Journal of Education for Pure Science- University of Thi-Qar
Vol.10, No.2 (June, 2020)
Website: jceps.utq.edu.iq Email: jceps@eps.utq.edu.iq

DOI: http://doi.org/10.32792/utq.jceps.10.02.010

A proposed technique for improving run length encoding

⁽Dhyaa Alrahman Latef Thajel Prof.Dr. Kadhim Mahdi Hashim

University of thi-qar, college of education for pure science

Received 20/2/2020 Accepted 6/8/2020 Published 30/11/2020

Abstract:

Image compression, is used to reduce the quantity of pixels used in image representation without excessively change image visualization. Reducing image size enhance images sharing, transmitting and storing. Data compression has become more important than ever, due to the increasing demand for internet use and exchange of a huge amount of images, videos, audio and documents as well as growing demand for electronic archiving by government departments that produce thousands of documents per day. In this paper, a proposed technique for improving run length encoding results will be presented.

The proposed technique is a lossless and completed technique, it is consisting of two parts the compression part and decompression part. The compression part contains some basic stages such as: pre-processing, run length encoding, replace maximum values by unused values, minimize levels, delta encoding, while the decompression part is the revers of compression part.

This technique is applied on twenty documents and compared with RLE. The experimental results showed that the proposed technique gives a higher compression ratio than the RLE.

Keywords: Data compression, run-length encoding, delta encoding, lossless.

Introduction:

Image compression is an application of data compression that encodes the original image with few bits. The objective of image compression is to reduce the redundancy of the image and to store or transmit data in an efficient form. The field of image compression continues to grow at a rapid pace. As we look to the future, the need to store and transmit images will only continue to increase faster than the available capability to process all the data. Even with the rapid growth in computer power and increase Internet bandwidth, the ability to process and transmit the desired amount of image data continues to be problematic [1].

2. Background:

2.1 Run-length encoding (RLE) [2, 3]:

Is one of the simplest forms of data compression methods, the principle of RLE is to exploit the repeating values in a source. This repeating string of characters is called a run. The algorithm counts the consecutive repeating amount of a symbol and uses that value to represent the run.

In RLE, runs of data are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, simple graphic images such as icons, line drawings, and animations.

2.2 Lossless compression:

reduces bits by identifying and removing statistical redundancy. No information is lost in lossless compression. Lossless compression methods reduce size whilst preserving all of the original image information, and therefore without degrading the quality of the data [4]. Some of these techniques are like run length encoding, entropy encoding, Huffman encoding, arithmetic coding, Lempel–Ziv–Welch coding, deflation, chain codes, delta encoding and block coding.

2.3 Delta encoding:

Delta encoding represents streams of compressed pixels as the difference between the current value and the previous value [5]. The first value in the delta encoded file is the same as the first value in the original image. While the following pixels in the encoded file are equal to the difference between the corresponding value in the input data, and the previous value in the input data [6].

2.4 BMP file format(.bmp):

The bitmap file format deal with graphic file related to microsoft windows OS. Normally these files are uncompressed so they are large. These files are used in basic windows programming [7]. BMP files always contain RGB data. The file can be1-bite: 2 colors (monochrome), 4-bit: 16 colors, 8-bit: 256 colors, 24-bit: 16777216 colors, mixes 256 tints of Red with 256 tints of Green and Blue [8]

3. The Proposed Technique:

3.1 Compression part;

The proposed compression part is described in figure (3.1).

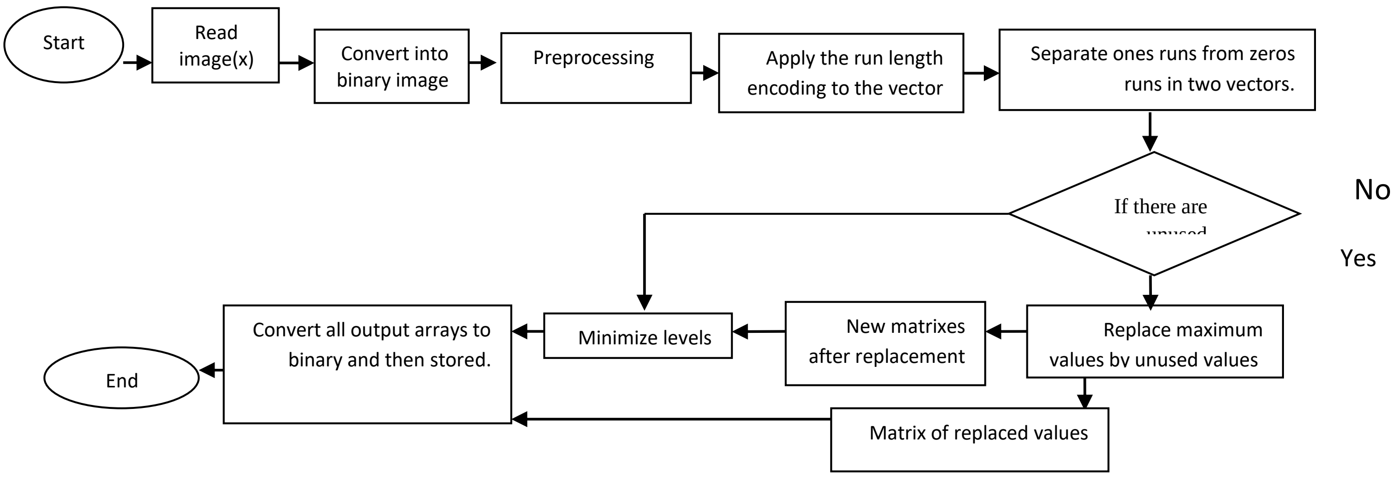
Start
Read image(x)
Convert into binary image
Preprocessing
Apply the run length encoding to the vector
Separate ones runs from zeros runs in two vectors.
If there are unused
No
Yes
Replace maximum values by unused values
New matrixes after replacement
Minimize levels
Convert all output arrays to binary and then stored.
End
Matrix of replaced values

Figure (3.1): The compression part for the proposed technique

3.1.1 Read image and convert into binary image:

The first step is reading a document image from its location in storage location in computer, if the input image is of gray level or color image, it will be converted to binary image and then compressed.

3.1.2 Preprocessing:

Matrix of image is converted into a row logical vector so that it will be ready to apply the run length encoding.

3.1.3 Apply the run length encoding to the vector:

In this step, run length encoding is applied to the logical vector which produced from the block processing stage. The output of this stage is a decimal vector representing the number of ones and the number of zeros in succession starting from the second position in the output vector where the first location is allocated to a sign in whether the input vector starts with ones or zeros. If it starts with ones, it takes zero and if it starts with zeros, takes one.

3.1.4 Separate one runs from zeros runs in two vectors:

Ones in the binary documents represent the background and zeros that represent the information and when applying the run length encoding ones produce decimal numbers greater than zeros this means the nature of numbers produced from ones differs from the nature of numbers produced from zeros, in this stage they will be separated into two matrices to increase the efficiency of the proposed data compression technique. In fact, the odd position will be separated from even position.

3.1.5 Replace maximum values by unused values:

This stage consists of many steps they will be explained in the following.

Step1: Input the integer matrix:

the output matrixes from the previous step will be the input matrixes in this step.

Step2: Calculate maximum number of bits that is needed and maximum value to represent data:

Find the histogram of integer matrix, and find how many numbers (levels) are used in integer matrix. The length of the histogram represents the number of levels; it will be called (L). Then find how many bits are need to represent (L) by using flowing formula: B=log2 (L). Where (B) is number of bits are need to represent (L).

If (B) is a fraction number, it is rounded to the larger integer. So the (ceil) function is used for rounding and the formula will be: B = ceil (log2 (L)). Then calculate maximum value to represent data (max value) by flowing formula: max value = $(2^B)-1$

Example (3.1): Let (A) input integer matrix, after calculating the histogram of A, the number of levels (the numbers used) was 280. The max bit and max value, can bd calculated as follows:

L = 280;

B = log2(280); B =8.129

There is no 8.129 bit this value is rounded to 9 by (ceil) function: -

B = ceil (8.129) = 9 bits;

The maximum value is calculated as: -
$2^9 - 1=511$.

Step3: Search for unused values (levels) that are smaller than max value

Example (3.2): Let (A) input integer matrix and the histogram information of A is represented in table (3.1), then:

NO	1	2	3	4	5	6
Gray scale value	3	4	5	7	510	1000
Number of repetitions	2000	1000	1000	2000	200	100

Table (3.1) histogram information of A

The values that were in A are 3, 4,5,7,510,1000, that mean L = 6; B = log2 (6) =2.584; *B* = ceil (2.584) = 3 bits; max value = 2^3-1 = 7 Searches for unused values that are less than 7 in A, there are four values 0,1, 2 and 6.

Step4: Check the conditions:

If maximum element in input matrix > max bit and unused numbers ≠ [], then go to next step else go to end without any change. In the example (3.2) the maximum element in input matrix is 1000 greater than 7 and unused numbers = [0, 1, 2, 6], the condition is met then go to the next step

Step5: Replace maximum number in input data by minimum number in unused values:

To complete the example (3.2) in this step the maximum value in the entered matrix will be replaced by the minimum unused value as follows:1000 replaced by 0. This process is repeated as long as the condition in the previous step is met so 510 replaced by 1.

In this case, the number 7 will be the maximum of the entered matrix, which is equal to the max value, not smaller than it, and the condition will not be met and the process will stop. The histogram information of A will be as follows:

NO	1	2	3	4	5	6
New values	0	1	3	4	5	7
Number of repetitions	100	200	2000	1000	1000	2000

Table (3.2) histogram information of matrix (A) after replacement

Step6: Calculate the size of the replaced values plus the new data and the size of the input data:

The replacement process is not suitable for small arrays, and the size of the outputs may be greater than the size of the inputs, but it is very useful for large arrays. In this step, the size of the input will be compared with the size of the outputs to ensure that, the replacement works only if there is a reduction in the size of the entered matrix.

To complete the example (3.2) and from the tables (3.1), (3.2) the size of the input and outputs are calculated as follows: A is the input matrix, number of its elements is 6300 and maximum value is 1000, this takes 10 bits so size of input will be 6300x10 = 63000 bits where maximum value after replacement is 7, this takes 3 bits so size of output will be 6300x3 = 18900 bits. And there are two values [1000, 510] every one take 10 bits replaced by another values [0, 1] every one take 1 bit, this means that, the cost of storing the replaced values costs 22 bit, which is added to the size of the outputs and the comparison is as follows: Size of input: 63000 bits; Size of outputs: 18900 +22=18922 bits; So: Size of input >> Size of outputs. In this case, go to the next step, and the outputs will be produced.

3.1.6 Minimize levels:

Take the indexes of large numbers that have few repetitions and then reduce the levels to the middle. This process produces two outputs, the first output is the indexes matrix that is sent to the delta encoding and then compression ones and then the max replace, and the second output is the new matrix after reducing the levels in middle. The size of the inputs is compared to the size of the outputs. If the output size is less than the input size, this process is repeated automatically. The steps for this stage will be illustrated by the following example:

Example (3.3): Let (A) input integer (odd position or even position) matrix after applying max replace steps and following table is representing the histogram information of A.

levels	0-7	8-15	16-31	32-63	64-127	128-137
repetitions	100000	200	100	50	70	10

Table (3-3): histogram information of A

To apply minimize levels stage the following steps are followed:

Step1: Calculate (max value), it has already been explained in the max replace stage.

L = 138; B= ceil(log2(138) = 8; max value = (2^8) -1=255.

Step2: Search for numbers > ((max value-1) /2)) subtract them from ((max value + 1) /2)) and take their indexes. (max value-1) /2 = (255-1) /2=127. From the table (3.3). there are 10 numbers > 127 let this numbers are existed in following indexes:

indexes	500	1000	3000	5000	10000	20000	40000	60000	80000	100000
numbers	128	129	130	131	132	133	134	135	136	137

Subtract this numbers from ((max bit+1) /2)) = ((255+1) /2)) =128, after Subtracting will be as following:

indexes	500	1000	3000	5000	10000	20000	40000	60000	80000	100000
numbers	0	1	2	3	4	5	6	7	8	9

0-7 8-15

And the histogram information of A will be change as following table:

levels	0-7	8-15	16-31	32-63	64-127
Number of repetitions	100008	202	100	50	70

Table (3.4) histogram information of A After Minimize levels for one iteration

Indexes matrix= [500,1000,3000,5000,10000,20000,40000,60000,80000,100000]

Step3: indexes matrix will be compressed by delta $\rightarrow$max replace. In this example the indexes matrix after compression will be as following: [500,500,2000,2000,5000,10000,20000,20000,20000,20000]. There has been no change in compression one's step (there is no ones to compression) and there has been no change in max replace part because it is not suitable for small matrixes.

Step4: calculate the size of outputs and inputs. From table (3.3) input matrix A has 100430 elements and maximum value was 137 it is take 8 bits. So size of A= 8x100430 = 803440 bits. From table (3.4) maximum value of A after minimize levels was 127 it is take 7 bits. So size of output A= 7x100430 = 703010 bits. And indexes matrix after compression has 10 elements and maximum value was 20000 it is take 15 bits. So size of indexes matrix after compression = 15x10 = 150 bits.
General size of outputs =150 + 703010= 703160 bits.

Step5: Compare the size of the input with the size of the outputs if the outputs size is smaller than input size go to step (1) and repeat until the size of the outputs will be greater than input size then go to end and give the outputs of last iteration. In this example general size of outputs (703160 bits) < size of input (803440 bits). So go to step (1) and repeat all steps for second iteration.

3.1.7 Convert all output vectors to binary and then stored.

Step1: Convert vectors to binary form and then convert them to logical row vectors.

Step2: The row vectors are connected together and the first final file is produced and stored.

Step3: Collect header information in one decimal vector.

Step4: Convert decimal header vector to binary form and then converts it to logical row vectors and it is stored in separate file. The header file key is the number of elements of the header file vector before converting from decimal into binary form.

3.2.2 Decompression parts: The decompression part reverses the compression stages completely.

4. Experimental Results:

The proposed compression technique was applied to the set of twenty testing images the average of compression time of the test images was 0.2259 second and average of decompression time was 1.4101 second. The sizes and compression ratios of RLE were obtained and compared with the sizes and compression ratios of the compressed files that were produced by the proposed compression technique. Table (4.1) show the comparison results between proposed technique and RLE.

No.	Original Size (KB)	CR of compressed files	
		RLE	Proposed technique
Image1	493	1.1650	3.2940
Image2	493	1.4522	3.9043
Image3	493	2.1052	6.8211
Image4	493	3.4050	12.0449
Image5	493	4.9310	11.3999
Image6	493	3.6974	11.9724
Image7	493	5.7435	13.9843
Image8	493	3.4026	9.6249
Image9	493	3.7526	11.8294
Image10	493	5.9728	18.5439

Image11	**493**	**5.4324**	**15.6909**
Image12	**493**	**2.8181**	**8.8516**
Image13	**493**	**4.0713**	**12.5874**
Image14	**493**	**2.2567**	**8.5316**
Image15	**493**	**4.2734**	**10.5262**
Image16	**493**	**2.8663**	**9.5036**
Image17	**493**	**3.4517**	**8.8065**
Image18	**493**	**4.2478**	**11.4998**
Image19	**493**	**4.3126**	**13.9561**
Image20	**493**	**10.4411**	**27.1576**
average	**493**	**3.9899**	**11.5265**

Result analysis: through experimental results, it was observed that the proposed technique gives a higher compression ratio (CR) than RLE.

5. Conclusions:

1- The results of the proposed technique were compared with run length encoding
2- Through experimental results, it was observed that the proposed technique gives a higher compression ratio (CR) than RLE.
3- The files that were decompressed are exactly the same as the original files, and there is no error ratio. This means that PSNR and SNR are zeros.
4- The bit rate of the proposed technique is the lowest compared to the RLE.

References:

[1] Wei-Yi Wei, An Introduction to ImageCompression.
[2] F. Semiconductor, "Using the Run Length Encoding Features on the MPC5645S, 2011.
[3] S. Gaurav Vijayvargiya and R. P. Pandey, "A Survey: Various Techniques of Image Compression, 2013.
[4] A. H. Hussein, S. Sh. Mahmud, R. J. Mohammed, Image Compression.
[5] M. Dipperstein, "Adaptive Delta Coding Discussion and Implementation, Online available: http://michael. dipperstein.com /delta /index. html
[6] S. W. Smith, "Data Compression, Sci. Eng. Guid. to Digit. Signal Process, 1997
[7] Shubham Venayak1, Sukhwinder Singh, "A Study of 2-D Image Compression Technique", International Journal of Enhanced Research in Management & Computer Applications, Vol. 3 Issue 3, March-2014
[8] Scott E Umbaugh, digital image processing and analysis, 2011.
[10] Digital Image Processing By Rafael C. Gonzalez and Richard Eugene Woods.
[11] Johnson, Stephen, Stephen Johnson on Digital Photography, 2006.
[12] Report Paula Aguilera, Comparison of different image compression formats (ECE 533 Proj