6.953	6.353	6.299	6.208	6.384	6.322	6.230	6.329	6.267	6.175
BALLOON	5.558	5.642	3.548	3.413	3.414	3.233	3.309	3.309	3.127	3.229	3.230	3.051
BARB	6.513	6.576	5.293	5.375	5.364	5.230	5.172	5.164	5.025	5.090	5.083	4.941
GOLDHILL	5.888	5.942	5.272	4.905	4.905	4.804	4.809	4.809	4.702	4.788	4.787	4.679
LENA	6.283	6.374	5.318	5.033	5.026	4.871	5.039	5.035	4.891	4.893	4.890	4.741
LENNAGREY	6.125	6.205	4.969	4.718	4.714	4.548	4.693	4.692	4.533	4.544	4.543	4.382
PEPPERS	6.187	6.249	5.323	5.002	4.995	4.835	5.112	5.110	4.949	4.954	4.951	4.792
REF12B	2.026	2.027	1.276	1.196	1.196	0.925	1.163	1.162	0.852	1.173	1.173	0.884
SHAPES	2.209	2.295	1.959	1.866	1.862	2.066	1.358	1.352	1.263	1.749	1.745	1.786
AVERAGE	**5.291**	**5.354**	**4.478**	**4.227**	**4.218**	**4.097**	**4.134**	**4.126**	**3.972**	**4.102**	**4.094**	**3.954**

4.4 Compression for context partitions with and without MTF/RLE

Also the effect of removal of both MTF and RLE from the context partitions based BWT pipeline scheme in lossless image compression for different types of predictors has been tested. The results have been shown in the table 4.7. It has been observed that without the use of any predictor and the removal of both the MTF and RLE have resulted in the best CR. We have proposed a new compression scheme BLICx which corresponds to BWT compression pipeline without MTF, RLE and context partitions.

Table 4.7: Compression for context partitions with and without MTF/RLE using different predictors.

IMAGE	Without predictor			MEAN			MED			GAP		
	BWT	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**	**BWT**	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**	**BWT**	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**	**BWT**	**BWT_{nomtf}**	**$BWT_{nomtf,norle}$**
AIRPLANE	0.155	0.155	4.323	4.398	4.395	4.238	4.398	4.267	4.109	4.334	4.332	4.167
BABOON	5.837	5.773	6.482	6.543	6.508	6.414	6.543	6.551	6.456	6.535	6.496	6.401
BALLOON	4.640	4.635	2.293	3.290	3.290	3.107	3.290	3.065	2.894	3.071	3.071	2.896
BARB	6.179	6.133	4.815	5.454	5.448	5.304	5.454	5.263	5.114	5.146	5.142	4.988
GOLDHILL	5.470	5.452	4.725	4.992	4.991	4.869	4.992	4.898	4.767	4.883	4.882	4.750
LENA	4.966	4.937	4.706	5.168	5.161	5.002	5.168	5.200	5.052	5.002	5.001	4.842
LENNAGREY	4.641	4.602	4.286	4.870	4.867	4.693	4.870	4.838	4.671	4.686	4.685	4.512
PEPPERS	5.760	5.710	4.619	5.126	5.122	4.958	5.126	5.247	5.084	5.101	5.101	4.938
REF12B	1.762	1.762	1.098	0.565	0.565	0.494	0.565	0.525	0.426	0.510	0.510	0.426
SHAPES	1.703	1.706	1.718	1.930	1.927	2.072	1.930	1.438	1.350	1.792	1.790	1.777
AVERAGE	**4.111**	**4.087**	**3.907**	**4.234**	**4.227**	**4.115**	**4.234**	**4.129**	**3.992**	**4.106**	**4.101**	**3.970**

4.5 Comparitive Results with text Compression schemes

The performance of our new researched method BLIC and BLICx with and without MED predictor for lossless compression of images using the BWT algorithm was compared with other standard text compression algorithms like PPM, WinZip, Bzip, Gzip, RAR. Table 4.8, gives the compression performance of the different coding methods in bits per pixel.

Table 4.8: Comparative results with standard text compression methods.

Image	PPM	WinZip	Bzip	Gzip	RAR	Proposed methods			
						With MED predictor		Without Predictor	
						BLICx	BLIC	$BLICx_{nomed}$	$BLIC_{nomed}$
AIRPLANE	4.59	5.74	4.97	5.73	4.62	4.10	4.14	4.32	4.87
BABOON	6.52	7.26	6.74	7.26	6.61	6.45	6.23	6.48	6.95
BALLOON	3.9	5.53	4.28	5.52	3.77	2.89	3.12	2.29	3.55
BARB	6.13	7.1	6.52	7.09	6.19	5.11	5.02	4.81	5.29
GOLDHILL	5.46	6.64	5.79	6.63	5.08	4.76	4.70	4.72	5.27
LENA	5.56	7.16	5.84	7.15	5.43	5.05	4.89	4.70	5.32
LENNAGREY	5.23	6.82	5.55	6.82	5.10	4.67	4.53	4.28	4.97
PEPPERS	5.31	7.1	5.63	7.09	5.65	5.08	4.94	4.61	5.32
REF12B	0.86	1.32	1.15	1.25	1.15	0.42	0.85	1.09	1.28
SHAPES	1.25	1.48	1.42	1.43	1.35	1.35	1.26	1.71	1.96
AVERAGE	**4.48**	**5.61**	**4.79**	**5.6**	**4.5**	**3.99**	**3.97**	**3.90**	**4.48**

From the results shown in table above, we can see that our variation of BWT for lossless image compression performed better than all the other text compression algorithms that were used in the analysis. Also, the comparative results between bzip and our method show that the preprocessing of image data enhanced the performance of this BWT based compression method.

4.6 Comparitive Results with image Compression schemes

Comparison results have been observed for four image sets Natural, Neurite, Slice, and Retinal. Tables 4.9, 4.10, 4.11 and 4.12 show the comparative performance of variations of BWT on different image sets, when compared with state of the art lossless image compression schemes such as CALIC, JPEG-LS, EDP, JPEG2000, SPIHT and PPAM. BWT_{nomtf} correspond to BWT compression pipeline without MTF. BLIC corresponds to BWT compression pipeline without MTF, RLE and context partitions. BLICx corresponds to BWT compression pipeline without MTF and RLE with context partions. As seen from the table BLIC seems to give the best reults among the propossed methods. All the values except the once based on BWT have been obtained

from Zhang [9] paper for comparison. MED predicted data has been used in all the BWT based methods.

Table 4.9: Comparative compression performance on natural image set.

Image	Entropy	EDP	PPAM1	PPAM2	CALIC	JPEG LS	SPIHT	JPEG 2000	BWT	Proposed Methods				
										Without Context Partitions		Context Partitions		
										BWT_{nomtf}	BLIC	BWT	BWT_{nomtf}	BLICx
BOAT	6.132	4.358	4.33	4.01	4.181	4.271	4.37	4.465	4.065	4.064	3.803	4.187	4.187	3.921
CAR	6.251	4.196	4.27	3.797	3.946	4.068	4.236	4.285	3.924	3.923	3.677	4.061	4.062	3.797
COUPLE	6.359	4.772	4.735	4.497	4.618	4.698	4.829	4.903	4.378	4.375	4.151	4.532	4.532	4.305
HOUSE	5.638	5.272	5.29	5.044	4.983	5.138	5.396	5.435	4.717	4.711	4.557	4.887	4.883	4.719
MAN	6.37	5.038	4.985	4.891	4.808	4.928	5.058	5.218	4.741	4.728	4.536	4.921	4.910	4.710
PARROT	6.178	3.565	3.544	3.245	3.327	3.48	3.532	3.695	3.381	3.387	3.061	3.607	3.612	3.265
PLANT	5.118	5.129	5.040	5.224	5.108	5.176	5.161	5.343	5.610	5.607	5.483	6.114	6.117	5.983
TREE	5.535	5.441	5.373	5.141	5.141	5.271	5.362	5.520	4.868	4.865	4.695	5.057	5.055	4.879
WATERW	5.948	5.178	5.152	4.914	4.793	4.963	5.206	5.3	4.628	4.623	4.429	4.793	4.789	4.584
ZELDA	6.267	4.03	3.839	3.746	3.908	4.029	3.971	4.054	3.760	3.760	3.487	3.875	3.876	3.596
Average	**5.980**	**4.698**	**4.656**	**4.451**	**4.481**	**4.602**	**4.712**	**4.822**	**4.407**	**4.404**	**4.188**	**4.603**	**4.602**	**4.376**

Table 4.10: Comparative compression performance on neurite image set.

Image	Entropy	EDP	PPAM1	PPAM1 (function)	CALIC	JPEG LS	SPIHT	JPEG 2000	BWT	Proposed Methods				
										Without Context Partitions		Context Partitions		
										BWT_{nomtf}	BLIC	BWT	BWT_{nomtf}	BLICx
NEURITE01	3.394	3.259	2.817	2.975	2.932	3.101	2.997	3.118	3.773	2.971	2.605	4.076	3.165	2.771
NEURITE02	3.46	3.503	3.23	3.381	3.299	3.4	3.342	3.445	3.963	3.223	2.874	4.212	3.411	3.031
NEURITE04	3.05	2.946	2.594	2.728	2.561	2.839	2.792	2.889	3.512	2.811	2.426	3.786	2.986	2.575
NEURITE05	3.768	3.822	3.601	3.776	3.702	3.811	3.675	3.79	3.453	2.771	2.377	3.691	2.935	2.513
NEURITE06	3.299	3.21	2.835	3.054	2.95	3.106	2.994	3.114	4.294	3.558	3.249	4.653	3.792	3.461
NEURITE07	3.637	3.672	3.442	3.621	3.553	3.661	3.533	3.627	3.716	2.980	2.609	3.985	3.165	2.766
NEURITE09	3.158	3.104	2.658	2.809	2.77	3.008	2.94	3.035	4.184	3.391	3.061	4.488	3.601	3.242
NEURITE10	3.212	3.111	2.737	2.922	2.768	3.013	2.932	3.016	3.859	3.123	2.766	4.151	3.320	2.930
NEURITE11	3.583	3.407	3.291	3.45	3.326	3.542	3.481	3.56	3.471	2.736	2.363	3.572	2.816	2.438
NEURITE14	3.442	3.38	3.104	3.287	3.128	3.315	3.297	3.392	3.471	2.746	2.373	3.572	2.828	2.449
Average	**3.400**	**3.341**	**3.031**	**3.200**	**3.099**	**3.280**	**3.198**	**3.299**	**3.770**	**3.031**	**2.670**	**4.019**	**3.202**	**2.818**

From the tables we can observe that our proposed method BLIC has outperformed all the other lossless image compression schemes used for compression in Natural and Neurite image sets. It was comparable in the retinal image set case. It didn't do quite good in the slice image sets.

Table 4.11: Comparative compression performance on slice image set.

Image	Entropy	EDP	PPAM1	PPAM1 (function)	CALIC	JPEG LS	SPIHT	JPEG 2000	BWT	Proposed Methods				
										Without Context Partitions		Context Partitions		
										BWT_{nomtf}	BLIC	BWT	BWT_{nomtf}	BLICx
SLICE00	2.63	2.18	1.507	1.626	1.764	1.842	2.12	2.053	2.760	2.763	2.633	3.114	3.127	2.900
SLICE01	2.606	2.155	1.576	1.693	1.757	1.821	2.1	2.043	2.771	2.773	2.641	3.124	3.138	2.912
SLICE02	2.773	2.343	1.626	1.832	1.928	1.995	2.14	2.188	2.936	2.945	2.819	3.315	3.329	3.109
SLICE03	2.76	2.375	1.736	1.947	2.069	2.166	2.14	2.314	3.183	3.193	3.056	3.580	3.594	3.379
SLICE04	2.908	2.601	2.019	2.215	2.438	2.59	2.37	2.621	3.636	3.647	3.499	4.074	4.086	3.878
SLICE05	2.898	2.518	1.942	2.128	2.295	2.409	2.41	2.471	3.429	3.439	3.305	3.852	3.865	3.663
SLICE06	2.607	2.155	1.655	1.871	1.898	2.01	2.22	2.169	2.981	2.989	2.840	3.362	3.375	3.127
SLICE07	2.399	2.01	1.609	1.804	1.725	1.86	1.87	2.034	2.765	2.768	2.583	3.094	3.107	2.839
SLICE08	1.882	1.608	1.397	1.524	1.443	1.59	1.5	1.77	2.397	2.404	2.183	2.685	2.699	2.420
SLICE09	1.637	1.441	1.304	1.477	1.225	1.402	1.39	1.591	2.102	2.108	1.826	2.335	2.347	2.019
Average	**2.510**	**2.139**	**1.637**	**1.812**	**1.854**	**1.969**	**2.026**	**2.125**	**2.896**	**2.903**	**2.739**	**3.253**	**3.267**	**3.025**

Table 4.12: Comparative compression performance on retinal image set.

Image	Entropy	EDP	PPAM1	PPAM1 (function)	CALIC	JPEG LS	SPIHT	JPEG 2000	BWT	Proposed Methods				
										Without Context Partitions		Context Partitions		
										BWT_{nomtf}	BLIC	BWT	BWT_{nomtf}	BLICx
RETINAL00	3.444	3.522	3.318	3.412	3.362	3.463	3.46	3.498	3.823	3.826	3.634	3.981	3.988	3.785
RETINAL03	3.35	3.406	3.174	3.337	3.222	3.313	3.334	3.374	3.704	3.707	3.513	3.853	3.860	3.654
RETINAL04	3.246	3.329	3.135	3.209	3.162	3.235	3.26	3.282	3.615	3.618	3.422	3.753	3.760	3.551
RETINAL09	3.297	3.384	3.191	3.245	3.232	3.321	3.331	3.369	3.692	3.695	3.499	3.838	3.845	3.637
RETINAL11	3.352	3.445	3.244	3.36	3.288	3.398	3.381	3.419	3.753	3.757	3.561	3.903	3.910	3.702
RETINAL12	3.346	3.419	3.213	3.298	3.264	3.358	3.363	3.392	3.728	3.732	3.536	3.877	3.884	3.676
RETINAL14	3.395	3.476	3.251	3.346	3.31	3.413	3.417	3.456	3.793	3.797	3.602	3.947	3.954	3.748
RETINAL15	3.326	3.405	3.192	3.312	3.24	3.33	3.341	3.376	3.704	3.707	3.512	3.851	3.859	3.651
RETINAL17	3.254	3.325	3.118	3.205	3.152	3.223	3.247	3.28	3.610	3.614	3.418	3.753	3.760	3.551
RETINAL18	3.359	3.435	3.211	3.366	3.266	3.367	3.365	3.399	3.733	3.736	3.542	3.882	3.889	3.683
Average	**3.337**	**3.415**	**3.205**	**3.309**	**3.250**	**3.342**	**3.350**	**3.385**	**3.715**	**3.719**	**3.524**	**3.864**	**3.871**	**3.664**

4.7 Coding Time

The table 4.13 gives the coding time for various images with sizes 256x256 and 512x512 using the different proposed methods with MED predictor.

Table 4.13: Coding time.

Image	BWT	Proposed Methods				
		Without Context Partitions		Context Partitions		
		BWT_{-mtf}	BLIC	BWT	BWT_{-mtf}	BLICx
LENA (256x256)	4.330	6.081	3.327	13.769	19.931	10.621
CAMERA (256x256)	2.579	2.877	1.342	10.008	20.841	13.743
BRIDGE (256x256)	1.556	2.622	2.486	18.541	27.137	11.234
AIRPLANE (512x512)	5.755	4.994	4.821	11.568	23.158	11.564
BABOON (512x512)	5.251	5.424	4.974	13.724	22.412	13.386
PEPPERS (512x512)	5.614	8.148	4.834	14.326	29.340	10.092

5. Conclusion and Future work

5.1 Conclusion

Motivated by the results of a theoretical analysis of the BWT, we performed a detailed empirical investigation of the impact of the MTF stage in BWT-based lossless image compression. We proposed different parameterized variants of the MTF, which showed how the performance varied with these certain algorithmic. In general, these variants produced limited improvement in image compression. However, they showed the general impact of MTF on image compression using the BWT. Guided by our empirical and theoretical analyses, we propose to eliminate the MTF and RLE in the BWT pipeline, when the objective is lossless image compression. Also the usage of predictors before BWT pipeline has helped to improve the performance in the case of lossless image compression. Further, based on the context ordering property of the BWT, we proposed to use BWT context partitions as the basis for lossless image compression. We thus presented two BWT-based coders for images, namely BLIC and BLICx. Both use neither the MTF nor RLE stages in compressing the image. BLICx differs from BLIC only in the use of BWT context partitions. Empirical results on standard test images show that the both BLIC and BLICx outperformed current state-of-the-art lossless image coders, such as JPEG-LS, CALIC and PPAM. The results therefore show that, contrary to popular belief the BWT cannot compress images; the BWT can indeed deliver superior performance in image compression. The culprit has been the MTF, and to some extent the RLE stages. The results also show the power of the context modeling ability of the BWT for images, even without initial prediction. We have used simple MED for prediction, and order-1 arithmetic coding.

5.2 Future work

The time efficiency of the BWT compression Scheme can be improved. Different kinds of predictors can be tested to obtain a better data that may be more suitable for the BWT compression. The results could be further improved by using more powerful spatial predictors, and improved entropy coders. The parameters range *r* and window size *w* in MTFW(*w*,*r*), MTFW2(*w*,*r*) may be tuned adaptively based on the image to obtain a better compression ratio. Higher order contexts can be tested to see the performance of the context based method. The performance of BLIC and BLICx should be observed for large images as BWT tends to work better as the data size is large.

Reference

1. Burrows M and Wheeler D (1994), A block sorting lossless data compression algorithm, Technical Report 124, Digital Equipment Corporation.

2. Mark Nelson, [online] **http://marknelson.us/1996/09/01/bwt/**

3. M.R. Nelson, Data Compression with Burrows Wheeler Transformation, Dr. Dobb's Journal, pp. 46-50, September 1996.

4. The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching Adjeroh, Donald, Bell, Timothy, Mukherjee, Amar 2008, XII, 352 p. 102 illus.

5. J. L. Bentley, D. D. Sleator, R. E. Tarjan, V. K. Wei, A Locally Adaptive Data Compression Scheme, Communications of the ACM-Vol. 29, No. 4, 1986.

6. S. Irani, Two results on the list update problem, Information Processing Letters, v.38 n.6, p.301-306, June 28, 1991

7. D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. Communications of the ACM, 28(2):202-208, 1985.

8. X. Li and M. T. Orchard, "Edge-directed prediction for lossless compression of natural images," IEEE Trans. Image Process., vol. 10, no. 6, pp. 813–817, Jun. 2001.

9. Zhang, Y., Adjeroh, D.A., Prediction by Partial Approximate Matching for Lossless Image Compression, IP(17), No. 6, June 2008, pp. 924-935.

10. M. Boliek, "New work item proposal: JPEG2000 image coding system," ISO/IEC JTC1/SC 29/WG1 N390, June 1996.

11. M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," IEEE Trans. Image Process., vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

12. N. Memon and X.Wu, "Recent developments in context-based predictive techniques for lossless image compression," The Computer J., vol. 40, pp. 127–136, 1997.

13. X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," IEEE Trans. Commun., vol. 45, no. 4, pp. 437–444, Apr. 1997.

14. A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 3, pp. 243–250, Jun. 1998.

15. William B. Pennebaker and Joan L. Mitchell (1993). JPEG still image data compression standard (3rd ed.). Springer. p. 291. ISBN 9780442012724.

16. H. Witten, M. Neal and G. Cleary, Arithmetic Coding for data Compression, Communications of the ACM, vol. 30, no. 6, pp. 520-540, June 1987.

17. H. Murakami, S. Matsumoto, Y. Hatori, and H. Yamamoto, “15/30 Mbit/s universal digital TV codec using a median adaptive predictive coding method,” IEEE Trans. Commun., vol. 35, pp. 637–645, June 1987.

18. Wu, X., Memon, N. D. and Sayood, K. (1995) A contextbased, adaptive, lossless/nearly-lossless coding scheme for continuous-tone images. ISO Working Document ISO/IEC/ SC29/WG1/N256.

19. R. Rivest. On self-organizing sequential search heuristics. Communications of the ACM, 19, 2:63-67, February 1976.

20. Bernhard Balkenhol, Stefan Kurtz, Yuri M. Shtarkov. “Modifications of the Burrows and Wheeler Data Compression Algorithm, " Proc. Data Compression Conf., pp. 188-197, 1999.

21. Switching Between Two On-line List Update Algorithms for Higher Compression of Burrows-Wheeler Transformed Data," dcc, pp.183, Data Compression Conference (DCC '00), 2000

22. Bachrach, R. and El-Yaniv, R. (1997). Online list accessing algorithms and their applications: Recent empirical evidence. In Proc. Eighth Annual Symp. Discrete Algorithms (New Orleans, LA), ACM, New York, pp. 53--62.

23. Adjeroh, D. and Nan, F. (2008). Suffix sorting via Shannon-Fano-Elias codes. In DCC, page to appear. IEEE Computer Society.

24. Adjeroh, D., Zhang, Y., Mukherjee, A., Powell, M., and Bell, T. (2002). DNA sequence compression using the Burrows-Wheeler transform. In IEEE Com- puter Society Bioinformatics Conference, pages 303–313.

25. Fenwick, P. M. (1996b). The Burrows-Wheeler Transform for block sorting text compression: Principles and improvements. Computer Journal, 39(9):731–740.

26. Ferragina, P. and Manzini, G. (2001b). An experimental study of an opportunistic index. In 12th ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 269–278.

27. Elias, P. (1987). Interval and recency rank source coding: Two on-line adaptive variable-length schemes. IEEE Trans. Inf. Theory, IT-33(1):3–10.

28. Elias, P. (1975). Universal codeword sets and representations of the integers. IEEE Trans. Inf. Theory, 21(2):194–203.

29. Cover T. M. and Thomas J. A. (1991). Elements of Information Theory: Wiley-Interscience, NY.

30. Arimura, M. and Yamamoto, H. (1998). Asymptotic optimality of block sorting data compression algorithm. IEICE Trans. Fundamentals, E81-A(10):2117–2122.

31. Effros, M., Visweswariah, K., Kulkarni, S. R., and Verdu, S. (2002). Universal lossless source coding with the Burrows Wheeler transform. IEEE Transactions on Information Theory, 48(5):1061–1081.

32. Wirth, A. I. and Moffat, A. (2001). Can we do without ranks in Burrows Wheeler transform compression? In IEEE DCC, 2001, pp. 419–428.

33. Krichevsky, R. E. and Trofimov, V. K. (1981). The performance of universal encoding. IEEE Transactions on Information Theory, 27(2):199–206.

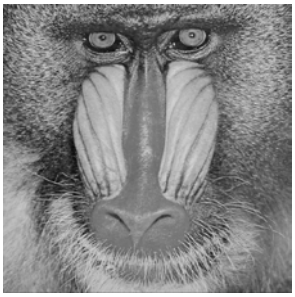
Appendix

1. Standard images- SET-1

Bridge

Airplane

Baboon



Ballon

Barb

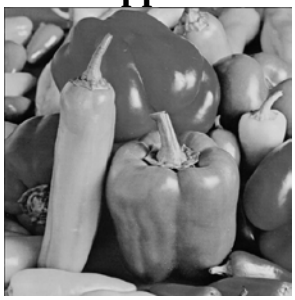
Camera

Couple

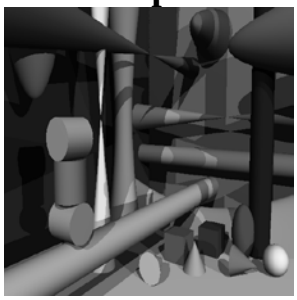
Goldhill

Lena

Peppers



Shapes



2. Standard images- SET-2

Boat

Car

Couple

House

Man

Parrot

Plant

House

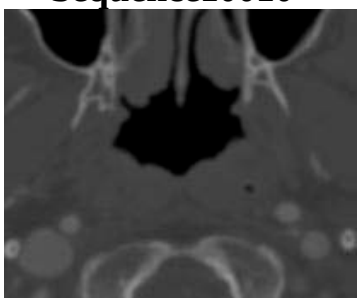
Waterwheel

Zelda

3. SLICE SET

Sequence10000

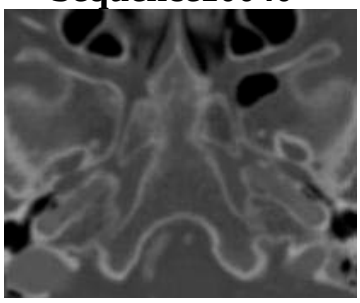
Sequence10010



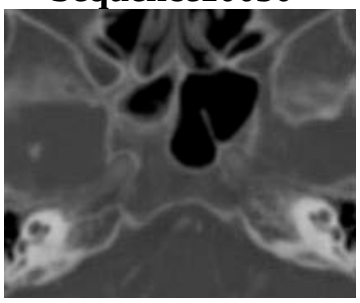
Sequence10020

Sequence10030

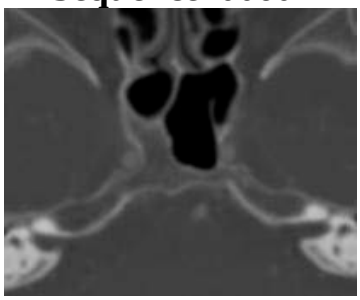
Sequence10040



Sequence10050

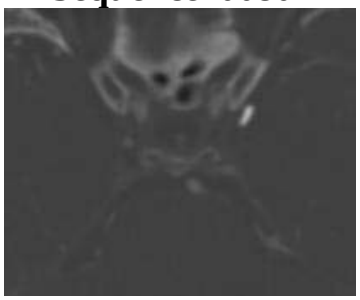


Sequence10060

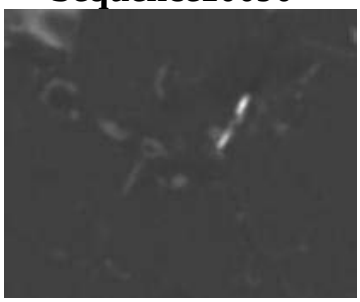


Sequence10070

Sequence10080

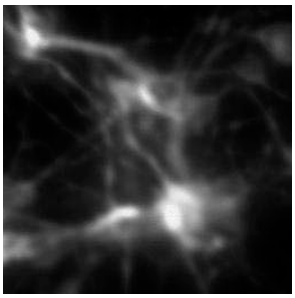


Sequence10090

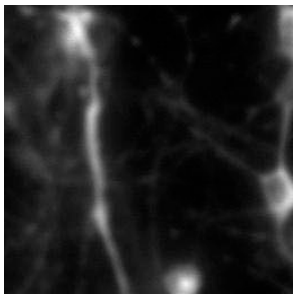


4. NEURITE SET

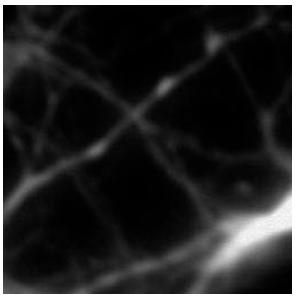
Neurite01



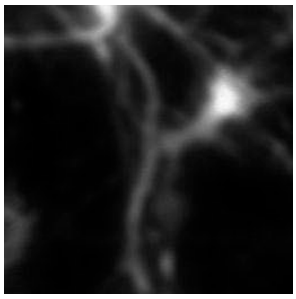
Neurite02



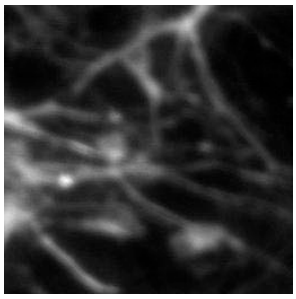
Neurite03



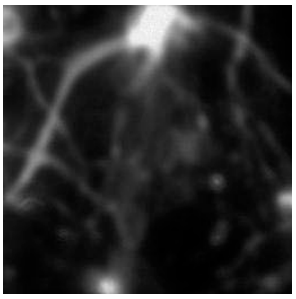
Neurite04



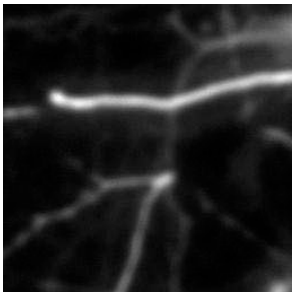
Neurite05



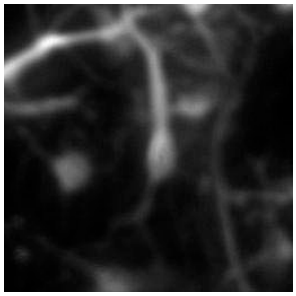
Neurite06



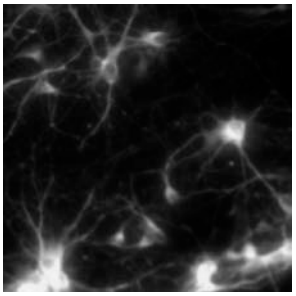
Neurite07



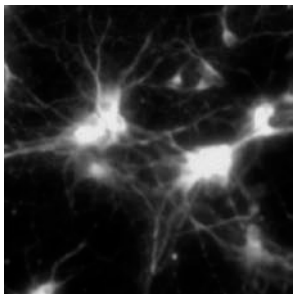
Neurite08



Neurite09

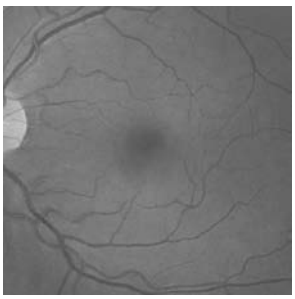


Neurite10

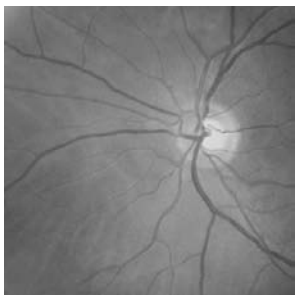


5. RETINAL SET

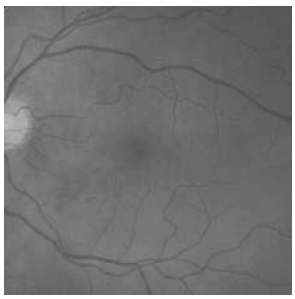
Retinal00



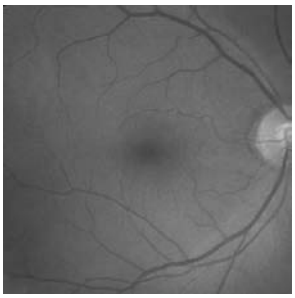
Retinal03



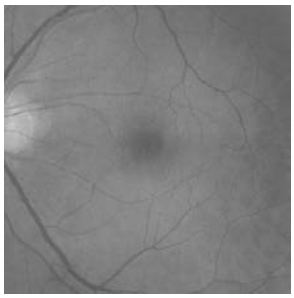
Retinal04



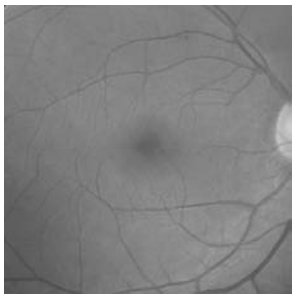
Retinal09



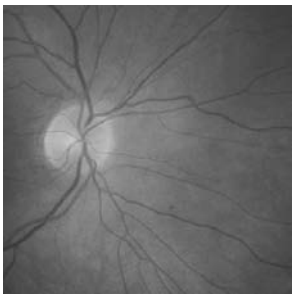
Retinal11



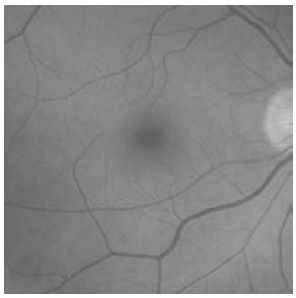
Retinal12



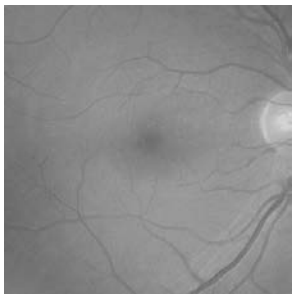
Retinal14



Retinal15



Retinal17



Retinal18