A PARTIALLY DECODABLE CODE FOR SCALABLE COMPRESSION OF SUPER HIGH DEFINITION IMAGES

Madoka Hasegawa, Shigeo Kato, Yoshifumi Yamada

Utsunomiya University
7-1-2, Yoto
Utsunomiya, Tochigi 321-8585, JAPAN

ABSTRACT

Multimedia communication systems using super high definition (SHD) images are widely desired in various communities such as medical imagery, digital museum and libraries and so on. There are, however, many problems in SHD image communication systems, because of high pixel accuracy and high resolution. We considered indispensable functions to be realized in SHD image application systems. These are summarized to three items, *i.e.*, *reversibility*, *scalability* and *progressibility*. This paper proposes a partially decodable coding method for realizing the scalability function. That is to say, when a whole image cannot be displayed on the monitor, a reduced image is displayed at first to select a region of interest. Then image data of the selected region is extracted from the code stream. For this purpose, a kind of partially decodable code should be introduced. We propose a new partially decodable coding method based on Golomb-Rice code.

1. INTRODUCTION

Various image coding techniques have been developed during the past several decades. Especially, in super high definition image (SHD image) application fields, such as extra high quality multimedia systems, medical image archival systems and digital museum, lossless coding methods are required since loss of any information is unacceptable. SHD images to be exchanged in such applications are 12 bit/pel for each color component and their resolutions are 4096 by 4096 or even larger, therefore highly sophisticated compression schemes are indispensable for their transmission. In this point of view, we already proposed indispensable functions to be realized in SHD image application systems[1]. These functions are summarized to next three points;

(1) ***Reversibility***

Many SHD image applications will not allow any degradation in image quality. In such applications, a lossless coding scheme in which the original image can be obtained with no distortion from the compressed data should be introduced.

(2) ***Scalability***

It is difficult to express the whole image in ordinary size display because the SHD image is approximately 4000 by 4000 pixels or even larger. Many users often hope to see the parts outside the displayed area of the image by scrolling the displayed partial image. For this purpose, it is needed to detect the starting bit position of any code word in the encoded sequence without sequential decoding. The concept of a scalable encoding scheme should be introduced in any SHD image coding method.

(3) ***Progressibility***

It is desirable that the outline of the image is represented at the early stage of transmission and the details of the image are built up progressively as transmission proceeds, because large amounts of data exist in SHD images. From this point of view, a hierarchical coding method is a potential technique for quick retrieval and management of large images such as SHD images.

There are many problems to realize both progressibility and scalability simultaneously because progressibility is usually realized by using a hierarchical structure such as image pyramid, whereas scalability is achieved by scrolling the image in the resolution of an original one. In this paper, we propose an efficient compression scheme for SHD images to satisfy scalability and reversibility by developing partially decodable variable length codes.

2. PARTIAL DECODABILITY OF VARIABLE LENGTH CODE

2.1 Partial decodability

In order to realize above-mentioned scalability, first, images are decompose into blocks of proper size, then, each block is needed to be independently encoded from pixels of other blocks. It is easy in the transform coding scheme, because they are intrinsically closed processes. On the other hand, in sequential processing schemes such as DPCM, it is needed to insert the raw data as a reference pixel at some intervals. It is, however, difficult to detect the code word corresponding to the reference pixel of the current block in the whole code sequence, if variable length code is exploited in entropy coding.

In FlashPix[2], this problem is solved by dividing the code sequence into blocks and these blocks are managed by a specified file structure. Though other partially decodable scheme is also proposed in [3], it is designed for parallel processing and not suitable for detection of the reference pixel of the current block. In this section, we propose a new partially decodable code to detect the starting code word of the chosen block by modifying Golomb code. In Golomb code [4], there exists only one code word which does not contain digit "0". Using this all-"1"-digits code for block detection, the starting position of the code word corresponding to the reference pixel of each block can be extracted independently from the entire code stream without sequential decoding. If this special code word is utilized as the block marker, continuous "1"-digits in code stream is confined to a certain limited length. Using this property, we can design a codebook with which the block marker can be detected in any position of the code stream without decoding from the head of the code stream. The length of the block marker is determined by the parameter m of Golomb code, which is called the order of Golomb code, and we prove that any code words of all-"1"-digits over $\lfloor \log_2 m \rfloor + 1$ bits are suitable to be the block marker, where function $\lfloor\ \rfloor$ means a round-off operation.

2.2 Golomb-Rice code and its characteristics

Golomb code has been known as an efficient code for geometric probability distribution source. When the order of Golomb code is given by power of 2, the code is called Golomb-Rice code (G-R code) [5]. A symbol n is encoded by following three steps with m-th order Golomb-Rice code.

Step-1: Represent n *modulo* m by binary number of k bits,

where k is $\lfloor \log_2 m \rfloor$.

Step-2: Put a sequence of $\lfloor n/m \rfloor$ zeros

Step-3: Put a "1" at the end to terminate the code word

Examples of G-R code of m=2,4,8 are shown in Table 1. In case of n=9 and m=4, n *modulo* m equals to 1 and $\lfloor n/m \rfloor$ is 2, so that the code word of symbol n is 01001. Thus, G-R code can be generated systematically, so that it will be used in entropy coding method of JPEG-LS[6] and JPEG-2000[7].

Table 1. Examples of Golomb-Rice code

n	m=2	m=4	m=8
0	01	001	0001
1	11	011	0011
2	001	101	0101
3	101	111	0111
4	0001	0001	1001
5	1001	0101	1011
6	00001	1001	1101
7	10001	1101	1111
8	000001	00001	00001
9	100001	01001	00101
10	0000001	10001	01001

Figure 1 shows a code tree of the G-R code of m=4. The number inside parenthesis indicates symbol value n. "0" is assigned to the left branch of the tree and "1" is assigned to the right branch of the tree. The black circles of the figure are nodes made in the *step-1*. Gray circles are nodes made in the *step-2*, and white circles are nodes made in the *step-3*. All code words of Golomb-Rice code are terminated by digit "1". We express the code word for symbol n as C(n). The code word $C(m$-1) consists of all "1" digits and it has the longest "1" run of k+1 bits at the length. For m=4, C(3) is the longest "1" run of 3 bits length. Other code words always contain "0" in the middle of the cord word. For m>2, $C(2^{k-1}$-1) has the longest "1" run of k bits on the tail of the code word, while $C(mj$-1) (j=2,3,4,...) has the longest "1" run of k bits on the head of the code word. The concatenation of $C(2^{k-1}$-1) and $C(mj$-1) makes the longest "1" run in the code sequence, excluding $C(m$-1). Thus, if the code word $C(m$-1) is never used to encode any symbol, the length of run which may appear in code sequence is limited to $2k$ bits, and run which is longer than $2k$+1 bits never exist in the coded data. These features are existed in a common variable length code such as Huffman code. G-R code is, however, especially easy to implement in the source coding systems because the code generating algorithm is more systematic and simple than other sophisticated codes.

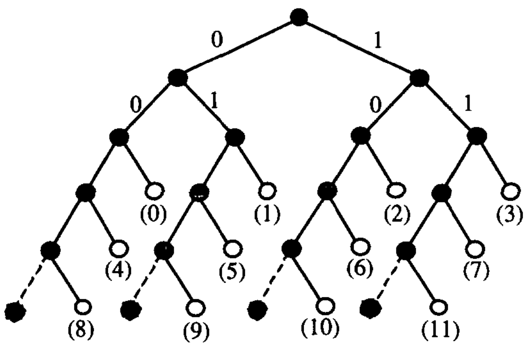

Figure 1. Code tree of Golomb-Rice code (m=4)

Reconstructing G-R code for partial decoding is described as follows; If the code word $C(m$-1) in G-R code book of order m is not used in encoding symbols, the longest "1" run in the code sequence will be $2k$ bits. Hence, we can construct $C(m$-1) by "1" of $2k$+1 bits and assign it to a block marker. However, such kind of codebook is not partially decodable, because "1" run in the code sequence exceeds $2k$+1 bits and become flexible when the tailing "1" digits of the previous code are followed by the block marker. The start position of the block marker cannot be determined without decoding the code positioned just before the block marker. For this reason, we propose a scheme to let the tailing "1" digits of previous code and block marker sum up to a fixed length of $2k$+1 bits. Figure 2 shows code examples of C(1),C(3),C(4) and C(2),C(3),C(5) in m=4. Notice that the block marker is 3 bits of "1" after C(1) and 4 bits of "1" after C(2).

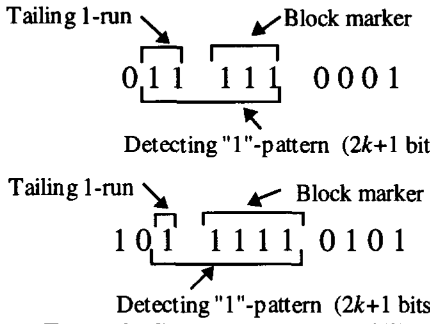

Figure 2. Concatenation of C(1)C(3)C(4) and C(2)C(3)C(5) for m=4

Hence, in both cases, the digit right after "1" of $2k$+1 (=5) bits is the beginning of next data. In the proposed scheme, location information is inserted just after the block marker. The scheme mentioned above is simple but not so efficient, because the detecting "1" pattern (including the tailing "1" of previous code and the block marker) is fixed to $2k$+1 bits regardless of the occurrence probability of the marker information. We further find that the detecting pattern can be set to arbitrary length, which is given by the shortest length of block marker l_d. When block size is larger, longer l_d should be used so that shorter code words can be assigned to symbols, and vice versa.

2.3 Proposal of modified Golomb-Rice code

This section describes the proposed modified Golomb-Rice code (MGR code). Suppose that the shortest block marker length of all "1" digits is l_d, modified Golomb-Rice code book containing all "1" block marker codes of arbitrary length can be constructed by following steps. However, because the shortest codes in Golomb-Rice codebook of order m have k+1 bits, l_d should be greater than or equal to k+1.

Step-1: Represent *n modulo m* by binary number of *k* bits.

If $k+\lfloor n/m \rfloor+1<l_d$,

Step-2 : Put a sequence of $\lfloor n/m \rfloor$ "1".

Step-3 : End the code with a "0".

If $k+\lfloor n/m \rfloor+1 \geq l_d$,

Step-2 : Putting a sequence of l_d-k-1 "1" after the binary representation of symbol *n* in step-1, put a sequence of $\lfloor n/m \rfloor-(l_d-k-1)$ "0".

Step-3 : Terminate the code word with a "1".

By this way we can assign a code of all "1" code of l_d bits or more to C(m(l_d -k)-1). If the code word $C(m(l_d$ -k)-1) is not used for coding symbols, the length of the longest tailing "1" run is l_d-1 bits, and the length of the longest heading "1" run is also l_d -1 bits. Hence, the length of "1" run in the code sequence never exceed 2l_d-2 bits. "1" run over 2l_d-1 bits length does not exist in the code sequence. Thus, the length of code word $C(m(l_d$ -k)-1) can vary according to the tailing "1" run of the previous code, so that the tailing "1" run of the previous code and C(m(l_d -k)-1) sum up to 2l_d-1 bits. When a "1" run of 2 l_d-1 bits or more is detected in any position of the code sequence, a block marker is found, and digits just after the heading 2 l_d -1 bits of "1" show the block location information. Table 2 shows the MGR codes of m=4, l_d =3 and m=4, l_d =5. Figure 3 shows the MGR tree of m=4, l_d =5, and the numbers inside parenthesis are symbol values. After the complete binary tree representing the binary part of G-R (MGR) code is constructed in step-1, the unary part of MGR code is constructed by spreading right nodes until the depth reaches l_d, and spreading left nodes thereafter as shown in Figure 3. The rightmost branch of the MGR tree is assigned to $C(m(l_d$- k)-1). Note that the length of C(m(l_d -k)-1) is flexible. Using the MGR code of m=4, l_d =3 in table 2, code sequence of symbol "-1", "block marker" and "3" is shown in the following.

011 111 1101
A

When continuous "1" digits in code sequence of $2\times 3-1=5$ bits is detected, the block marker is found so that the start position of symbol "3" can be determined as "A" in above example.

Table 2 Examples of G-R code and MGR code.

n	Data	G-R code	Data	MGR code	data	MGR code (ld=5)
0	0	001	0	001	0	000
1	-1	011	-1	011	-1	010
2	1	101	1	101	1	100
3	-2	111	Block marker	111 or 1111	-2	110
4	2	0001	-2	0001	2	0010
5	-3	0101	2	0101	-3	0110
6	3	1001	-3	1001	3	1010
7	-4	1101	3	1101	-4	1110
8	4	00001	-4	00001	4	00111
9	-5	01001	4	01001	-5	01111
10	5	10001	-5	10001	5	10111
11	-6	11001	5	11001	Block marker	11111 or 111111 or ... or 111111111
12	6	000001	-6	000001	-6	001101
13	-7	010001	6	010001	6	011101
14	7	100001	-7	100001	-7	101101
15	-8	110001	7	110001	7	111101
16	8	0000001	-8	0000001	-8	0011001
17	-9	0100001	8	0100001	8	0111001
18	9	1000001	-9	1000001	-9	1011001
...	...	......	...	......	...	....

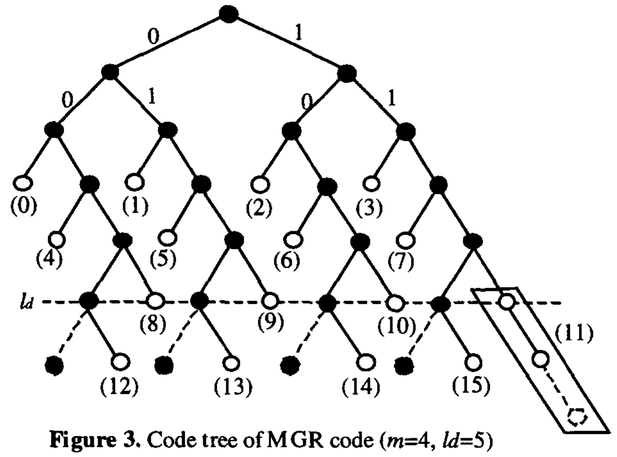

Figure 3. Code tree of MGR code (m=4, ld=5)

3. SIMULATION RESULTS

3.1 Application for SHD images

We now present our simulation results, using SHD images. Three SHD images used in the simulation are called PM001, PA022, and SA001. There are 4096 by 4096 pixels in these images and they are quantized by 12 bits/pel for R, G, and B components respectively. Y component is computed from R, G, and B components for the simulation. In the simulation, SHD images are first divided into 512 by 512 blocks and each block is coded by DPCM method, and the following equation (1) is used for prediction,

$$\hat{x} = a + b - c \tag{1}$$

where $\hat{x}$ is a prediction value, a is the left pixel value of current pixel, b is the upper pixel value and c is the upper-left pixel value, respectively.

The parameter m of G-R code can be computed by the simple equation [6]. The parameter m=32 (k=5) for PM001 and m=16 (k=4) for PA022 and SA001 are used. Table 3 shows the coding results. In this table, column "entropy" means the memoryless entropy of prediction errors, "GR" is the average code length using conventional Golomb-Rice code, and "η_{GR}" shows its efficiency. While "MGR" shows the average code length using the proposed MGR code with optimal parameter l_d, and "η_{MGR}" is its efficiency. However, in MGR, there contain extra 6 bits after each block marker, for indicating the location of current block. Table 3 shows that MGR can achieve almost the same coding efficiency as conventional Golomb-Rice code.

Table 3. Entropy and the average code length of GR code and MGR code

Image	Entropy (bits/pel)	GR (bits/pel)	η_{GR} (%)	MGR (bits/pel)	l_d (bits)	η_{MGR} (%)
PM001	6.871	7.071	97.2	7.071	35	97.2
PA022	5.861	5.900	99.3	5.901	18	99.3
SA001	6.324	6.429	98.4	6.430	25	98.4

GR : Golomb-Rice Code
MGR: Proposed partially decodable Code with detecting code word of arbitrary length

3.2 The Effect of the Code Length l_d

Figure 4 shows the relationship between the average code length of MGR and the code length of block marker l_d, using image PM001. In Figure 4, the parameter l_d= 6 so that block marker can be detected when an all "1" digits run of 11 bits or more is found. This is because all "1" digits codes in Modified Golomb-Rice code book are assigned for block marker and the encoder can select the proper length code to fill up an all "1" digits run to 11 bits according to the tailing "1" digits of previous code. Immediately after these heading 11 bits "1" digits is the block location information. Figure 4 shows that average code length increases with decrease of the block size. This is due to the increase of block marker bits and block effects. Figure 4 also shows that average code length decreases when l_d increases, but when l_d is larger than 8, the variation of l_d has little affect to coding efficiency. Because the parameter l_d of MGR can be set to arbitrary length corresponding to the block size, l_d given effective coding rate can be selected, i.e., when block size is rather small, small value of l_d is recommended to use, and vice versa.

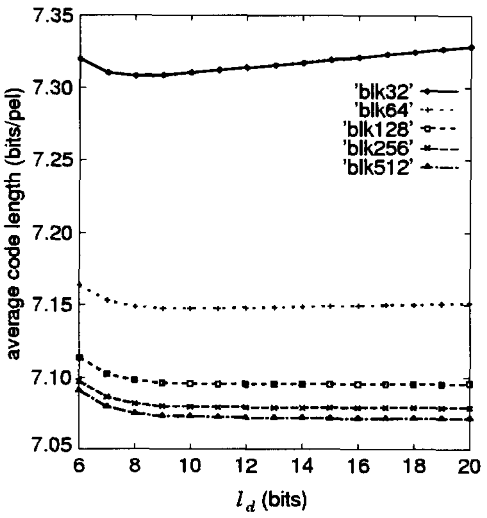

Figure 4. The effect of the code length l_d

4. CONCLUSIONS

In this paper, by assigning all "1" digits codes in Golomb-Rice code book to block marker and using the property of Golomb-Rice code, a partially decodable coding method was proposed. When the proposed code is applied to SHD images, almost the same coding efficiency can be obtained, compared to conventional Golomb-Rice code.

5. ACKNOWLEDGEMENT

This work was supported by Japan Science Promotion's Project for Research for the Future Program (Multi-Media Advanced Information and Communication Systems) "Extra High Quality Audio-Visual System for Creation of Future A-V Works and Two-Way Presence Communication (JSPS-RFTE 97P00601)".

6. REFERENCES

[1] M. Hasegawa, S. Kato and Y. Yamada, "A lossless compression scheme for super high definition images," Proceeding of PCS'97, Berlin, pp.761-765, September 1997.

[2] FlashPix format product web page of KODAK Digital Products : http://www.kodak.com/US/en/digital/flashPix/

[3] K. Iwata, T. Uematsu and E. Okamoto, "Proposal of partially decodable Ziv-Lempel code," IEICE (A), Vol.J79-A, No.11, pp.1899-1906, November 1996.

[4] Solomon W. Golomb, "Run-Length encodings," IEEE trans. on Information Theory, vol. IT-12, pp.399-401, July 1966.

[5] R. F. Rice, "Some practical universal noiseless coding techniques," Tech. Rep. JPL-79-22, Jet Propulsion Laboratory, Pasadena, CA, March 1979.

[6] M. J. Weinberger, Gadiel Seroussi and Guillermo Sapiro, "LOCO-I: A low complexity context-based lossless image compression algorithm," Proceeding of Data Compression Conference, Snowbird, Utah, pp.140-149, April 1996.

[7] ISO/IEC JTC1/SC29 Committee : "JPEG2000 image coding system," ISO/IEC JTC1/SC29/WG1, N390R, March, 1997