New concept of universal coding using one step reversible low contrast mapping (1RLCM)

H Mesra 1, H Tjandrasa 2 and C Fatichah 2

1 Department of Mathematics, Hasanuddin University, Tamalanrea, Makassar, Indonesia
2 Department of Informatics, Sepuluh Nopember Institute of Technology, Surabaya, Indonesia

hendra@unhas.ac.id

Abstract. Universal coding was developed for compressing data that the probability distribution of a symbol is unknown. Universal coding methods encode a symbol using some bits as code based on the coding algorithm. This research introduces a new concept of universal coding by encoding two symbols using Reversible Low Contrast Mapping (RLCM) transform. The proposed method transforms a pair non negative integer to a non-negative integer and a binary number. This research also introduces a compression scheme namely Average Encoding (AE) to compress a sequence of data using the proposed coding method. This method generally yields a higher compression ratio than the Elias Delta, Elias Gamma, and Fibonacci coding when it is tested using the results of differential encoding of some testing images.

1. Introduction

Universal coding for integers in data compression is a method for coding integer data into a variable-length of bits without statistical information of data [1]. The first method of this field was introduced by Levenstein in 1968 [2]. Furthermore, the method was extended by Elias in three series, namely Gamma code, Delta code, and Omega code [3]. The other methods on this research field were the Even-Rodeh code [4], the Punctured Elias code [5]. Some coding methods such as The Golomb coding [6], Rice coding [7], the Exponential-Golomb coding, Extended Golomb coding [8], and Subexponential coding [9], a non-universal coding, were also developed from this coding approach. Some researchers used Fibonacci sequence to develop universal coding methods [10–12].

The new research in this field was the Narayana coding that was developed based on the Narayana sequences [13]. Basically, the Narayana coding was similar to Fibonacci coding but it used a different recursive equation. Other research collaborated binary coding and Fibonacci coding [1]. Some researches also developed a new number representation using (2,3)-code [14–16].

At first, the RLCM transform was introduced as a reversible watermarking method [17]. This implementation resulted that the embedding capacity of the watermarking method was sizeable, the watermark data could be extracted without the need of other information (blind watermarking), and the algorithm had low complexity. The research in [18] introduced the lossless image compression based on this property. The research implemented the watermarking method by dividing an image to a fixed size of blocks. The blocks were indexed starting from the first block until the last block. The blocks with big index numbers became the watermark and they were embedded into the blocks with small index.
numbers. Without using any pre-compression process, the compression ratio of the method was higher than the Huffman compression using the tested images. Implementation of the RLCM transform in different viewpoint was introduced as Cyclic Reversible Low Contrast Mapping (CRLCM) method [19]. The method did not use the algorithm as in the reversible watermarking method, but the CRLCM method compressed the image in a cyclic manner to extend the embedding capacity of all pixels in the image. The method is not only for image compression, but it is also applicable for compression of an integer sequence.

This research develops a new concept compression using a different approach from the previous implementations of the RLCM transform. The proposed method makes a mapping code for all pairs of positive integers into a pair of positive integers and binary code. This concept differs from common universal coding methods which map a positive integer into binary code.

2. One step Reversible Low Contrast Mapping (1RLCM) Coding

RLCM is a pair of functions for transforming a pair of positive integers into other positive integers. The RLCM transform is defined as:

\[
(x', y') = \left( \left\lfloor \frac{3x-y}{2} \right\rfloor, \left\lfloor \frac{3y-x}{2} \right\rfloor \right)
\]  

(1)

The inverse transform is defined as:

\[
(x, y) = \left( \left\lfloor \frac{3x'+y'}{4} \right\rfloor, \left\lfloor \frac{3y'+x'}{4} \right\rfloor \right)
\]  

(2)

If a pair \((x, y) \in D\) i.e. \(x' \geq 0, y'_\text{max} \geq 0, x' \leq L,\) and \(y'_\text{max} \leq L\) then the transform results in a pair of odd or even positive integers. \(L = 255\) for gray level images [17] and \(y'_\text{max}\) is defined as \(y'_\text{max} = \arg \max\{|x - y_0|, |x - y_1|\}\) where \(y_0\) is \(y\) after its LSB changed with ‘0’ and \(y_1\) is \(y\) after its LSB changed with ‘1’.

In this research, the domain of the RLCM transform is redefined i.e. the pair \((x, y) \in D\) if \(x' \geq 0\) and \(y' \geq 0\) because the maximum value of data is unknown. The proposed method divides all possible pairs \((x, y)\) in four groups as shown in Table 1.

**Table 1. The Groups of Pairs.**

| Groups | Conditions |
|--------|------------|
| A      | \(x = y\)  |
| B      | \((x, y) \in D\) and \((x', y') \in D\) |
| C      | \((x, y) \in D\) and \((x', y') \not\in D\) |
| D      | \((x, y) \not\in D\) |

For example, dividing of pair groups causes the pair \((2, 2)\) is a member of A, the pair \((3, 2)\) is a member of B because \((3, 2) \in D\) and its transform results is \((4, 2) \in D\). The pair \((4, 2)\) is a member of C because \((4, 2) \in D\) but its transform results is \((5, 1) \not\in D\). The pair \((5, 1)\) is a member of D because \((5, 1) \not\in D\). Visually, the division of pair groups is shown in Figure 1.
This proposed method is developed based on the property of RLCM transform result i.e. the pair of odd or even number. The average value of two odd or even number is an integer. The coding concept is transforming a pair \((x \in \mathbb{Z}^+, y \in \mathbb{Z}^+)\) into a pair \((r \in \mathbb{Z}^+, \text{binary code})\) using the rules shown in Table 2. The value of \(r\) is the average of pair \((x', y')\) and the binary code is a binary code of \(t = \min(x', 2r - x')\) that encoded using variable-length code depend on the \(r\) value and the pair \((x, y)\) group. If a pair is member of \(A\) then the value of \(t\) is encoded using the minimal length i.e. \(\lceil \log_2(r) \rceil + 1\). If the pair is member of \(B\), then the \(t\) value is encoded by adding a bit more from its minimal length i.e. the bit ‘0’ for identify \(x' < y'\) and the bit ‘1’ for \(x' > y'\). The pair in group \(C\) needs an extra bit from the code length of group \(B\) to distinguish it with a pair in group \(D\). This method uses bit ‘1’ to sign the group \(C\) and bit ‘0’ to sign the group \(D\). All pairs in group \(D\) is encoded by adding a bit from the code length of group \(C\) to encode the value of \(m = x' + y' \mod 2\).

Table 2. The rule of the \(r\) value and the code length of the \(t\) value.

| Groups | \(r\)          | Code length of \(t\) |
|--------|----------------|----------------------|
| A      | \(x\)          | \(\lceil \log_2(r) \rceil + 1\) |
| B      | \(x' + y'\)    | \(\lceil \log_2(r) \rceil + 2\) |
| C      | \(\frac{x' + y'}{2}\) | \(\lceil \log_2(r) \rceil + 3\) |
| D      | \(\frac{x + y}{2}\) | \(\lceil \log_2(r) \rceil + 4\) |

The rule in Table 2 is not optimal for encoding all pairs that have \(r = 0\) and \(r = 1\) because there is a group without a member. There is only a pair which has \(r = 0\) and no member of pairs is in the group \(B\) for \(r = 1\). In this research, all pairs which \(r = 0\) and \(r = 1\) are encoded using special code as shown in Table 3.

Table 3. Code for \(r = 0\) and \(r = 1\).

| \((x, y)\) | Code          |
|------------|---------------|
| (0, 0)     | (0, 0)        |
| (1, 0)     | (0, 1)        |
| (0, 1)     | (0, 0)        |
| (1, 1)     | (1, 1)        |
| (2, 0)     | (1, 0)        |
| (0, 2)     | (1, 0)        |
According to the coding rules in Table 2 and Table 3, the 1RLCM code generator is developed using the following algorithm:

If the pair \((x,y)\) is in the map in Table 3, encoding the pair uses its mapping code.
Else:
  If \((x = y)\):
    Set \(b\) is a binary code of \(x\) in \(\lceil \log_2 x \rceil + 1\) bits;
    Encode \((x,y)\) to \((x,b)\); // group A
  Else:
    If \((x,y) \in D\):
      Calculate \((x',y')\) using equation (1);
      Calculate \(r = \frac{x' + y'}{2}\);  
      Calculate \(t = \min(x',2r - x')\);
      Set \(b\) is a binary code of \(t\) in \(\lceil \log_2 r \rceil + 1\) bits;
      If \((x',y') \in D\):
        If \(x' < y'\) then encoding \((x,y)\) to \((r,0'b)\) else encoding \((x,y)\) to \((r,1'b)\); // group B
      Else:
        If \(x' < y'\) then encoding \((x,y)\) to \((r,10'b)\) else encoding \((x,y)\) to \((r,11'b)\); // group C
    Else:
      Calculate \(r = \left\lfloor \frac{x + y}{2} \right\rfloor\);
      Calculate \(t = \min(x,2r - x)\);
      Calculate \(m = (x + y) \mod 2\);
      Set \(b\) is a binary code of \(t\) in \(\lceil \log_2 r \rceil + 1\) bits;
      If \(x \leq y\) then encoding \((x,y)\) to \((r,m0'b)\) else encoding \((x,y)\) to \((r,m1'b)\). // group D

Based on the code generator, Table 4 shows some code examples. In the example, the pair \((3,3)\) is encoded into \((3_{10}, 11_2)\) where \(r = 3\) and the ‘11’ is a binary code for \(t = 3\). No additional bits are needed for this pair because it is a member of A. The pair \((2,4)\) is encoded into \((3_{10}, 1001_2)\) because \((2,4) \in D\) and \((x',y') = (1.5) \notin D\) hence the pair is in group C with \(r = 3\) and \(t = 1\). The first bit in ‘1001’ is a code for group C, the second bit is code for \(x' < y'\) and the ‘01’ is a code for the \(t\) value.

The pair \((7,6)\) is encoded into \((7_{10}, 11100_2)\) because \((7,6) \in D\) and \((x',y') = (8,6) \in D\). The pair is in group B therefore no bit is added for coding this group. The first bit in ‘1110’ is a code for \(x' > y'\) and the ‘110’ is code for \(t = 6\). The pair \((21,1000)\) is encoded into \((511_{10}, 10000000101001_2)\). The pair is in group D and the \(t\) value is 21. The first bit is a code for \((21 + 1000) \mod 2\), the second bit is a code for the group D and the third bit is a code for \(x' < y'\). The remaining bits are the binary code for 21.

| \((x,y)\) | Code          | Group |
|----------|---------------|-------|
| \((3,3)\) | \((3_{10}, 11_2)\) | A     |
| \((2,4)\) | \((3_{10}, 1001_2)\) | C     |
| \((5,1)\) | \((3_{10}, 00101_2)\) | D     |
| \((7,6)\) | \((7_{10}, 1110_2)\) | B     |
| \((50,100)\) | \((75_{10}, 100011001_2)\) | C     |
| \((21,1000)\) | \((511_{10}, 100000010101_2)\) | D     |

The decoding process is taken from the rear to the front of the code sequence. The first step is calculating the number bits that was used to encode the \(t\) value according to the value of \(r\). For example, the code \((511_{10}, 100000010101_2)\) has \(r = 511\), therefore, the \(t\) value is encoded using \(\lceil \log_2 511 \rceil + 1\) bits.
1 = 9 bits. The next step is converting the last nine bits from the code sequence into an integer i.e. \( t = 000010101 \_ 2 = 21 \). Because of \( r \neq t \), the pair \((x, y)\) is not in group A. It needs some bits to get the original pair. Getting the next bit to identify the relation of \( x^{} \) and \( y^{} \). In this case, the bit is ‘0’ hence the relation is \( x^{} < y^{} \). According to this relation, the first prediction of the pair \((x^{}, y^{})\) is \((t, 2r - t) = (21,1001)\). Because of \((21,1001) \notin D\), the pair is probably in group C or D. The next bit of the sequence is ‘0’ hence the original pair is \((x, y) = (x^{}, y^{}) = (21,1000)\). In detail, the decoding algorithm is developed as follows algorithm:

If \( r = 0 \) or \( r = 1 \) decodes \((r, \text{binary code})\) based on map in Table 3
Else:
  Calculate \( k = \lfloor \log_2 r \rfloor + 1 \);
  Get and decode \( k \) bits from the rear of binary code to an integer value \((i)\);
  If \( r = t \) then \((x, y) = (t, t) \) // group A
  Else:
    Get a bit from the rear of binary code \((i)\).
    If \( i = 0 \) then \((x^{}, y^{}) = (t, 2r - t) \) else \((x^{}, y^{}) = (2r - t, t)\);
    If \((x^{}, y^{}) \in D \) then transforming \((x^{}, y^{})\) to \((x, y)\) uses (2); // group B
    Else:
      Get a bit from the rear of binary code \((j)\).
      If \( j = 1 \) then transforming \((x^{}, y^{})\) to \((x, y)\) uses (2); // group C
      Else:
        Get a bit from the rear of binary code \((m)\);
        If \( i = 0 \) then \((x, y) = (x^{}, y^{} - m) \) else \((x, y) = (x^{} - m, y^{})\) // group D

The coding concept of the proposed method is different significantly from the general universal coding method. The length code for encoding a positive integer varies from \( \lfloor \log_2 r \rfloor + 1 \) to \( \lfloor \log_2 r \rfloor + 4 \) according to the relation of the \( x^{} \) and \( y^{} \); and the \( x^{} \) and \( y^{} \). Table 5 shows some code examples for Elias Gamma, Elias Delta, and Fibonacci coding. The three methods use fixed codes for positive integers.

### Table 5. Sample codes for Elias Gamma, Elias Delta, and Fibonacci Coding.

| Integer | Binary | Gamma Code | Delta Code | Fibonacci Code |
|---------|--------|------------|------------|----------------|
| 1       | 1      | 1          | 1          | 11             |
| 2       | 10     | 010        | 0100       | 011            |
| 3       | 11     | 011        | 0101       | 0011           |
| 4       | 100    | 00100      | 01100      | 1011           |
| 5       | 101    | 00101      | 01101      | 00011          |
| 6       | 110    | 00110      | 01110      | 10011          |
| 7       | 111    | 00111      | 01111      | 01011          |
| 8       | 1000   | 0001000    | 00100000   | 000011         |
| 9       | 1001   | 0001001    | 00100001   | 100011         |
| 10      | 1010   | 0001010    | 00100010   | 010011         |

### 3. Average Encoding (AE)

The 1RLCM coding method cannot be directly implemented to compress a sequence of data because the output of this method is a pair of integer and binary code. This research also introduces a new compression scheme for the proposed coding method namely Average Encoding. This scheme can be used for other coding method that use similar concept coding as 1RLCM. Figure 2 shows the
compression scheme. At the first compression step, the $a[0]$ value is without a pair therefore it requires an initial value ($c$) to create a pair. In this method, the initial value is $c = a[0]$. Then, this pair is encoded using 1RLCM to give ($r[0]$, binary code). The value of $r[0]$ is used with $a[1]$ as a pair in the next coding process. The compression process is continued until all symbols are encoded. The compression result of this method is all binary codes from the encoding process and the last value of $r$. The last $r$ value can be encoded using another universal coding method e.g. the Fibonacci code.

Figure 2. Encoding process on Average Encoding Scheme.

Figure 3 shows the decoding process on this scheme. The decoding process is performed from the rear to front of the sequence of compression result. The first process extracts the $r[n-1]$ value from the bit sequence using the Fibonacci code. The next process gets some bits according to the value of the $r[n-1]$ using the 1RLCM coding method to get the $a[n-1]$ and $r[n-2]$ values. The $a[n-1]$ value is an original symbol of data. The $r[n-2]$ value is used for extracting the ($r[n-3], a[n-2]$) pair. The decoding process is executed until the original data is extracted.

Figure 3. Decoding process on Average Encoding Scheme.
4. Experimental Result

In this research, the proposed method is tested to compress the grayscale images in [20]. The images are transformed using differential encoding and linearized by scanning all pixels using snake-like row major scan path before the compression process is performed. Figure 4 shows the comparison of original Lena image and its differential encoding result. The effectiveness of this coding method is measured using compression ratio i.e. the ratio of original data size to the compressed data size.

![Figure 4. Lena image and its differential encoding result.](image)

Figure 5 shows the compression ratio of the proposed method compared with the Elias Gamma, Elias Delta, and Fibonacci coding for the common images. The comparison shows that the proposed method has generally higher compression ratio than the compared methods except on the France and Washsat images.

![Figure 5. Comparison of the proposed method, Elias Gamma, Elias Delta, and Fibonacci coding on common images.](image)

The compression ratio of the proposed method is small on the France and Washsat images because the weakness of the Average Encoding to encode a data pattern with zero sequences and punctuated by some big integers. For example, the sequence of $A = [0,0,10,0,0,0,0,0]$ is encoded using 14 bits using Elias Gamma coding. But the proposed method encodes the sequence in 23 bits. The encoding process of the sequence is shown in Table 6.
Table 6. Encoding process of A using AE and 1RLCM.

| Data Pairs | $r(c = 0)$ | Binary codes |
|------------|------------|--------------|
| 0 (0,0)    | 0          | 0            |
| 0 (0,0)    | 0          | 0            |
| 10 (0,10)  | 5          | 000000       |
| 0 (5,0)    | 3          | 10101        |
| 0 (3,0)    | 2          | 10101        |
| 0 (2,0)    | 1          | 01           |
| 0 (1,0)    | 0          | 11           |
| 0 (0,0)    | 0          | 0            |

Figure 6 shows the compression ratio comparison of the proposed method and the compared methods using the medical images. In general, the proposed method has higher compression ratio than the compared methods. Despite the images contain a lot of sequences such as the A sequence, but the proposed method still has higher compression ratio because the proposed method is very effective to compress the big integers in the images.

Figure 6. Comparison of the proposed method, Elias Gamma, Elias Delta, and Fibonacci coding on medical images.

5. Conclusion

The proposed method is developed using different concept from the common method of universal coding. The proposed coding maps a pair $(x \in \mathbb{Z}^+, y \in \mathbb{Z}^+)$ to $(r \in \mathbb{Z}^+, \text{binary code})$. The coding method cannot be directly implemented to compress a sequence of data. This research also introduces a new compression scheme i.e. the Average Encoding to use the proposed coding as a compression method.

The compression result shows the compression method has higher compression ratio than the Elias Gamma, Elias Delta, and Fibonacci coding. The Average encoding has weakness to compress a specific data pattern with zero sequences and punctuated by some big integers, but the proposed method effective to encode big integers.

This is the first paper of this coding concept therefore the coding method can still be improved to generate an optimal coding method. The Average Encoding is a simple scheme to implement this coding method as a compression method. This concept can still be investigated further to get an optimal compression scheme.
6. References

[1] Nangir M, Behroozi H and Aref M R 2015 A New Recursive Algorithm for Universal Coding of Integers *Journal of Information Systems and Telecommunication* 3 (1) pp 1-6.

[2] Levenstein V E 1968 On the redundancy and delay of separable codes for the natural numbers *Probl. Cybern* 20 pp 173–179.

[3] Elias P 1975 Universal codeword sets and representations of the integers *IEEE Trans. Inf. Theory* 21 (2) pp 194–203.

[4] Even S and Rodeh M 1978 Economical Encoding of Commas Between Strings *Commun ACM*, 21 (4) pp 315–317.

[5] Fenwick P 1996 Punctured Elias codes for variable-length coding of the integers *Department of Computer Science University of Auckland New Zealand*.

[6] Solomon W G 1966 Run-Length Encodings *IEEE Trans Info Theory* 12 pp 399–401.

[7] Rice R F 1979 Some practical universal noiseless coding techniques *Tech. Rep. Jet Propulsion Laboratory Pasadena, CA*. pp 79-22

[8] Somasundaram K and Dominic S 2007 Extended golomb code for integer representation *IEEE Trans. Multimed* 9 (2) pp 239–246.

[9] Li L and Chakrabartey K 2004 On Using Exponential-Golomb Codes and Subexponential Codes for System-on-a-Chip Test Data Compression *J. Electron. Test*. 20 (6) pp 667–670.

[10] Apostolico A and Fraenkel A 1987 Robust transmission of unbounded strings using Fibonacci representations *IEEE Trans. Inf. Theory* 33 (2) pp 238–245.

[11] Fraenkel A S and Kleinv S T 1994 Robust universal complete codes for transmission and compression *Discrete Appl. Math* 64 (1) pp 31–55.

[12] Thomas J H 2007 Variations on the Fibonacci universal code’ *ArXiv Prepr. Cs0701085*.

[13] Kirthi K and Kak S 2014, The Narayana Universal Code *ArXiv160107110 Cs Math*.

[14] Anisimov AV 2009 Integer representation in the mixed base (2,3) *Cybern. Syst. Anal.* 45 (4) pp 503–516.

[15] Anisimov AV 2013 Prefix Encoding by Means of the (2,3)-Representation of Numbers *IEEE Trans. Inf. Theory* 59 (4) pp 2359–74.

[16] Anisimov AV and Zavadskiy I O 2014 Forward error correcting codes by means of the two-base (2,3)-numeration system 2014 *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)* pp 107–111

[17] Hendra 2008 Reversible Watermarking Using Integer Transform *Thesis Gadjah Mada University*.

[18] Mesra H, Tjadrasa H and Faticah C 2016 Lossless Image Compression Method Using Reversible Low Constrast Mapping (RLCM) *Journal of Theoretical and Applied Information Technologi* 86 (1) pp 18-25

[19] Mesra H, Tjadrasa H and Faticah C 2016 New Lossless Compression Method Using Cyclic Reversible Low Contrast Mapping (CRLCM) *Int. J. Electr. Comput. Eng* 6 (6) pp 2836-45

[20] Starosolski R 2007 Simple Fast and Adaptive Lossless Image Compression Algorithm *Softw Pr. Exper* 37 pp 65–91.