International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 6, December 2016, pp. 2836~2845
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i6.12848

New Lossless Compression Method using Cyclic Reversible Low Contrast Mapping (CRLCM)

Hendra Mesra[1], Handayani Tjandrasa[2], Chastine Fatichah[3]
[1]Department of Mathematics, Hasanuddin University, Makassar, Indonesia
[2,3]Department of Informatics, Sepuluh Nopember Institute of Technology, Surabaya, Indonesia

Article Info

Article history:

Received Aug 03, 2016
Revised Nov 04, 2016
Accepted Nov 18, 2016

Keyword:

Cyclic manner
Lossless compression
Queue
Recursive indexing
RLCM

ABSTRACT

In general, the compression method is developed to reduce the redundancy of data. This study uses a different approach to embed some bits of datum in image data into other datum using a Reversible Low Contrast Mapping (RLCM) transformation. Besides using the RLCM for embedding, this method also applies the properties of RLCM to compress the datum before it is embedded. In its algorithm, the proposed method engages Queue and Recursive Indexing. The algorithm encodes the data in a cyclic manner. In contrast to RLCM, the proposed method is a coding method as Huffman coding. This research uses publicly available image data to examine the proposed method. For all testing images, the proposed method has higher compression ratio than the Huffman coding.

Corresponding Author:

Hendra Mesra,
Departement of Mathematics,
Hasanuddin University,
Jalan Perintis Kemerdekaan KM 10, Tamalanrea, Makassar-90245, Indonesia
Email: hendra@unhas.ac.id

1. INTRODUCTION

Data compression is a method to reduce the size of data in order to save storage space and bandwidth usage on the Internet. The development of compression method is based on the fact that there are a lot of redundancies on data. The redundancy can be: *Spatial redundancy*, correlation between neighboring pixel values in case of grey scale image; *Spectral redundancy*, correlation of color planes or spectral bands in a color image; *Temporal redundancy*, correlation between image sequences in adjacent video frames; *Coding redundancy*, utilization the probability distribution associated with the occurrence of the symbols; and *Psycho visual redundancy*, human vision does not respond with equal sensitivity to all visual information [1]. These differences cause different approaches in the compression method. The Huffman and Arithmetic coding are developed to reduce the redundancy coding using statistical model [2]. The JPEG-LS [3] and CALIC [4] are compression methods based on the relation of neighborhood pixels. Therefore, both methods reduce the spatial redundancy on an image. The JPEG compression, a lossy image compression, is developed by optimizing the psycho-visual redundancy in frequency domain.

Compression methods are developed using different approaches. There are four models that commonly used in compression methods [5]:

a. Physical model: the model is obtained based on data generation processes and empirical observation of the statistics of the data.
b. Probability model: based on probability theory to describe the data
c. Markov model: based on Markov chain theory to model the data
d. Composite model: using combination of several models in a method.

Physical model compression is commonly found in speech compression. The model using a speech generator model such as Linear Prediction Coding (LPC) model to encoding original speech and using a synthesizer to decode the speech [6]. The compression method such as Huffman and Arithmetic encoding are developed using Probability model. The Run-Length-Encoding (RLE) is an example of compression method using Markov model. Some complex compression method such as Burrow-Wheeler Compression Algorithm (BWCA) and Lempel-Ziv Markov chain Algorithm (LZMA) are examples of Composite models.

Recently, some researches in lossless compression methods commonly aim to optimize existing compression method for specific data type [7-18] or to improve the existing compression method by transforming data to other form before compression process or by combining several compression method [19-22]. One of novel researches in compression method is Asymmetric Numerical System (ANS) [23-24]. The method utilizes a finite state automation and a probability model in its algorithm.

A new approach compression method developed in [25] did not reduce redundancy in data but embedding some parts of data to other parts uses reversible watermarking technique. The study applied Reversible Contrast Mapping (RCM) transform [26] as the base of the algorithm. The other research on this field is a compression method using Reversible Low Contrast Mapping (RLCM) [27]. Both methods divide an image as blocks of data. The blocks are created by dividing the image as a square of 8×8, 16×16, or 32×32 pixels. The blocks are grouped to host blocks and watermark blocks. The watermark blocks are embedded into the host blocks in order to reduce the number of blocks. The compression scheme is shown in Figure 1.

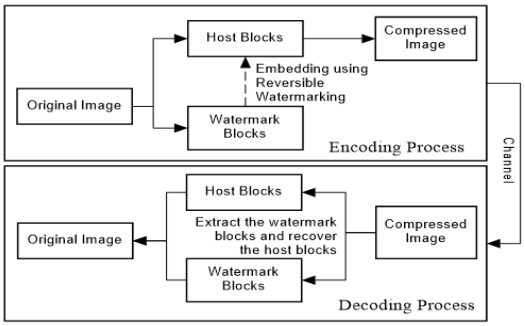


Figure 1. The Compression Scheme of RCM and RLCM [27]

On the compression scheme, the pixels on a block divide into pairs (*x*, *y*) but only pixel *y* is used for embedding data, therefore the capacity of blocks was not optimal. Since the algorithm still applied the basic algorithm such as in the reversible watermarking method. This study develops a new compression method by utilizing the RLCM and its properties. The proposed method optimizes all pixels for embedding data and does not divide the image as blocks but processes the compression on pairs of pixels directly. Therefore, the method can be applied only to one-dimension of data. Since the proposed method is an encoding method as Huffman coding, the method can be used for a coding step in a complex compression method.

2. REVERSIBLE LOW CONTRAST MAPPING

The RLCM is a simple mapping function for transforming a positive integer into another positive integer number in a domain *D* [28]. The function is defined by:

$$x' = \left\lfloor \frac{3x - y}{2} \right\rfloor \text{ and } y' = \left\lfloor \frac{3y - x}{2} \right\rfloor \quad (1)$$

The inverse function is defined by:

$$x = \left\lfloor \frac{3x' + y'}{4} \right\rfloor \text{ and } y = \left\lfloor \frac{3y' + x'}{4} \right\rfloor \quad (2)$$

The pair $(x, y) \in D$, if it satisfies the condition $0 \le \lceil (3x-y)/2 \rceil \le L$ and $0 \le \lceil (3y-x)/2 \rceil \le L$, where L is 255 in grayscale image domain.

The RLCM has a property, that is if $(x, y) \in D$ then the values of x' and y' are always an even or odd pair. The property was exploited as reversible watermarking method in [28] and as lossless image compression in [27]. In both researches, the embedding bit of watermark was performed in the Least Significant Bits (LSB) of y', whereas the LSB of x' became a control bit for extracting process. However, some pairs (x',y') cannot extract with correct results when the pairs were in the margin domain of D, that is $\lceil (3x-y)/2 \rceil = 0$, $\lceil (3x-y)/2 \rceil = L$, $\lceil (3y-x)/2 \rceil = 0$, or $\lceil (3y-x)/2 \rceil = L$. The changing of the LSB of y' can lead to $(x, y) \notin D$.

Therefore, the RLCM watermarking algorithm introduced a new y value to ensure that although the LSB of y' is changed, the pair (x, y) is still in D. The new value of y is computed by equation [27]:

$$y_{\max} = \arg\max_{y}(|x-y_0|, |x-y_1|) \quad (3)$$

where y_0 is the y value after its LSB is changed to "0" and y_1 is the y value after its LSB is changed to "1". The y_{max} is the y_0 or y_1 which causes the difference with x becomes maximum in an absolute value. Thereby, if $(x, y_{max}) \in D$, although the LSB of y' value is changed to "0" or "1" the inverse transform is still in $(x, y) \in D$.

Based on the watermark embedding algorithm in [28], there are three possible values of pairs (x,y) such as shown in Figure 2. The embeddable pairs are all pairs on A region. It can be used to embed a bit of watermark. All pairs on B region are the changeable pairs. The pairs cannot be used for embedding watermark, but the pairs must be transformed using (1). And the non-embeddable pairs are on C region. The pairs on this region can not be transformed by (1) therefore the LSB of y must be saved for recovery process.

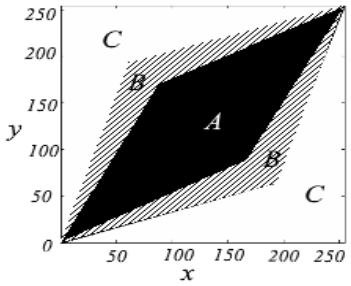


Figure 2. The Classification Pairs (x, y) in RLCM Algorithm [28]

This research will apply additional properties of RLCM. If a pair on B region and it is transformed by (1) then the result is a pair on C region. The pair cannot be used for embedding, thus both of x and y are an odd or even pair. Consequently, the average value of the pair is an integer number. This property will be exploited in this study to encode the pair with fewer bits than its original. The property is described in the following properties:

Property 1:
If $(x', y'_{\max}) \notin D$ and $x'+y' \le L$ then $t = \min(x', y')$ can be encoded in $k = \lfloor \log_2(\lfloor (x'+y')/2 \rfloor - 1) \rfloor$ bits.

Property 2:
If $(x', y'_{\max}) \notin D$ and $x'+y' > L$ then $t = L - \max(x', y')$ can be encoded in $k = \lfloor \log_2(L - \lceil (x'+y')/2 \rceil) \rfloor$ bits.

3. RESEARCH METHOD

This study develops a new approach for a lossless compression method by using RLCM that implemented in a different way from the previous research in [27]. The proposed scheme is shown in Figure 3. The proposed method uses all datum for embedding and compresses its using the RLCM properties. The method adopts the concept of Queue in the algorithm as a temporary storage for the compression result.

Besides, the proposed method utilizes Recursive Indexing in its algorithm. A Queue is a List with First In First Out (FIFO) data structure where the insertion and deletion operations are carried out at the rear end and the front end of the list respectively [29]. The insertion operation is called enqueue and the deletion operation is called dequeue. Recursive Indexing is a method to encode data with limit symbols. Let a set of symbols $\beta=\{b_0,b_1,b_2,\cdots,b_K\}$ with K+1 symbols. A number $n=mK+R$ where $n\in Z^+$ and $R = n$ mod (K+1) can be encoded using symbols in β in a sequence of β^* as defined by [30]:

$$\Im:\alpha\mapsto\beta^*$$

Thus:

$$\Im(n)=\underbrace{b_K b_K b_K\cdots b_K}_{m\text{ times}}b_R$$

This method is very well used to encode a set of $\alpha=\{a_0,a_1,a_2,\cdots,a_m\}$, where a_i are generally smaller than the K+1. Thus, if a_i<K +1 then a_i is encoded using a symbol with $\lceil\log_2(K+1)\rceil$bits. Whereas, if $a_i \geq K + 1$ then a_i is encoded using (m+1)$\lceil\log_2(K+1)\rceil$ bits.

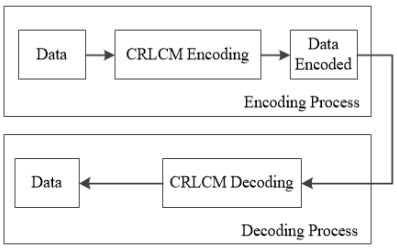


Figure 3. The Compression Process

3.1. Encoding Method

The proposed method performs the encoding process in a cyclic manner as shown in Figure 4. The same bits are dequeued from the Queue to embed to a pair (x, y) by transforming the pair using RLCM several times (iterations) until the transformed pair (x', y') is in region C on Figure 2. The number of bits embedded in a pair is the embedding capacity of the pair. Based on the RLCM properties, the average (h) of x' and y' is an integer value, therefore the average can be used as x on the next cycle. The value of x' (l) is encoded using the properties 1 or 2 of RLCM before it is enqueued to the Queue. In this research, the iteration number (i) is transformed by calculating its difference (d) to a prediction value (p) before the number is encoded using recursive indexing. Furthermore, the encoded bits are enqueued into the Queue. If the pair (x, y) is originally in region C then the remainder (r) of $\frac{x+y}{2}$ must be enqueued to the Queue. The order of enqueuing process is described in Figure 4. The detail of encoding method is described as follows:

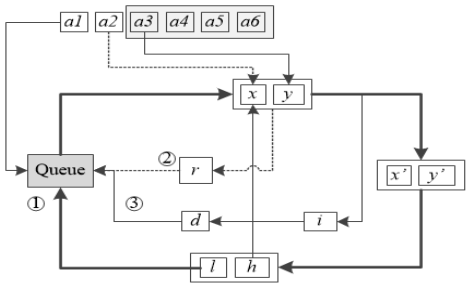


Figure 4. Encoding Algorithm

1) At the first step, the first datum is converted into bits and then put it in a queue by enqueue operation. The second (x) and third datum (y) are used for embedding data.
2) Embed some bits from the queue to pair (x, y) using the following algorithm:
   a. Calculate the value of y_{max} using (3).
   b. Set $i = 0$.
   c. If $(x, y_{max}) \in D$
      1. Calculate the value of x' and y' using (1).
      2. Calculate the y'_{max} of x' and y' using (3).
      3. Set $i = i + 1$;
      4. If $(x', y'_{max}) \in D$
         a) dequeue a bit (m) from the queue.
         b) Set LSB(y') = m.
      5. Set $x = x'$ and $y = y'$.
      6. If $(x, y_{max}) \in D$ then repeat step (i) to (v).
   d. If $(x, y_{max}) \notin D$
      1. If $i = 0$ then $r = (x+y) \bmod 2$.
      2. If $i > 0$ then $r = \varnothing$.
   e. Set $x' = x$ and $y' = y$.

   The algorithm results three values: x', y', and the iteration number (i).
3) Based on the properties 1 and 2 of RLCM, transform the x' and y' to l and h using (4) and (5) respectively.

$$l = \begin{cases} 2t & \text{if } 2t \le (x'+y') \\ 2t-1 & \text{if } 2t > (x'+y') \end{cases} \quad (4)$$

$$h = \begin{cases} \lfloor (x'+y')/2 \rfloor & \text{if } x'+y' \le L \\ \lceil (x'+y')/2 \rceil & \text{if } x'+y' > L \end{cases} \quad (5)$$

4) Compress the l value using the properties of RLCM and then enqueue it to the queue. If there is the remainder (r), enqueue it too.
5) Calculate the prediction (p) value of i using (6).

$$p = \begin{cases} \lfloor \log_2 (z/|s-z|) \rfloor + 1 & \text{if } s \ne z \text{ and } s > 0 \\ \lfloor \log_2 (s) \rfloor & \text{if } s = z \text{ and } s > 0 \\ 0 & \text{if } s = 0 \end{cases} \quad (6)$$

   where s = min (h, L-h) and z is the right neighbor of y.
   The iteration number of neighboring datum commonly alike although the embedded data is different. Therefore, the value of i can be predicted. Calculate the difference (d) of i and p using (7). And then encode the value using recursive indexing, then enqueue the value into the queue.

$$d = \begin{cases} 2|i-p|-1 & \text{if } i < p \\ 2(i-p) & \text{if } i \ge p \end{cases} \quad (7)$$

6) Transform h using $h = h\text{-}1 \bmod (L+1)$ in order that the difference to the z value becomes smaller.
7) Continue the process to the next cycle by using (h, z) as (x, y) then repeat the step 2 to 7 until all datum is processed.
8) On the last datum, the iteration number (i) is directly encoded using recursive indexing before it enqueue into the queue. The last h value is also encoded in eight bits and then enqueue to the queue.
9) The remaining bits on the queue at the last cycle are the compression result of this method.

3.2. Decoding Algorithm

The result of encoding process is all remaining bits in the queue. To extract the original data from the queue, the position of the front and rear end of the queue must be exchanged. The decoding process is the

inverse step of the encoding. Thus, the decoding process extracts the original data from the last to the first datum. Based on Figure 5, the decoding algorithm is described as follows:

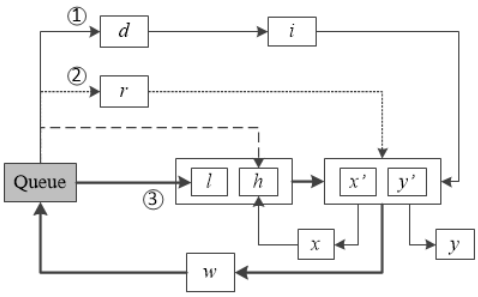


Figure 5. Decoding Algorithm

1) Dequeue the eight bits of the value *h* from the queue. At the next cycle, this value is obtained from the value of *x*.
2) Transform the value *h* using $h = h+1 \bmod (L+1)$.
3) At the first cycle, dequeue the iteration number (*i*) directly. At the next cycle, the *i* value is calculated using (8).

$$i = \begin{cases} \frac{d+1}{2} + p & \text{if } d \bmod 2 = 1 \\ \frac{d}{2} + p & \text{if } d \bmod 2 = 0 \end{cases} \quad (8)$$

The *d* value is dequeued from the queue and the *p* value is calculated using (15) on the *h* value and the *y* value from previous cycle. The number of bits is dequeued depending on the recursive indexing rules.

4) If $i = 0$ then dequeue a bit of remainder (*r*). On this condition, the value of *h* is influenced by the rounding operation in (5). To restore the original value of *x'* and *y'*, the new *H* value must be calculated using (9).

$$H = \begin{cases} 2h+r & \text{if } h \le \frac{L}{2} \text{ and } i = 0 \\ 2h-r & \text{if } h > \frac{L}{2} \text{ and } i = 0 \\ 2h & \text{if i} > 0 \end{cases} \quad (9)$$

5) Dequeue the value of *l*. The number of bits are dequeued based on the RLCM properties on value of *h*.
6) Calculate the value of *t* using (10).

$$t = \begin{cases} \frac{l}{2} & \text{if } l \bmod 2 = 0 \\ \frac{2L-i-1}{2} & \text{if } l \bmod 2 = 1 \end{cases} \quad (10)$$

7) Calculate the value of *x'* and *y'* based on the value of *t* and *h*, there are four possibilities for the values:
   a. If $h \le \frac{L}{2}$ and *l* is even number, then $x' = t$ and $y' = H - t$.
   b. If $h \le \frac{L}{2}$ and *l* is odd number, then $x' = H - t$ and $y' = t$.
   c. If $h > \frac{L}{2}$ and *l* is even number, then $x' = L - t$ and $y' = H - L + t$.
   d. If $h > \frac{L}{2}$ and *l* is odd number, then $x' = H - L + t$ and $y' = L - t$.
8) If $i > 0$ then extract the embedded bits from pair (*x', y'*) using the following algorithm:
   a. w = array();
   b. Calculate y'_{max} using (3)
   c. If $(x', y'_{max}) \notin D$ then
      1. calculate *x* and *y* using (2)
      2. $x' = x, y' = y$

d. If $(x', y'_{max}) \in D$ then
   1. $w[] = LSB(y')$
   2. $LSB(y') = LSB(x')$
   3. Calculate x and y using (2)
   4. $x' = x, y' = y$

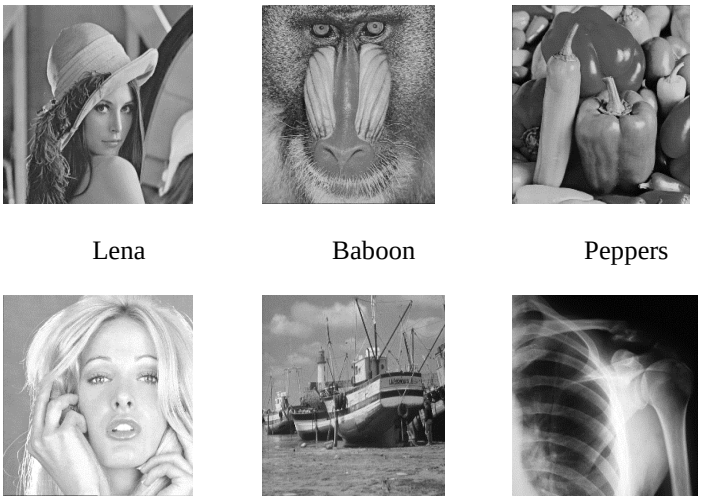
e. Repeat steps (b) to (d) in i times.

This algorithm extracts the embedded bits (w) in the pair (x', y') and restore the original value (x, y). The y value is a datum from the original data. The x value is used as h in the next cycle.

9) The last step on a cycle is to enqueue w into the queue.
10) Repeat step 2 to 9 as a new cycle to extract all of the y values.

4. RESULTS AND ANALYSIS

This research applies the proposed method on images which were used in [25] and [27] to compare the three methods. The testing images are gray scale images with size of 512×512 as shown in Figure 6. To apply the proposed method in images data, all pixels in the image is linearized by scanning the image using snake-like row major scan path. Performance of the proposed method is measured using the Compression Ratio (CR) that is the ratio of original data size to the size of the compressed data [31].



Lena Baboon Peppers

Tiffany Boat X-ray

Figure 6. Testing Images [27]

Table 1. The Comparison of Compression Ratio for the Proposed Method and Huffman Coding

No	Images	Compression ratio	
		Huffman Coding	Proposed Method
1	Lena	1.078	1.383
2	Baboon	1.090	1.117
3	Peppers	1.054	1.347
4	Tiffany	1.151	1.322
5	Boat	1.114	1.308
6	X-ray	1.031	1.640
Average		1.086	1.353

This research uses K=3 in the recursive indexing. The selection of K is based on frequency distribution of the difference (d) of iteration number (i) and its predictions (p) in (7). As example, the frequency distribution of the difference value on Lena image is shown on Figure 7. The frequency of -1, 0, and 1 values are higher than others. The value of =3 makes the encoding in Recursive Indexing using two bits or its multiples.

The testing result of the proposed method is shown on Table 1. The table also shows the compression result of proposed method and its comparison with the Huffman encoding. Figure 8 shows the comparison of the proposed method with the Huffman, the RCM, and the RLCM compression method. The figure shows that the proposed method results in higher compression ratio on all testing images except in the Baboon image.

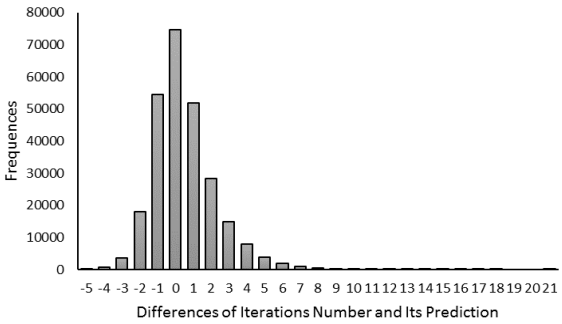


Figure 7. Histogram of the Difference of Iteration Number and its Prediction on Lena Image

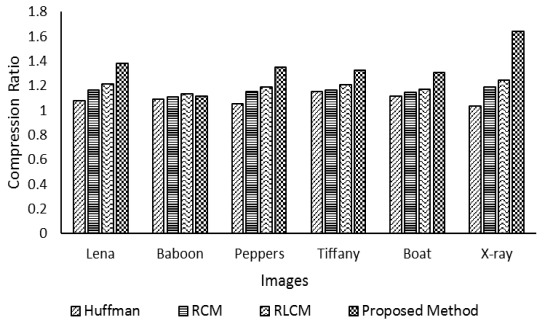


Figure 8. The Comparison of the Huffman, The RCM, The RLCM and the Proposed Method

The use of the properties 1 and 2 as a compression method in the proposed method is analyzed separately in this research. The effectiveness of the properties in compressing all pixels is shown in Table 2. The table shows that the images of Lena, Baboon, Peppers, and Boat have compression results almost the same because of the average of pixel values around the value $L/2$. The properties of RLCM have smaller compression result on the images of Tiffany and X-ray than others because their average pixel intensities are away from the $L/2$ value. According to the properties of RLCM, if the value of h is small then the value of l is encoded using fewer bits.

Besides analyzing the effectiveness of the properties of RLCM, this research also analyzes the embedding capacity of all pixels in an image. The images of Lena, Peppers, Boat, and X-ray have nearly the same of compression results and higher data embedding capacity than the images of Baboon and Tiffany. Baboon image has small of data embedding capacity because the contrast is higher than the others. Whereas Tiffany image has the average pixel values away from $L/2$ value. In this method, the pair with value about $L/2$ has higher data embedding capacity than the other positions. Although the X-ray image also has the average away from $L/2$ value, but it has contrast lower than the others.

Table 2 also shows the number of bytes are needed for encoding the iteration number using recursive indexing for all testing images. This table describes that the proposed method takes about 50% of the compression result size to save the values. It is caused by, the iteration number of a pair usually about

0-7, but the value can be very big when the values of *x* and *y* are similar and the embedded bits is similar to LSB of *y*. For example, the difference value of a pair and its prediction can achieve 21 such as shown in Figure 7. By using recursive indexing with *K*=3, the value is encoded using 30 bits. Compression result of this method is closer to the summing result of the byte residual after embedding process and the space to store the iteration number.

Table 2. The Analysis Result of the Compression Algorithm Steps

No	Images	Original Data (bytes)	Averages	Compression Result by RLCM Properties (bytes)	Total Embedding Capacity (bytes)	Residual after Embedding Process (byte)	Space for the Iteration Number (bytes)	Compression Result (byte)
1	Lena	262144	123.98	221676	129246	92497	97033	189501
2	Baboon	262144	128.75	226620	94257	132386	102266	234653
3	Peppers	262144	111.77	217958	123550	94765	99946	194623
4	Tiffany	262144	202.45	188852	91297	99007	99292	198298
5	Boat	262144	129.45	222988	123089	100069	100431	200492
6	X-ray	262144	99.42	190573	125699	65001	94749	159833

5. CONCLUSION

The proposed method is developed to optimize the data embedding capacity of the RLCM compression method by using all pixel pairs for embedding data. Besides, this method uses the other RLCM properties to encode datum using a fewer number of bits. In contrast to RLCM compression method, the CRLCM is an encoding method therefore it can be used as a coding step in other compression method.

The method uses two-step coding to encoding a datum, that is the RLCM properties and the Recursive indexing therefore the compression ratio of this method is higher than Huffman coding. The method has also higher compression ratio than the RCM and the RLCM compression methods. In this research, space for saving the iteration number brings about 50% of compression result. By changing the method for coding the iteration number, it is expected that the compression ratio is increased.

REFERENCES

[1] S. Rao and P. Bhat, "Evaluation of lossless compression techniques", in *Communications and Signal Processing (ICCSP), 2015 International Conference on*, 2015, pp. 1655–1659.
[2] D. Salomon, *Data compression: the complete reference*, Fourth Edition. Springer Science & Business Media, 2007.
[3] M.J. Weinberger, G. Seroussi, and G. Sapiro, "From LOCO-I to the JPEG-LS standard", in *1999 International Conference on Image Processing, 1999. ICIP 99. Proceedings*, 1999, vol. 4, pp. 68–72 vol.4.
[4] X. Wu and N. Memon, "CALIC-a context based adaptive lossless image codec", in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings, 1996 IEEE International Conference on*, 1996, vol. 4, pp. 1890–1893.
[5] I.M. Pu, *Fundamental Data Compression*. Butterworth-Heinemann, 2005.
[6] L. Sun, I.H. Mkwawa, E. Jammeh, and E. Ifeachor, "Speech Compression", in *Guide to Voice and Video over IP*, Springer London, 2013, pp. 17–51.
[7] D. Adjeroh, Y. Zhang, A. Mukherjee, M. Powell, and T. Bell, "DNA sequence compression using the Burrows-Wheeler Transform", in *Bioinformatics Conference, 2002. Proceedings. IEEE Computer Society*, 2002, pp. 303–313.
[8] S.K. Mukhopadhyay, S. Mitra, and M. Mitra, "A lossless ECG data compression technique using ASCII character encoding", *Comput. Electr. Eng.*, vol. 37, no. 4, pp. 486–497, Jul. 2011.
[9] T.G. Shirsat and V.K. Bairagi, "Lossless medical image compression by integer wavelet and predictive coding", *ISRN Biomed. Eng.*, vol. 2013, 2013.
[10] K. Kalajdzic, S. H. Ali, and A. Patel, "Rapid lossless compression of short text messages", *Comput. Stand. Interfaces*, vol. 37, pp. 53–59, Jan. 2015.
[11] G. Campobello, O. Giordano, A. Segreto, and S. Serrano, "Comparison of local lossless compression algorithms for Wireless Sensor Networks", *J. Netw. Comput. Appl.*, vol. 47, pp. 23–31, Jan. 2015.
[12] D. Venugopal, S. Mohan, and S. Raja, "An efficient block based lossless compression of medical images", *Opt. - Int. J. Light Electron Opt.*, vol. 127, no. 2, pp. 754–758, Jan. 2016.
[13] M. Chłopkowski and R. Walkowiak, "A general purpose lossless data compression method for GPU", *J. Parallel Distrib. Comput.*, vol. 75, pp. 40–52, Jan. 2015.

[14] B. Žalik and N. Lukač, "Chain code lossless compression using move-to-front transform and adaptive run-length encoding", *Signal Process. Image Commun.*, vol. 29, no. 1, pp. 96–106, Jan. 2014.
[15] M. Isenburg, P. Lindstrom, and J. Snoeyink, "Lossless compression of predicted floating-point geometry", *Comput.-Aided Des.*, vol. 37, no. 8, pp. 869–877, Jul. 2005.
[16] P. Praveena, "Implementation of LOCO-I Lossless Image Compression Algorithm for Deep Space Applications", *Int. J. Reconfigurable Embed. Syst. IJRES*, vol. 3, no. 1, pp. 25–30, Mar. 2014.
[17] M.F. Ukrit and G.R. Suresh, "Super-Spatial Structure Prediction Compression of Medical", *Indones. J. Electr. Eng. Inform. IJEEI*, vol. 4, no. 2, Jun. 2016.
[18] Z. Huilai, "A Complete Lattice Lossless Compression Storage Model", *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, no. 8, pp. 6332–6337, Aug. 2014.
[19] E. Syahrul, "Lossless and nearly-lossless image compression based on combinatorial transforms", Université de Bourgogne, 2011.
[20] A. Alarabeyyat *et al.*, "Lossless Image Compression Technique Using Combination Methods", *J. Softw. Eng. Appl.*, vol. 5, no. 10, p. 752, 2012.
[21] M. Effros, "PPM performance with BWT complexity: A new method for lossless data compression", in *Data Compression Conference, 2000. Proceedings. DCC 2000*, 2000, pp. 203–212.
[22] J. Abel, "Improvements to the Burrows-Wheeler compression algorithm: After BWT stages", *ACM Trans Comput. Syst.*, 2003.
[23] J. Duda, "Asymmetric numeral systems", *ArXiv Prepr. ArXiv09020271*, 2009.
[24] J. Duda, "Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding", *ArXiv Prepr. ArXiv13112540*, 2013.
[25] H. Mesra, H. Tjandrasa, and C. Fatichah, "A new approach for lossless image compression using Reversible Contrast Mapping (RCM)", presented at the International Conference of Information, Communication Technology and System (ICTS), 2014, 2014, pp. 71–76.
[26] D. Coltuc and J.-M. Chassery, "Very fast watermarking by reversible contrast mapping", *Signal Process. Lett. IEEE*, vol. 14, no. 4, pp. 255–258, 2007.
[27] Hendra Mesra, Handayani Tjandrasa, and Chastine Fatichah, "Lossless Image Compression Method Using Reversible Low Constrast Mapping (RLCM)", *JATIT J.*, vol. 86, no. 1, 2016.
[28] Hendra, "Reversible Watermarking Using Integer Transform", Thesis, Universitas Gadjah Mada, 2008.
[29] M.H. Alsuwaiyel, *Algorithms: Design Techniques and Analysis*. World Scientific, 1999.
[30] K. Sayood and S. Na, "Recursively indexed quantization of memoryless sources", *Inf. Theory IEEE Trans. On*, vol. 38, no. 5, pp. 1602–1609, 1992.
[31] S. Sahni, B.C. Vemuri, F. Chen, C. Kapoor, C. Leonard, and J. Fitzsimmons, "State of the art lossless image compression algorithms", in *IEEE Proceedings of the International Conference on Image Processing*, 1997, pp. 948–952.