ERROR FREE IMAGE COMPRESSION USING MODIFIED DUPLICATION FREE RUN-LENGTH CODING

M.Madhulika[1], Mr. P.Sreenivasulu[2]

[1]*PG scholar, Department of ECE, PBR VITS, Kavali, Andhrapradesh, (India)*

[2]*Associate Professor, Department of ECE, PBR VITS, Kavali, Andhrapradesh, (India)*

ABSTRACT

In this paper, a modern lossless image compression algorithm using duplication free run-length coding (RLC) is proposed. To encode entropy rule-based generative coding method is proposed. Based on these rules variable length code words was generated and the resulting code words are utilized to encode image. The proposed method is the first RLC algorithm that encodes the case of two consecutive pixels of the same intensity into a single codeword, hence gaining on compression. Also, the number of occurrence (i.e., run) that can be encoded in a single codeword is infinite. When compared to the other methods this method gives better compression ratio.

Keywords*: *Image, Compression, Run Length Coding

I. INTRODUCTION

Image processing is a technique to enhance raw images received from cameras/sensors placed on satellites, space probes or pictures taken from day to day life for various applications there are two methods of processing an image.

1.1 Analog Image Processing

Analog image processing refers to the alteration of image through electrical means the best example is television image.

1.2 Digital Image Processing

In this case digital computers are used to process image the image will be converted to digital form using scanner, digitizer and then process it. The advantage is versatility, repeatability, and original data preservation.

Digital image has become a major multimedia content in recent years when an image is stored in raw as an array of pixels the size is usually large. While transferring data or to store data, image processing technique called compression technique is used .compression is a very essential tool for archiving image data transfer on the network etc. There are two compression techniques lossy and lossless. Image can be compressed so that perfect reconstruction is possible called lossless .An approximate of the original image can be regenerated called lossy.

While lossy compression can be applied to encode images captured leisure, it cannot be applied in medical, satellite imaging where the original image must be reconstructed without any lost of information

There are various encoding methods to compress an image here we are using run length coding .Run length coding is a significant lossless image compression approach and it is part of JPEG. The RLC encodes a sequence of pixels with the same intensity value by first recording the actual intensity value followed by the number of occurrence which is referred as run. For example, the sequence of pixels 9999995555555 is encoded by RLC as (9 6 5 7), where 6 and 7 are the runs for the intensity values 9 and 5, respectively. The RLC is highly efficient in encoding image with long runs of pixels with the same intensity value (such as black and white documents) but inefficient in encoding image of high spatial activity [5]. The inefficiency is due to the high variation in pixel intensity values in which case there are more *intensity-run* pairs to encode and hence increasing the file size. For example, a sequence of dissimilar pixels 1 2 3 4 5 will be encoded as (1 1 2 1 3 1 4 1 5 1) by RLC. Obviously, adding the runs increases the size) and we name this behavior of RLC as *duplication problem*.

II. CONVENTIONAL METHODS

To solve the duplication problem, two main techniques were applied in the literature to implement the flag bits. For run-length encoder applied to raster-scanned pixels. The first technique reserves a fixed codeword and treats it as a flag. The flag precedes a pixel that is followed by a count. In [6], the codeword that is associated to a pixel intensity value that does not exist (in the image) is reserved and used as the flag. If the pixel intensity levels are 256 then the 255 pixels are used and 256 pixel intensity is used as flag. However, this technique is not applicable if all 256 intensity levels appear.

The second technique reserves one bit in the 8-bit representation (eg., say the MSB) of a grayscale image and uses it as the flag bit [9], [10]. The flag bit is set to 0 if the following d it 7-bits represent a codeword of an actual intensity value, and is set to 1 if the following bits represent a run. This technique overcomes the duplication problem, but it is not applicable if the image contains more than 128 unique intensity values. In addition, the number of consecutive pixels (of the same intensity) that can be coded by one run (flag) is at most 128 (i.e., 7-bits). For the rest of the paper, we refer to this technique as Traditional RLC2 (TRLC2 hereinafter).

III. RULE-BASED GENERATIVE CODE

R1 A general expression always start with d, followed by x, and ends with y. These are referred to as *starting and ending* rule.

R2 In any codeword derived from a general expression, d can be an individual permutation, or multiple similar permutations repeated in consecutive or non-consecutive order while x and y are always single permutations.

R3 d is a specific element in the set of all possible arrangements P. In other words, $\{d\} \in P$ such that $|\{d\}| = 1$ where $|X|$ denotes the cardinality of the set X.

R4 x can assume any element in the set $X : P \setminus \{d\}$.

R5 y can assume any element in the set $Y : P \setminus \{d\}$.

The rule-based generative coding method is proposed in this section. A general expression is a combination of three groups of bits that we referred to as starting group d, middle group x and ending group y. These groups of bits are arranged so that they satisfy R1 as stated in Fig. 1. Table illustrates a few general expressions that could be produced, and the list continues endlessly. For each of the general expressions, we assign permutations of n-bits (from the set of all possible permutation P) to d, x and y to form the code words. Here, a permutation refers to one of the 2Npossible arrangements that could be achieved with n-bits. The assignments are carried out so that they satisfy R2, R3, R4 and R5 as stated in Fig. 1. In this paper, we set n = 2and hence P = {00, 01, 10, 11}. Table II (i.e., under the column of EBD) records some of the code words constructed from the first general expression of d x y in Table I.

```
1.          Let number_of_d = 1;
2.          Let position_of_d = 1;
3.     Let length_of_general_expression = 3;
4.     Let position_of_x=1;
5.     General_expression=Array[number_of_general _expression][number_of_d];
6.     for (i=0;i<number_of_d; i++)
7.     Gerenal_expression[number_of_general_expression][i] = d;
8.     if (position_of_x < length_of_general_expression – 2)
 9.      {
10.  General_expression[number_of_general_expression][length_of_general_expressio n-2]= d;
11.  General_expression [number_of_general_expression][posi tion_of_x] = x;
12.  General_expression [number_of_general _expression][number_of _d-1] = y;
13.  Number_of_general_expression =number_of _general_expression + 1;
 14.  Position_of_x = position_of _x +1;
 15  goto 7;
 16   }
 17  if (position_of   x = length_of_general_expression)
 18 {
19  general_expression[number_of_general_expression][length_of_general_expressio n-2] = x;
20  general_expression [number_of_general_expression][num ber_of_d-1] = y;
21   number_of_general_expression = number_of_general_expression + 1;
22   lengt h_of_general_expression = length_of_general_expression + 1;

23.      position_of_x =1;
 24    number_of_d = number_of_d +1;
 25     goto 5;
 26      }
```

Fig. 2.Pseudo-code for constructing the general expression

Table I: General Expressions for Deriving Codewords

1	d x y
2	d x d y
3	d d x y
4	d x d d y
5	d d x d y
6	d d d x y
7	d x d d d y
8	d d x d d y
9	d d d x d y
10	d d d d x y
11	d x d d d d y
12	d d x d d d y
13	d d d x d d y
14	d d d d x d y
15	d d d d d x y
16	d x d d d d d y
17	d d x d d d d y
18	d d d x d d d y
19	d d d d x d d y
20	d d d d d x d y
21	d d d d d d x y
22	d x d d d d d d y

After choosing specific permutation(s) for d and x, there are two options in assigning elements of Y to y to form an actual codeword. The first option is assigning the elements at the pre-encoding stage and the codeword becomes *ending bit dependent* (EBD). In this case, the groups of permutations d, x and y all together make the code words uniquely decodable. Hence, if the general expression d x y is used for deriving code words and the permutation of 00 is assigned to d, the first codeword will be (d, x, y) = (00, 01, 01). In the next codeword, values of d and x remain the same but y is set to10. The same permutations are assigned to d and x until y takes all available possible arrangements that satisfy R5 in Fig. 1.

TABLE II: EBD AND EBI CODEWORDS FOR THE GENERAL EXPRESSION d x y

General expression = d x y

EBD	EBI
00 01 01	00 01 y
00 01 10	00 10 y
00 01 11	00 11 y
00 10 01	01 00 y
00 10 10	01 10 y
00 10	01 11 y

11	
00 11 01	10 00 y
00 11 10	10 01 y
00 11 11	10 11 y
11 00 00	11 00 y
11 00 01	11 01 y
11 00 10	11 10 y

In particular, the number of assignable permutations to d is 2N (according to R3), the number of assignable permutations to x is 2N− 1 (according to R4), and the number of assignable permutations to y is 2N − 1 (according to R5). Hence, the number of EBD code words that can be derived from one particular general expression is determined by Eq. (1).

$$[2_N - 1]_2 \times 2_N \qquad (1)$$

Table II shows some examples of EBD code words generated with the general expression d x y. The second option is keeping y undefined in the pre- encoding stage and assigning a permutation to y at the encoding stage. Hence, the codeword becomes *ending bit independent* (EBI). In this case, the permutations of d and x are determined but y is left undefined temporarily. A permutation is only assigned to y at the actual encoding stage. Note that it is optional to assign any element (i.e., permutation) from Y to y in a codeword at the encoding stage, as long as the permutation satisfies R5 in Fig. 1. Note that both EBI and EBD share the same general expression. Table II list all the possible EBI code words derived from the general expression d x y. In the first codeword (d, x, y) = (00, 01, undefined). At the encoding stage, y can be assigned to one of three permutations, i.e., 01, 10, or 11, according to R5. In particular, the number of assignable permutations to d is 2N (according to R3) and the number of assignable permutations to x is 2N − 1 (according to R4) while y is undefined. Hence, the number of EBI code words that can be derived from one general expression is determined by Eq. (2).

$$[2N - 1] \times 2N \qquad (2)$$

Regardless of the permutation value assigned to y at the en- coding stage, the codeword can be uniquely decoded because each codeword is unique in at least one of the following ways:

1) The general expression used in deriving the codeword,
2) The number of repeated d in the codeword, or
3) The position of x with respect to d in the codeword.

An image is thus encoded by using combinations of the aforementioned encoding modes. The pseudo-code for the DF- RLC algorithm is shown in Fig. 3, and Table III summarizes the aforementioned encoding modes. It is obvious that the proposed coding method overcomes the duplication problem and hence we name the proposed method as *duplication free run-length coding* (DF-RLC hereinafter).

TABLE III : SUMMARY OF ENCODING MODES

Case	Syntax
I	(intensity, S1)
II	(intensity, S2)
III	(intensity, S3), following by "actual run-length"

IV. LOSSLESS IMAGE COMPRESSIONG WITH DUPLICATION-FREE RLC

In this section, we propose a lossless image compression algorithm based on the aforementioned EBD and EBI code- words. The purpose of not defining y in EBI code words is to make the code words adaptive to the requirements of a particular encoding specification, if any. For that, permutations are assigned to y according to the encoding requirements during the actual encoding stage. Here, the proposed code is applied to implement RLC and to uniquely differentiate flags in an efficient way.

The encoding stage starts by assigning code words to the intensity levels. The assignment is done according to the probability of occurrence (PoO) of each level such that the intensity levels of higher PoO are assigned to shorter code words, and vice versa. The pixels are then scanned in raster order and encoded by using EBI code words. Since the ending group y is not defined at the pre-encoding stage, y in each codeword is utilized as a flag to determine the status of the next (neighbor) codeword. In particular, the flag can be set to indicate three status, namely, S1, S2 and S3, where S1< S2 < S3. Each status is a permutation that satisfies R5.

In case the encoder reads a run of three or more pixels of the same intensity value, these pixels are encoded as a codeword that represents their intensity level. y in this codeword is set to S3 to indicate that the next codeword encodes the count of pixels in the run. Here, the count of run is encoded by using the EBD code words described in Section III, which can encode an infinite number of pixels when considering more than one general expression.

In case the encoder reads a run of two pixels, these pixels are encoded into a single codeword in which its y is set to S2, indicating that the encoded pixel intensity value is repeated twice and the next codeword encodes the intensity level of the next pixel(s). To the best of our knowledge, DF-RLC is the first RLC approach that gains a reduction (in terms of file size) from a run of only two pixels by encoding them to a single codeword. Other RLC approaches encode the pair of pixels to two code words, i.e., one codeword for the intensity value and another codeword for the count [6], [7]. Some approaches just leave the pair of pixels as they are [9], [8], [10].

Finally, in case the encoder reads a single pixel (i.e., not repeated or run is unity), this pixel is encoded by a single codeword, and its y is set to S1 to indicate that the next codeword is the intensity level of the next pixel(s).

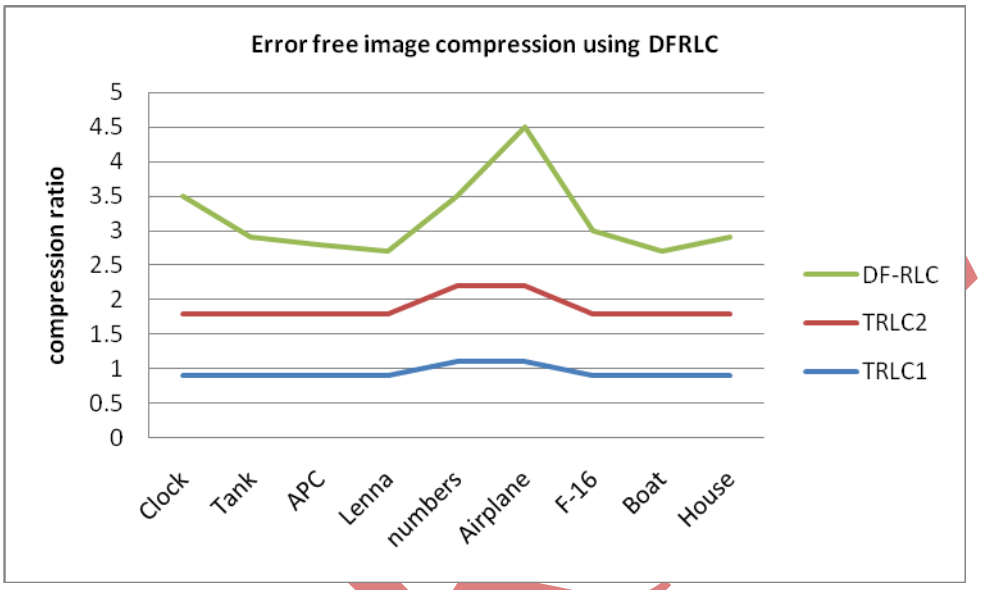
```
1   Let i= 0;
2   Let count = 0;
3   Let i = i +1;
4   Read Pixel(i).
5   If (Pixel(i) = Pixel(i-1))
6   {
7   count=count+1;
8   goto 3;
9   }
10  else
11  {
12  If (count 3)
13  {
14  Encode pixels with EBI codeword and set y to S3
15  Encode count with EBD codeword
16  }
17  If (count = 2 )
18  Encode pixels with EBI codeword and set y to S 2
19  If (count = 1 )
20  Encode pixels with EBI codeword and set y to S1
21  }
22  Goto 2 Until scanning all images pixels
```

V. EXPERIMENTAL RESULTS

TABLE IV: COMPRESSION RATIO AND HEADER SIZE FOR DF-LC AND TRADITIONAL RLC METHODS

Image name	CR of Traditional RLC 1	CR of Traditional RLC 2	CR of DF-RLC (*Global*)
Clock	0.9	0.9	1.7
Tank	0.9	0.9	1.1
APC	0.9	0.9	1
Lenna	0.9	0.9	0.9
numbers	1.1	1.1	1.3
Airplane	1.1	1.1	2.3

F-16	0.9	0.9	1.2
Boat	0.9	0.9	0.9
House	0.9	0.9	1.1



VI. CONCLUSIONS

A novel lossless image compression algorithm using duplication free run-length coding was proposed. An entropy rule-based generative coding method was proposed to produce code words that are capable of encoding both intensity level and different flag values into a single codeword. The proposed method gains compression by reducing a run of two pixels to only one codeword. Our method has no duplication problem that the traditional run-length coding algorithms usually suffer, and the number of (pixel) occurrences that can be encoded by a single run is infinite. The proposed method was com- pared to TRLC1 and TRLC2. Experimental results verified that DF-RLC outperformed TRLC1 and TRLC2 in terms of compression ratio when encoding the same image. As future work, we want to achieve higher compression ratio by taking the statistics of the image into consideration. We also want to apply the proposed rule-based generative coding method in other applications.

REFERENCES

[1] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," *Commun. ACM*, vol. 29, no. 4, pp.320–330, 1986.

[2] T. Acharya and P.-S. Tsai, *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*. Wiley-Interscience, 2004.

[3] R. Togneri and C. J. S. DeSilva, *Fundamentals of Information Theory and Coding Design*. Boca Raton, FL, USA: CRC Press, Inc., 2003.

[4] W. Zhan and A. El-Maleh, "A new collaborative scheme of test vector compression based on equal-run-length coding (erlc)," *International Conference on Computer Supported Cooperative Work in Design*, vol. 0, pp. 21–25, 2009.

[5] M. Luse, *Bitmapped Graphics Programming in C++*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer graphics: principles and practice (2nd ed.)*. Boston, MA, USA:Addison-Wesley Longman Publishing Co., Inc., 1990.

[7] Intergraph Corporation, "Intergraph raster file format guide," 1994.

[8] A. Glassner, *Adaptive Run-Length Encoding in Graphics Gems II, edited by James Arvo*. Academic Press, 1991.

[9] M. Dipperstein, "Run length encoding discussion and implementation," 2008. [Online]. Available: http://michael.dipperstein.com/rle/index.html

[10] "Run length coding, Leadtools imaging developer SDKs." [Online]. Available: www.pds site.com/LEAD/SDK/Raster/raster- compression.html