Color Image Compression Using Bit Reduction And Burrows Wheeler Transform

S.Annadurai
Department of Computer Science and Engineering, Govt. College of Technology
Coimbatore – 641013, India
anna_prof@yahoo.com

R.Shanmugalakshmi
Research Scholar
Department of Computer Science and Engineering, Govt. College of Technology
Coimbatore – 641013, India
ureka_udaya@yahoo.com

ABSTRACT

In this paper a new Image Compression technique for Color Images using *Bit Reduction and Burrows Wheeler Transform is proposed.* This method adopts the advantages in BWT based Text compression to compressing Images also. A Bit Encoding schemes that select blocks of pixel values from the source image and convert those pixel values in to equivalent Alphabets (A-Z, a-z), thereby we acquire a new encoded text file consisting of only the alphabets. The BWT based compression method can be now applied to the above said text file to compress it. To increase the compression rate some variant methods in the Encoding phase are also proposed in this paper. The experimental results have shown a significant improvement in the compression ratios compared to other techniques.

1. INTRODUCTION

Color images are widely used in almost all applications. High quality color images with higher resolution and huge memory spaces are the favorite of modern people. Color image compression is an important technique to reduce the image space and retain high image quality. The popularly used JPEG 2000 is the current color image compression standard which is based on Discrete Cosine Transform and run-length encoding procedures.

But, the wavelet Transform based image compression algorithm has excellent compression performance and is the core technique for the JPEG-2000 standard. Our paper deals with the process of *Compressing color Images using Encoding and BWT compression technique.*

The work is organized as follows. The compression and decompression techniques are discussed in Section 2. The detailed implementation is presented in Section 3. Experimental results are given in Section 4. Possible improvements are discussed in Section 5. Finally the concluding part is given in Section 6.

2. COMPRESSION AND DECOMPRESSION

The basic idea in this proposed method is to convert the RGB pixel values in to equivalent alphabets and then applying BWT Compression to compress the text file to the resultant compressed file.

The proposed Compression algorithm consists of two phases:

1) Encoding and
2) BWT Compression

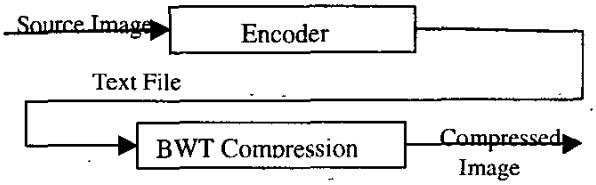


Fig 1 Block diagram of Compression Algorithm

2.1 Encoding Procedure

The Basic idea in this Encoding Scheme is to convert the pixel values of the image in to corresponding equivalent alphabets. Splitting the source file in to blocks and then converting each block of pixels in to equivalent alphabets does this. Scanning the input file for first 52 distinct pixel values chooses the blocks. Now each and every pixel value is assigned one corresponding alphabet and the assigned alphabet is placed in the Encoded file. Each occurrence of the pixel value will be replaced by the same alphabet in the

Encoded file. After the current block is over the source file is scanned for next 52 distinct characters for forming the next block. The process of block formation and Encoding is repeated until the end of the source file is reached.

2.2 BWT Compression

The idea in BWT Compression is to apply a reversible transformation to a block of text to form a new block that contains the same characters, but is easier to compress by simple compression algorithms. The transformation tends to group characters together so that the probability of finding a character close to another instance of the same character is increased substantially [1]. Text of this kind can easily be compressed with fast locally adaptive algorithms, such as move-to-front coding [4] in combination with the Arithmetic coding.

The Move-to-Front encoding (MTF) plays a vital role in this algorithm by increasing the number of zeros in the resulting strings before applying to Run Length Encoding (RLE) such that it reduces the redundancy of multiple characters by the redundancy of zeros. Thus it reduces the size of the strings considerably after the RLE encoding (second time) hence improving the performance of Arithmetic encoding. The fig 2 shows the various steps involved in the BWT compression technique.

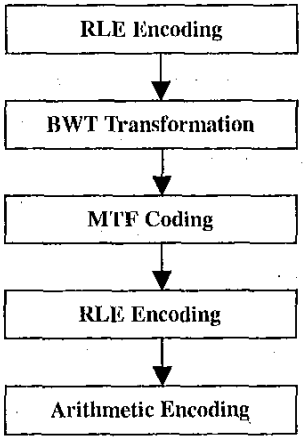


Fig 2 Stages in BWT Compression

The resultant of the BWT compression algorithm is the final compressed file. The mappings of pixel values to the alphabets are done using a 34 byte header ford each block.

2.3 Decompression

Decompression is done in two stages:
1) BWT Decompression and
2) Decoding Procedure

BWT Decompression is just a reverse process of the compression algorithm. Various stages involved in BWT Decompression are shown in fig 3.

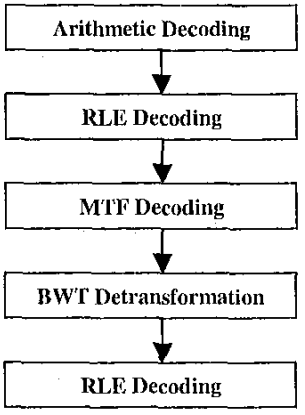


Fig 3 Stages in BWT Decompression

Decoding is the process of converting the alphabets in to their original pixel values. The header for each block will be 34 bytes with 2 bytes for specifying the block size and 32 bytes for storing all the 256 character occurrences. From the 34 byte header information the block index corresponding to the first two byte is obtained. The remaining 32 bytes are used to map the character to the corresponding pixel values. Since transform process is a reversible process the resultant image is same as the original image.

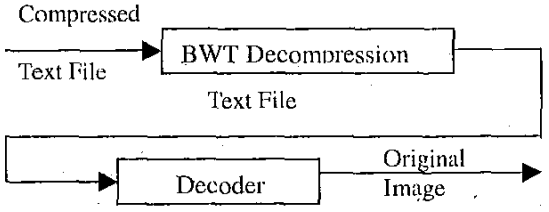


Fig 4 Stages of Decompression Technique

3. IMPLEMENTATION DETAILS

The proposed Compression technique has two main stages as mentioned before. BWT Compression is a well known technique which has been used in Text Compression. This BWT compression is used in the second stage of the compression. Construction of the Text file containing only the 52 alphabets from the image file without any loss is one of the major concerns. The mapping is done using a 34-byte header. This header will be for each and every block.

Scanning the input file until 52 distinct characters are obtained chooses blocks. Now the header is constructed by writing the size of the block in the first 2 bytes of the 34 bytes. The remaining 32 bytes aid in the mapping part. For each distinct pixel value of the source image corresponding ASCII value is found & bit in the array is set using the ASCII value as the index. Now the alphabets are assigned one by one for the distinctly occurred pixel values by scanning the array part sequentially. After mapping the alphabets, the pixel values are substituted by the assigned alphabets through out the block. So the block now will be containing only the alphabets. This process of dividing the source image in to blocks and constructing the header and also the encoded alphabets are applied till end of the source file is reached. After obtaining the encoded file BWT Compression is applied to get the final compressed file.

In the Decoding part, the ASCII value of the alphabet is taken and the rank is determined. The number of bits that are set in the 32 byte header part is counted till the count becomes equal to that of the rank. The index or the position in the 32 byte header will give the original Pixel value. In this way all the blocks are scanned and the resultant Decompressed file is obtained and the same is illustrated in fig 4.

4. EXPERIMENTAL RESULTS

The proposed method has been implemented. Various color images of varying sizes and of type 24 bit and 256 bitmap are taken for testing. The images used in the experimentation are shown in fig 5. It has been found that the quality of the final decompressed image is exactly the same as that of the original image as shown in fig 6.

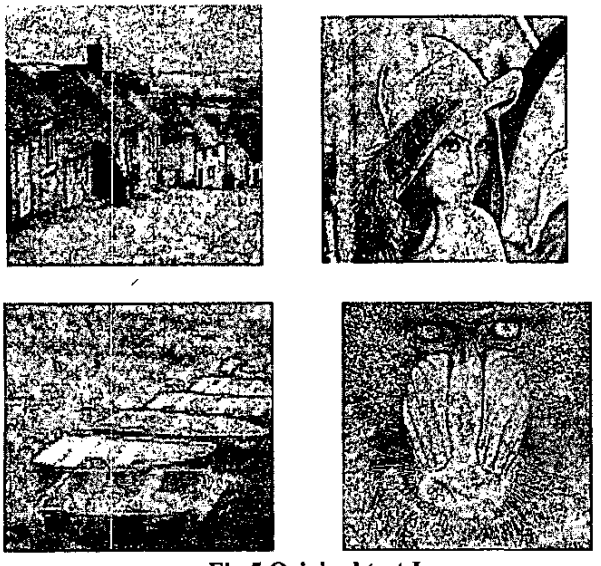
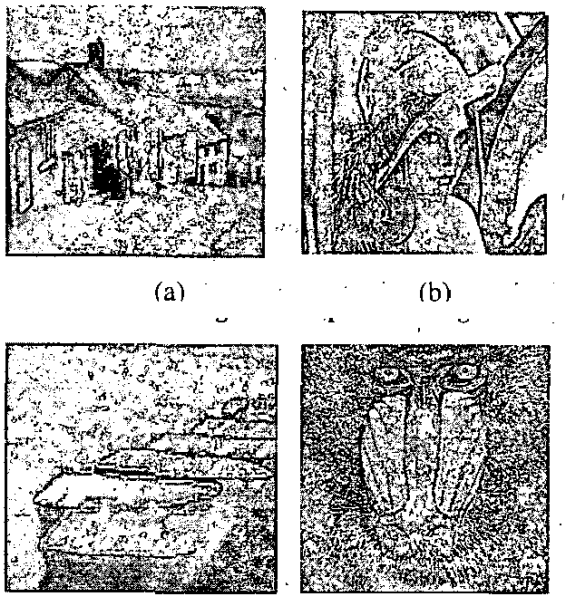


Fig 5 Original test Images

(a) Golden Hill Image (b) Lena Image
(c)Boat Image (d) Mandrill Image



(a) (b)

(a) Golden Hill Image (b) Lena Image
(c)Boat Image (d) Mandrill Image

Table 1 Experimental Results

File Name	Original Size (bytes)	Proposed Method (bytes)	Compression ratio (%)
Golden Hill. Bmp	49,102	45,261	8
Lena.bmp	196,608	192,512	2.1
Boat.bmp	44,339	41,062	7.4
Mandrill.bmp	196,608	175,104	10.9

5. FUTURE WORK

The proposed method for image compression is a loseless compression technique. The compression ratio achieved using the proposed approach is as high as 11.To increase the compression ratio above this one can use lossy compression technique at the cost of loss quality in images. Currently the authors are concentrating on developing a hierarchical approach which will consider all the three redundancies (coding, inter pixel & psycho visual) and eliminate them to the most extend possible. It is expected that this approach will produce compression ratios as high as 15.

6. CONCLUSION

This paper has aimed at providing a novel app
BWT transform, which is used for Text Cor (a)

the Image Compression. Encoding Technique uses a one to one mapping of the pixel values with the Alphabets. BWT Compression technique works by applying a reversible transformation to a block of text to make redundancy in the input more accessible to simple coding schemes. Also, the results have shown that the proposed method achieves compression comparable with statistical compression techniques. In this paper a new approach for lossless image compression using BWT is proposed. In this approach the redundancies of the pixels are eliminated in two different steps. Due to this, compression ratios obtained with different types of images are as high as 10.9. The conventional approaches like Huffman coding and Run Length Coding results compression ratios only 5 to 6. The experimental results shows that the proposed approach gives 50% more compression of images than the conventional approach.

7. REFERENCES

[1] M. Burrows and D. J. Wheeler. "A Block – Sorting Lossless Data Compression Algorithm," *SRC Research Report 124, Digital Systems Research Center, Palo Alto, CA*, May 1994.

[2] Rafael C. Gonzalez and Richard E. Woods "Digital Image Processing".

[3] Ambigananthan Ragavan and Prem Anand "Implementation of BWT Compression".

[4] J.L. Bentley, D.D. Sleator, R.E. Tarjan, and V.K. Wei. "A Locally Adaptive Data Compression Algorithm," *Communications of the ACM, Vol. 29, No. 4*, April 1986, pp. 320–330.