Gazi University Journal of Science
GU J Sci
29(4): 971-980 (2016)

Comp-Bit-List Size Improvement in Mespotine RLE and its Applications

Shiva PUTRA[1,♠], H.S.SHESHADRI[2], V.LOKESHA[3]

[1]*Department of ECE, Assistant Professor, Dr Ambedkar Institute of Technology, Bengaluru,560056,India and Research scholar @ Jain University, Bengaluru, India.*

[2]*Department of ECE, Professor and Research Dean, PES College of Engineering, Mandya,571401,India*

[3]*Department of Mathematics, Associate Professor, Vijayanagara Sri Krishnadevaraya University, Bellary,583101,India*

Received: 01/04/2016 Accepted: 03/09/2016

ABSTRACT

Run Length Encoding (RLE) is one of the simplest and primitive lossless data compression technique. RLE sometimes doubles the size of compressed data stream. To overcome this disadvantage, several algorithms, one of which is Mespotine RLE (MRLE), have been introduced. This paper introduces modification to MRLE technique in which the constant size 'Comp-Bit List' has been replaced by 'Variable Size Comp-Bit List' and refers to the new technique as improved – MRLE (iMRLE) technique. This paper discusses the details of 'Variable Size Comp-Bit List' and utilizes this concept for lossless compression and decompression of 8-bit grayscale medical images and extends the concept to 16-bit grayscale medical images. Image quality metrics such as Compression Ratio (CR), Root Mean Square Error (RMSE), Peak Signal-to-Noise Ratio (PSNR) and Entropy are used to check the quality of decompressed image obtained using iMRLE technique. Finally, the compression ratio achieved for existing MRLE and iMRLE techniques for 8-bit and 16-bit grayscale images have been assessed and iMRLE is found to produce best results for lossless compression and decompression of medical images.

Keywords: *Mespotine-RLE, iMRLE, variable size comp-bits, medical image, lossless compression.*

1. INTRODUCTION

Data compression is the art or science of representing information in compact form [1]. Data compression techniques usually eliminate redundant data and unnoticeable data which are far beyond normal human

♠Corresponding author, e-mail: shivaputra@dr-ait.org

perception [2]. The principal use of data compression techniques is to minimize the size of raw data thus eliminating unwanted bits which are of less importance and further encoding it to a form which is much suitable for storage and transmitting purposes [3].

Continued research in data compression algorithms paves way for faster access to data, improved compression logic and better storage techniques which ultimately result in considerable cost savings [4] [5].

As discussed in [6], compression techniques can be broadly classified into lossy and lossless.

Lossy compression techniques [7] permanently eliminate the data which are either redundant or are imperceptible by humans thus retaining vital data which conveys the intended information to the end user. Eliminating unwanted data can be done either in time-domain [8] or frequency domain [9]. Time domain operations include operating on the data directly (Ex: pixel values in images) [10] whereas frequency domain operations involve transforming data to frequency domain using Fourier Transforms [11], Discrete Cosine Transforms [12], Wavelet Transforms [13] and then eliminating undesired frequencies and converting back to time-domain. In all such techniques, it is practically impossible to recover original data from compressed data. However, there is minimal compromise in the quality of the compressed data. Lossy compression is usually employed in applications where higher compression ratio is required with little compromise in image quality [14]. Some of the standard lossy compression techniques include Transform coding [15], JPEG [16], Fractal Compression [17], etc.

Lossless compression techniques encode the data using suitable data encoding algorithm so that the original information can be decoded without any loss [18]. The primary purpose of lossless compression is to recover back the original information from compressed data. Such techniques are best suited for compression and decompression of text files, medical images, archives files, etc. Majority of the lossless compression techniques use statistical operations which can be sometimes slower compared to lossy compression techniques [19]. Some of the standard lossless compression techniques include Run-Length encoding (RLE) [20], Mespotine Run – Length encoding (MRLE) [21], Huffman Coding [22], Lempel-Ziv-Welch (LZW) [23], Arithmetic Coding [24], etc.

This paper studies Mespotine RLE, which is a lossless compression & decompression technique and introduces modifications to existing MRLE by replacing the constant size MRLE Comp-Bit List by Variable Size Comp-Bit List and refers to the new technique as improved-MRLE (iMRLE) technique. This technique is employed to compress and decompress 8-bit, 16-bit grayscale medical images to assess the performance of iMRLE in contrast to the existing MRLE technique.

2. LITERATURE REVIEW

Run Length Encoding technique is an entropy encoding technique which is lossless and is independent of the type of information being compressed [25]. With this technique it is possible to recover the exact original information from compressed data without any loss in data or quality. It allows user to obtain perfect replica of the original message.

RLE encoding technique can be discussed with an example. Consider the sample uncompressed data {B,C,A,D,D,D,D,D,D,E,E,E,E}. The RLE encoded data is{1,B,1,C,1,A,6,D,4,E}. In RLE encoded data stream, odd values represent run-count and even values represent run-value. Since each character occupies 1 byte (or 8 bits) of storage memory, the uncompressed data listed above occupies 13 bytes of storage memory and the compressed/encoded data occupies 10 bytes of storage memory.

Table 1. Rle Encoding Scheme Examples

Original data (Sample Pixel values of an 8-bit grayscale image)	Compressed/ Encoded Data	Original Data Size (Bytes)	Compressed Data Size (Bytes)
{B}	{1, B}	1	2
{E, E, E, E, E, E, E}	{7, E}	7	2
{A, A, C, B, B, B, B}	{2, A, 1, C, 4, B}	7	6
{B, B, C, C, A, A, A, A}	{2, B, 2, C, 4, A}	8	6
{A, C, E, E, D, D, A, E}	{1, A, 1, C, 2, E, 2, D, 1, A, 1, E}	8	12
{F, G, A, C, B, A, C, D, A, B}	{1, F, 1, G, 1, A, 1, C, 1, B, 1, A, 1, C, 1, D, 1, A, 1, B}	10	20
{A, B, C, D, E, F, G, H}	{1, A, 1, B, 1, C, 1, D, 1, E, 1, F, 1, G, 1, H}	8	16

As shown in Table – 1, the advantage of RLE scheme is that it requires a minimum of 2 bytes in best case scenario. This happens when the original data has a single character or has all characters same. However, RLE encoding scheme sometimes produces compressed data whose size is more than that of the original data. In the worst case scenario the size of compressed data is double the size of uncompressed data. This happens if the consecutive characters or all characters in the original data are different.

As known, RLE sometimes doubles the size of compressed data. To overcome this disadvantage several modifications were introduced to RLE.

Tsukiyama's method [26] transforms uncompressed data which includes two regions in which the first region consists of series of data whose occurrences is less than a predetermined value and second region consists of multiple occurrences of the data in region one. Such data can be compressed in two steps. First step counts the occurrences of the character string greater than the predetermined threshold and in second step it combines the data and its occurrences. This process is continued till the end of the data string is reached.

In [27] the author discusses data compression using both RLE and statistical encoding. In this technique a flag byte symbol is inserted between the run value and run count and in this technique, multiple statistical encoding tables are selected based on previously occurring data.

In [28] the author discusses the method for compressing a digitized waveform into a sequence of N-bit words which includes selection of the corresponding bit values from N data words and generating a value based on bit values. The next N input words are selected and the corresponding bits are used to generate next value. The steps are repeated for each bit of the input sequence and the generated data is run-length encoded to produce a compressed data.

In [29] the author discusses the design and implementation of a new RLE algorithm which is based on data chunking and packing which exploits the Cray gather-scatter vector hardware and multiple processors. This approach reduces the input-output and file storage requirements on average by an order of magnitude. By using this method applications such as the integration of environmental and global climate models become practical in real-time.

In [30] a mixed DCT and RLE technique has been introduced. The new technique is discussed for grayscale image compression and the experimental results that this method is advantageous as it is simple, fast with minimal error.

A new FPGA based compression technique has been discussed in [31] which reduces the size of bit stream while maintaining minimum decompression ratio. This technique discusses the smart arrangements of compressed bits which can significantly remove undesired overhead. It also discusses the combination of bitmask-based compression and RLE of repetitive patterns.

3. IMAGE QUALITY METRICS

This section discusses parameters which are required to assess image quality.

A. Root Mean Squared Error (RMSE)

RMSE [32] determines the square root of Mean Squared Error (MSE). Root Mean Squared Error is simply the square root of Mean Squared Error. Mean Squared Error is a parameter to evaluate the similarity between two images. MSE is the average of square of the pixel differences of compressed and decompressed image. The value of RMSE for compressed and decompressed images must be as least as possible. Ideal value is zero. In such cases, the two images under test are identical. RMSE is given in Equation (1) and MSE is given by Equation (2).

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (1)$$

$$RMSE = \sqrt{MSE} \quad (2)$$

B. Peak Signal-to-Noise Ratio (PSNR)

PSNR [33] depends on the Mean Squared Error (MSE) between original image and decompressed image. PSNR is measured on a logarithmic scale and its unit is Decibels (dB). When two images are similar, the value of MSE is minimum and the value of PSNR is maximum. In ideal case, when compressed and decompressed image are identical, MSE value is zero. In such cases the PSNR value is infinity.

The equation for PSNR is given in Equation (3) and Equation (4). Equation (3) depends on MSE and Equation (4) depends on RMSE. Either of the two equations can be used to determine PSNR.

$$PSNR = 10\log_{10}\frac{(2^n - 1)^2}{MSE}dB \quad (3)$$

$$PSNR = 20\log_{10}\frac{(2^n - 1)}{RMSE}dB \quad (4)$$

In an 8-bit grayscale image the maximum value of an image pixel is 255 i.e. $(2^n - 1)$.

C. Entropy

Histogram represents the probability of occurrence of different gray levels in a grayscale image. In order to express the distribution of different gray levels of a grayscale image as a single quantity, entropy is used. The individual pixels in the image can be considered as the symbols produced by information source with different gray levels as its states.

Entropy [34] or average information is defined as the expected value of information contained in each pixel value. It is given in bits by the Equation (5).

$$H = -\sum_{i=0}^{L-1} p(i)\log_2 p(i) \quad (5)$$

Maximum entropy occurs when all pixel values occur with equal probability $(1/L)$. It is given by equation (6).

$$H = -\sum_{i=0}^{L-1} \frac{1}{L} \log_2 \frac{1}{L} = \log_2 L \quad (6)$$

If the image under test is an 8-bit grayscale image, then the number of gray levels L is 256 $(2^8 = 256)$. As a result the maximum possible entropy is 8 bits.

D. Percentage Improvement in Compression Ratio

Percentage improvement factor is used to compare the compression ratio achieved using MRLE and iMRLE techniques. It is determined using Equation (7).

$$PI = \frac{iMRLE\ CR - eMRLE\ CR}{eMRLE\ CR} \times 100 \quad (7)$$

where

PI=Percentage Improvement

iMRLE CR=Improved MRLE Compression Ratio

eMRLE CR=Existing MRLE Compression Ratio

Percentage improvement factor in turn uses the values of MRLE compression ratio given by Equation (8) and iMRLE compression ratio given by Equation (9).

$$MRLE\ \ CR = \frac{IISB}{CISB + 32 bytes MRLE\ \ comp - Bit\,List} \quad (8)$$

where

IISB=Input image size in bytes

CISB=Compressed image size in bytes

$$iMRLE\ \ CR = \frac{IISB}{CISB + \mathrm{var}\,iable\ \ comp - Bit\,List} \quad (9)$$

4. MESPOTINE RLE (MRLE)

Mespotine RLE introduces few modifications to RLE to reduce the size of compressed data. First, it introduces the concept of compressible bit called 'Comp-Bit'. Number of MRLE Comp-Bits needed to compress the data is equal to the number of different characters in the data. In [21] author discusses the concept of MRLE considering 256 ASCII values. Since there are 256 ASCII values, 256 different Comp-Bits are needed, one for each character. The 256 Comp-Bits are combined to get 256 bits or 32 byte MRLE Comp-Bit List.

As an example, MRLE encoding and decoding steps as described in [21] is discussed here considering the sample data DDAAABCAACBDAAADBBBBB with an assumption that the sample data can have any length but is comprised of only four characters {A, B, C, D}.

A. Counting Occurrences

Counting occurrences of a pixel value is done batch-wise and stored in a variable Counter, which is required to generate MRLE Comp-Bit List in section 4.2.

Table 2. Counting Occurrences for MRLE Comp-Bit List Generation

Number of occurrences in first batch	Counter
One	-1
Two	0
Three	1
Four	2
Five	3
Six	4
Seven	5

B. Generating MRLE Comp-Bit List

Generating Comp-Bit List for the uncompressed data DDAAABCAACBDAAADBBBBB is discussed in the below steps.

Above data has 4 different characters A, B, C and D. So, consider 4 comp-Bits, one for each character.

Start with 'A'. In the data, DDAAABCAACBDAAADBBBBB, First batch of 'A' has three occurrences (1), second batch of 'A' has two occurrences (0) and third batch of 'A' has three occurrences (1). Sum the numbers: (1) + (0) + (1) = (2) > 0. Comp-Bit for 'A' = 1. So, 'A' is a compressible character.

Start with 'B'. In the data, DDAAABCAACBDAAADBBBBB, First batch of 'B' has one occurrence (-1), second batch of 'B' has one occurrence (-1) and third batch of 'B' has five occurrences (3). Sum the numbers: (-1) + (-1) + (3) = 1 > 0. Comp-Bit for 'B' = 1. So, 'B' is a compressible character.

Start with 'C'. In the data, DDAAABCAACBDAAADBBBBB, First batch of 'C' has one occurrence (-1), second batch of 'C' has one occurrence (-1). Sum the numbers: (-1) + (-1) = (-2) < 0. Comp-Bit for 'C' = 0. So, 'C' is not a compressible character.

Start with 'D'. In the data, DDAAABCAACBDAAADBBBBB, First batch of 'D' has two occurrences (0), second batch of 'D' has one occurrence (-1), third batch of 'D' has one occurrence (-1). Sum the numbers: (0) + (-1) + (-1) = (-2) < 0. Comp-Bit for 'D' = 0. So, 'D' is not a compressible character.

So, the MRLE Comp-Bit List is: 1100 (4 bits).

C. MRLE encoding/compression using MRLE Comp-Bit List

Uncompressed data is DDAAABCAACBDAAADBBBBB and MRLE Comp-

Bit List is 1100. MRLE encoding or compression is discussed in below steps.

First character in data DDAAABCAACBDAAADBBBBB is ‘D’. Comp-Bit for D is ‘0’, which indicates ‘D’ is not a compressible character. So, in the encoded data, retain ‘D’ as it is. Encoded Data is D.

Second character in data DDAAABCAACBDAAADBBBBB is ‘D’, Comp-Bit for ‘D’ = ‘0’, not compressible, retain ‘D’ as it is. Encoded Data is DD.

Third character in data DDAAABCAACBDAAADBBBBB is ‘A’, Comp-Bit for ‘A’ = ‘1’, compressible, apply RLE (A occurs 3 times). Encoded Data is DDA3. Skip next 2 characters i.e. AA.

Sixth character in data DDAAABCAACBDAAADBBBBB is ‘B’, Comp-Bit for ‘B’ = ‘1’, compressible, apply RLE (B occur 1 time). Encoded Data is DDA3B1.

In similar steps, one can get encoded data as DDA3B1CA2CB1DA3DB5. Size of uncompressed data is 21 characters = 21x8 = 168 bits (1 byte or 8 bits for each character). Size of compressed data is 18 characters = 18 x 8 = 144 bits + 4 Comp-Bits.

D. MRLE decoding/decompression using MRLE Comp-Bit List

Compressed data is DDA3B1CA2CB1DA3DB5 and MRLE Comp-Bit List is 1100. MRLE decompression is discussed in below steps.

First character in encoded data DDA3B1CA2C1B1DA3DB5 is ‘D’, Comp-Bit for ‘D’ is ‘0’, not compressible, retain ‘D’ as it is. Decoded data is D.

Second character in encoded data DDA3B1CA2C1B1DA3DB5 is ‘D’, Comp-Bit for ‘D’ is ‘0’, not compressible, retain ‘D’ as it is. Decoded data is DD.

Third character in encoded data DDA3B1CA2C1B1DA3DB5 is ‘A’, Comp-Bit for ‘A’ is ‘1’, compressible, next character to ‘A’ is ‘3’ which indicate occurrence. So, ‘A’ has three occurrences. Decoded data is DDAAA. Skip one position.

Fifth character in encoded data DDA3B1CA2C1B1DA3DB5 is ‘B’, Comp-Bit for ‘B’ is ‘1’, compressible, next character to ‘B’ is ‘1’ which indicate occurrence. So, ‘B’ has one occurrence. Decoded data is DDAAAB. Skip one position.

Seventh character in encoded data DDA3B1CA2C1B1DA3DB5 is ‘C’, Comp-Bit for ‘C’ is ‘0’, not compressible, Retain ‘C’ as it is. Decoded data is DDAAABC.

In similar steps, one can get decompressed data as DDAAABCAACBDAAADBBBBB = 21 characters = 21x8 = 168 bits.

5. DISADVANTAGES OF MRLE

As discussed in [21], major advantage of MRLE technique is that it performs best compared to Packbits, Tsukiyama’s Method and Standard RLE techniques. However, MRLE technique specifies fixed size for MRLE Comp-Bit List posing as a disadvantage. For 256 different characters, size of MRLE Comp-Bit List is 256 bits or 32 bytes which is fixed even if the uncompressed data has only one character out of 256 different characters.

As an example, consider a data stream having five different characters, A, B, C, D and E which require 5 Comp-Bits, one for each character. If the uncompressed data is ABCDE (5x8 = 40 bits), then the MRLE Comp-Bit List is 00000 (5 bits) and MRLE compressed data is ABCDE (5 x 8 = 40). So, the overall size of compressed data is the sum of the size of MRLE Comp-Bit List and MRLE compressed data i.e., (40 bits compressed data) + (5 bits Comp-Bit List) = 45 bits. The disadvantage is that even though, all comp-bits are zeros ‘00000’, it is still being stored. In this case, the decompression steps can be designed such that, if only compressed data is present without Comp-Bit List, then it must assume all Comp-Bits are zero.

Consider another example in which in which the data stream has five different characters, A, B, C, D and E which require 5 Comp-Bits, one for each character. If the uncompressed data is AAAAB (5x8 = 40 bits), then the MRLE Comp-Bit List is 10000 (5 bits) and MRLE compressed data is A4B (3x8 = 24 bits). So, the overall size of compressed data is the sum of size of MRLE Comp-Bit List and MRLE compressed data i.e., (24 bits compressed data) + (5 bits Comp-Bit List) = 29 bits. The disadvantage here is that even though, only one comp-bit is 1, five Comp-Bits, ‘10000’, is being stored which isn’t required.

This paper improves the aforesaid disadvantages by introducing the concept of iMRLE – Improved MRLE Comp-Bit List.

6. PROPOSED METHODOLOGY – ‘IMPROVED MRLE (IMRLE)’ OR ‘VARIABLE SIZE MRLE COMP-BIT LIST’

iMRLE introduces modifications to existing MRLE Comp-Bit List, and discusses how the fixed size of 32 bytes MRLE Comp-Bit List can be modified to a Variable size Comp-Bit List.

MRLE algorithm considers 256 ASCII values, so there are 256 Comp-Bits, one for each character. So, length of Comp-Bit-List will be 256 bits or 32 bytes (32 bytes is fixed). As an example, for an 8-bit grayscale image, each pixel value ranges from 0 to 255. There are 256 (0 to 255) different values. For, 256 different pixel values, 256 bits (32 bytes) MRLE Comp-Bit-List is required.

Consider the MRLE Comp-Bit List obtained for a chest x-ray medical image shown in Table I.

100000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
0000000000000000000000000000001

Comp-Bit value is ‘1’ for pixel values ‘0’, ‘255’; and ‘0’ for other pixel values, which is an indication that ‘0’ and ‘255’ are compressible. In the above 32 byte MRLE Comp-Bit List, only 2 Comp-Bits are useful. Storing the remaining Comp-Bits is not useful. This is achieved using improved – MRLE Comp-Bit List.

A. Generating Variable Size Comp-Bit List from MRLE Comp-Bit List

Generating Variable Size Comp-Bit-List (iMRLE Comp-Bit List) from MRLE Comp-Bit List can be described with an example.

Consider the 256 bits or 32 Bytes MRLE Comp-Bit-List for 256 pixel values obtained for 8 bit chest x-ray image as shown below.

100000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
0000000000000000000000000000001

Existing MRLE algorithm stores the above 256 bits or 32 bytes Comp-Bit-List along with encoded data. As an improvement, this paper suggests the next few steps.

Convert the above 256 bits to 32 byte integers *integerData*. Here, each integer value represents 8-bit data. Ex: 128 is the decimal equivalent of binary 10000000.

integerData = {128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}

Store the previous 32 byte *integerData* in a temporary variable *numbersNew.*

numbersNew = [128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Extract non-zero values of numbersNew to a new variable indexVal.

indexVal =[128, 1]

Consider another variable, cmpBitList1 initialized with 32 zero bits, one for each value in numbersNew.

cmpBitList1 = [00000000000000000000000000000000]

If any value in *numbersNew* is greater than zero, set its corresponding comp-bit to ‘1’ in *cmpBitList1*. So, the updated *cmpBitList1* is

cmpBitList1 = [10000000000000000000000000000001]

Convert the above 32 bits binary data to 4 bytes *cmpBitList2*.

cmpBitList2 = [128, 0, 0, 1]

Concatenate *cmpBitList2* and *indexVal* to get *cmpBitListNew*

cmpBitListNew = [128, 0, 0, 1, 128, 1]

So, iMRLE Comp-Bit List is [128, 0, 0, 1, 128, 1] in which the first 4 values indicates header and the remaining values indicate index. As a result, 32 byte fixed size MRLE Comp-Bit-List has been converted to 6 bytes Variable Size Comp-Bit-List (iMRLE Comp-Bit List).

A Variable size Comp-Bit List can have a minimum of zero bytes and a maximum of 32 bytes for data having 256 different characters.

B. Regenerating original 32 Byte MRLE Comp-Bit List from Variable Size Comp-Bit List (iMRLE Comp-Bit List) during decompression

MRLE decompression steps require MRLE Comp-Bit List. So, one must obtain MRLE Comp-Bit List from Variable Size Comp-Bit List. This is described with an example. Suppose that the iMRLE Comp Bit List stored is [128, 0, 0, 1, 128, 1] along with compressed data. Regenerating MRLE Comp-Bit List from iMRLE Comp-Bit List is discussed below with an example.

Extract first 4 bytes of iMRLE Comp Bit List and save in *cmpBitListNewRX*.

cmpBitListNewRX = [128, 0, 0, 1]

Extract remaining bytes in iMRLE Comp Bit List and save in *indexValRx*.

indexValRx = [128, 1]

Convert *cmpBitListNew* to 32 bit binaries

cmpBitListNewBin = [10000000000000000000000000000001]

Create a variable *numbersRegen* with 32 zeros (integers)

numbersRegen = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Replace values of *numbersRegen* by indexVal, at the positions indicated by *cmpBitListNewBin*. So the updated *numbersRegen* is

numbersRegen = [128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Convert above 32 byte integer data to 256 bits binary data which gives MRLE Comp-Bit List

100000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
0000000000000000000000000000001

By using above 256 MRLE Comp-Bit List, compressed data can be decompressed.

7. IMPLEMENTATION

Implementation of iMRLE comp-Bit List generation can be grouped into any of the three categories as discussed below.

A. Category 1 (Best Case)

Assume MRLE Comp-Bit-List in which all 256 comp-bits are zeros.

000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
0000000000000000000000000000000

Generate Variable Size MRLE Comp-Bit List from MRLE Comp-Bit List as shown below

numbersNew = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

cmpBitList1 = [00000000000000000000000000000000]

cmpBitList2 = [0, 0, 0, 0]

indexVal = [ ]

cmpBitListNew = [0, 0, 0, 0]

In this category 32 Bytes Mespotine Comp-Bit-List has been converted to 4 bytes Variable size Comp-Bit-List. Store only compressed data without Variable-Size Comp-Bit List. As a result 32 bytes of storage memory is saved.

During decompression MRLE Comp-Bit List must be recovered from Variable Size Comp-Bit List. Since only compressed data is present without iMRLE Comp-Bits, the decompression logic assumes 256 bits or 32 byte MRLE Comp-Bit List with all zeros.

000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
0000000000000000000000000000000

Decompression is performed on compressed data using the above newly generated MRLE Comp-Bit List.

B. Category 2

Assume 32 byte MRLE Comp-Bit-List as shown below.

100000000000000100000100000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000011000001000000000000
000000000000000000000000000000000000000000000
0000000000000000000000010000001

New Variable Size Comp-Bit-List is generated as represented in below steps.

numbersNew = [128, 2, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 193, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 129]

cmpBitList1 = [11100000000000000000100000000001]

cmpBitList2 = [224, 0, 8, 1]

indexVal = [128, 2, 8, 193, 129]

cmpBitListNew = [224, 0, 8, 1, 128, 2, 8, 193, 129]

In this category 9 Bytes Variable size Comp-Bit List is used instead of fixed 32 byte MRLE Comp-Bit-List and as a result 23 bytes of storage memory is saved. The size of Variable Size MRLE Comp-Bit List can vary from 4 to 36 bytes depending on the values of MRLE Comp-Bit List. This category stores Variable Size Comp-Bit List if its size is less than or equal to 31 bytes.

During decompression MRLE Comp-Bit List is obtained from Variable Size Comp-Bit List as shown in below steps.

Extract first 4 bytes of cmpBitListNew and save in *cmpBitListRx*

cmpBitListNew = [128, 0, 0, 1]

Extract remaining bytes in cmpBitListNew and save in *indexValRx*

indexValRx = [128, 1]

Convert *cmpBitListNew* to 32 bit binaries

cmpBitListNewBin = [10000000000000000000000000000001]

Create a variable *numbersRegen* with 32 zeros (integers)

numbersRegen = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Replace values of *numbersRegen* by indexVal, at the positions indicated by *cmpBitListNewBin*. So the updated *numbersRegen is*

numbersRegen = [128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Convert above 32 byte integer data to 256 bits binary data which gives MRLE Comp-Bit List

100000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000
0000000000000000000000000000001

Using above 256 bit MRLE Comp-Bit List, original data can be obtained by decompressing compressed data.

C. Category 3 (Worst Case)

Assume 32 byte MRLE Comp-Bit-List as shown below.

100000000000000100000100000000000000000000001
000000010000000100000001000000010000000100000
001000000010000000100000001000000010000000100
000001000000010000000100011000001000010000000
100000001000000010000000100000001000000010000
0001000000010000000100010000001

New Variable Size Comp-Bit-List is generated as represented in below steps *numbersNew = [128, 2, 8, 0, 0, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 193, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 129]*

cmpBitList1 = [11100111111111111111111111111111]

cmpBitList2 = [231, 255, 255, 255]

indexVal = [128, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 193, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 129]

cmpBitListNew=[231, 255, 255, 255, 128, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 193, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 129]

In this category, Variable Size Comp-Bit-List size 34 bytes which is more than 31 bytes. Hence, ignore the variable size comp-bit list and transmit the existing MRLE Comp-Bit-List along with compressed data. If the size of Variable Size Comp-Bit List is greater than or equal to 31 bytes, existing MRLE Comp-Bit List is transmitted instead of Variable size Comp-Bit List, because decompression process is faster in such cases. Decompression is performed on compressed data using MRLE Comp-Bit List as described in section 4.4.

8. IMPLEMENTATION RESULTS AND ANALYSIS

Sample 8 bit and 16 bit medical images used in the MATLAB implementation of MRLE and improved-MRLE (iMRLE) techniques are shown in Table III. Implementation results in Table V shows that the compression ratio achieved using improved-MRLE is more than existing MRLE technique. The compression ratio achieved using improved-MRLE technique increases as pixel depth of an image increases i.e., 8 bit, 16 bit, 32 bit and so on. This is because the number of comp-bits required for 8 bit, 16 bit and 32 bit images are 256 bits (32 byte), 65536 bits (8192 bytes) and 4294967296 bits (536870912 bytes) respectively. Since improved-MRLE technique uses Variable Size Comp-Bit List, its size reduces significantly when it is stored compared to MRLE technique. As a result the Percentage Improvement using Compression Ratio factor increases for 16-bit grayscale compared with 8 bit images as shown in Table V. Percentage Improvement factor for 32 bit images will be higher than 16-bit images and so on. As the quality of the image increases, higher compression rates are achieved and Percentage Improvement in Compression rate factor increases.

The RMSE and PSNR achieved using MRLE and also improved-MRLE are zero and infinity respectively. Such results are evident because the compression and decompression is lossless and the decompressed image is identical to the original image.

Table 3. Input Images used in MRLE and Improved MRLE (iMRLE Techniques)

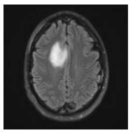
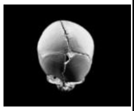
Input image name	**Input image**	**Image type**	**Resolutio n in Pixels**	**Image size in bytes**
Brain Tumor		8-bit grayscal e	256 x 256	65536
Chest X-Ray		8-bit grayscal e	400 x 329	13160 0
Skull		8-bit grayscal e	350 x 280	98000
Arm Fractur e		16-bit grayscal e	120 x 160	38400
Ankle		16-bit grayscal e	150 x 150	45000
Spine		16-bit grayscal e	130 x 130	33800

Table 4. MRLE Implementation Results for 8-bit and 16-bit Medical images

Input image	**MRLE compressed image size in bytes**	**MRLE compression ratio**	**MRLE Comp Bit List in bytes**
Brain Tumor	43006	1.5238	32
Chest X-Ray	107473	1.2244	32
Skull	26731	3.6662	32
Arm Fracture	20922	1.8354	8192
Ankle	27932	1.6111	8192
Spine	22908	1.4755	8192

Table 5. iMRLE Implementation Results for 8-bit and 16-bit Medical images

Input image	iMRLE compressed image size in bytes	iMRLE compression ratio	iMRLE Comp Bit List in bytes	Percentage improvement in compression ratio
Brain Tumor	42979	1.5248	5	0.0628
Chest X-Ray	107447	1.2247	6	0.0242
Skull	26705	3.6697	6	0.0955
Arm Fracture	13755	2.7917	1025	52.1031
Ankle	20765	2.1671	1025	34.5106
Spine	15741	2.1473	1025	45.5303

9. CONCLUSION

MRLE proves to be a better method for lossless compression and decompression of medical images. However the primary disadvantage of MRLE is that the size of the MRLE Comp-Bit List is fixed. It is 32 bytes for 8-bit image (256 different pixel values), 8192 bytes for 16-bit image (65536 different pixel values) and so on. As a solution to this problem, this paper has introduced modification to MRLE referred to as iMRLE technique in which the actual MRLE Comp-Bit List has been replaced by Variable Size Comp-Bit List. Implementation details show that for practical medical images, the size of Variable Size Comp-Bit List is less than the MRLE Comp-Bit List. So, better compression rate is achieved using improved – MRLE technique. The compression ratio achieved using improved-MRLE technique increases as pixel depth of an image increases i.e., 8 bit, 16 bit, 32 bit and so on because the size of Variable-Size Comp-Bit List will be significantly less compared to MRLE Comp-Bit List. On an average, the compression ratio achieved using improved – MRLE technique increases by 0.0608 percentage for 8 bit grayscale medical images and 44.048 percentage for 16 bit grayscale medical images.

Acknowledgment

The authors would like to thank the Management, Panchajanya Vidya Peetha Welfare Trust (Regd), Bengaluru, Dr. C. Nanjundaswamy, Principal, Dr. R.Murali, Dr M V Mandi and Dr S Ramesh of Dr. Ambedkar Institute of Technology, Bengaluru for their assistance, suggestions, insight and valuable discussion over the course of this research work

.

CONFLICT OF INTEREST

No conflict of interest was declared by the authors.

REFERENCES

[1] Khalid Sayood, “Introduction to Data Compression”, *Newnes Publications*, 2012, ISBN 978-012-41-5796-5

[2] Mohammad Ali Kakhaee, Farokh Marvasti, “A Survey on Digital Data Hiding Schemes: Principals, Algorithms, and Applications”, *The ISC International Journal of Information Security* 5.1: 5, 2013

[3] Omar Adil Mahdi, Mazin Abed Mohammed, Ahmed Jasim Mohamed, “Implementing a novel approach an convert audio compression to text coding via hybrid technique”, *International Journal of Computer Science Issues* 9.6: 53-59, 2012

[4] Kishore A Kotteri, Amy E. Bell, Joan E. Carletta, “Design of multiplierless, high-performance, wavelet filter banks with image compression applications”, *Circuits and Systems I: Regular Papers, IEEE Transactions* on 51.3: 483-494, 2004

[5] Qin Lu, “Low-complexity and energy efficient image compression scheme for wireless sensor networks”, *Computer Networks* 52.13: 2594-2603, 2008

[6] Amine Nait-Ali, Christine Cavaro-Menard, “Compression of Biomedical Images and Signals”, *John Wiley & Sons*, 2008, ISBN: 978-1-84821-028-8

[7] Asha Latha, Permender Singh, “Review of Image Compression Techniques”, *International Journal of Emerging Technology and Advanced Engineering*, ISSN 2250-2459, ISO 9001: 2008 Certified Journal, Volume 3, Issue 7, 2013

[8] Konstantinos N, Plataniotis, Anastasios N. Venetsanopoulos, “Color image processing and applications”, *Springer-Verlag New York*, 2013, ISBN: 3-540-66953-1 560

[9] Uvais Qidwai, C.H. Chen, “Digital Image Processing: An Algorithmic Approach with MATLAB”, *Chapman and Hall/CRC*, 2009, ISBN 9781420079500

[10] Roger Bourne, “Fundamentals of Digital Imaging in Medicine”, *Springer-Verlag London*, 2010, ISBN 978-1-84882-086-9.

[11] P. Mansfield, I. L. Pykett, “Biological and medical imaging by NMR”, *Journal of Magnetic Resonance*, 213.2: 513-531, 2011

[12] Qiang Yang, Hua Jun Wang, Xue Gang Luo, “An Improved Algorithm for Color Medical Image Compression Based on DCT”, *Applied Mechanics and Materials*, Vol. 602, 2014

[13] Ruchika, Mooninder Singh, Anant Raj Singh, “Compression of Medical Images Using Wavelet Transforms”, *International Journal of Soft Computing and Engineering (IJSCE)*, ISSN: 2231-2307, 2012

[14] D. Smutek, “Quality measurement of lossy compression in medical imaging”, *Prague medical report* 106.1: 5-26, 2005

[15] Michael Lustig, “Compressed sensing MRI”, *Signal Processing Magazine, IEEE 25.2*: 72-82, 2008

[16] Shaou-Gang Miaou, Fu-Sheng Ke, Shu-Ching Chen, “A lossless compression method for medical image sequences using JPEG-LS and interframe coding”, *Information Technology in Biomedicine, IEEE Transactions* on 13.5: 818-821, 2009

[17] S. Bhavani, K. Thanushkodi, “A survey on coding algorithms in medical image compression”, *International Journal on Computer Science and Engineering 2.05*: 1429-1434, 2010

[18] Ming-Bo Lin, Jang-Feng Lee, Gene Eu Jan (2006); “A Lossless Data Compression and Decompression Algorithm and its Hardware Architecture”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 14, No. 9: 925 – 936

[19] Salomon, David, Giovanni Motta, “Handbook of data compression”, *Springer Science & Business Media*, 2010

[20] B.W.R Agung, F. P. Permana, “Medical image watermarking with tamper detection and recovery using reversible watermarking with LSB modification and run length encoding compression, Communication”, *Networks and Satellite (ComNetSat), IEEE International Conference, 2012*

[21] Meo Mespotine, “Mespotine-RLE-basic v0. 9-An overhead-reduced and improved Run-Length-Encoding Method”, *arXiv preprint arXiv: 1501.05542*, 2015

[22] Yu-Chen Hu, Chin-hen Chang, “A new lossless compression scheme based on Huffman coding scheme for image compression”, *Signal Processing: Image Communication* 16.4: 367-372, 2010

[23] K. Rajeswari, K. Kavitha, G. Boopathi Raja, “High Efficient Image Compression Using Lempel-Ziv-Welch Algorithm”, *Digital Image Processing* 7.2: 44-47, 2015

[24] Jiaji Wu, “Arithmetic coding for image compression with adaptive weight-context classification”, *Signal Processing: Image Communication* 28.7: 727-735, 2013

[25] Fred Halsall, “Multimedia Communications – Applications, Networks, Protocols and Standards”, *Pearson Education Ltd*, 2008, ISBN – 978-81-317-0994-8

[26] Tokuhiro Tsukiyama, “Method and system for data compression and restoration”, *U.S. Patent No. 4,586,027*, 1986

[27] Edward L Hauck, “Data compression using run length encoding and statistical encoding”, *U.S. Patent No. 4,626,829*, 1986

[28] Douglas C Stevens, “Bit-wise run-length encoding for data compression”, *U.S. Patent No. 5,049,880*, 1991

[29] G. Davis, “Parallel run length encoding compression: Reducing I/O in dynamic environmental simulations”, *International Journal of High Performance Computing Applications* 12.4: 396-410, 1998

[30] Ling-fang Zhu, Ren-ren Liu, “DCT and RLE Mixed Lossy Compression of Gray Image Based on Matlab”, *Computer Knowledge and Technology* 21: 074, 2009

[31] P. Hemnath, V. Prabhu, “Compression of FPGA bitstreams using improved RLE algorithm”, *Information Communication and Embedded Systems (ICICES) International Conference on. IEEE*, 2013

[32] Hasan Demirel, Gholamreza Anbarjafari, “Discrete wavelet transform-based satellite image resolution enhancement”, *Geoscience and Remote Sensing, IEEE Transactions on* 49.6: 483-495, 2011

[33] J.C Yoo,; C. W. Ahn, “Image matching using peak signal-to-noise ratio- based occlusion detection”, *Image Processing, IET 6.5:* 483-495, 2012

[34] M. Rangaraj Rangayyan, “Biomedical image analysis”, *CRC press*, 2004

[35] Shivaputra, H.S. Sheshadri, V. Lokesha “An Efficient Lossless Medical Image Compression Technique for Telemedicine Applications”, Computer Applications: An International Journal (CAIJ), Vol.2, No.1, pp-63-69, 2015

[36] Shivaputra, H. S. Sheshadri and V. Lokesha “An Exquisite Approach for Image Compression Technique using Lossless Compression Algorithm for ROI & Non-ROI regions” International Journal of Computational Science, Information Technology and Control Engineering (IJCSITCE) Vol.2, No.1/2, pp 1-13, 2015.

[37] Shivaputra, H. S. Sheshadri and V. Lokesha “A Naïve Visual Cryptographic Algorithm for the Transfer of a Compressed Medical Images” International Journal of Recent Contributions from Engineering, Science & IT (iJES), eISSN: 2197-8581, Vol.3, No.4, pp 26-36, December 2015.