A New Construction Algorithm for Symmetrical Reversible Variable-Length Codes from the Huffman Code

Wook-Hyun Jeong and Yo-Sung Ho

Kwangju Institute of Science and Technology (K-JIST)
1 Oryong-dong, Puk-gu, Kwangju, 500-712, Korea
{whjeong, hoyo}@kjist.ac.kr
http://vclab.kjist.ac.kr

Abstract. Variable-length codes (VLCs) improve coding performance using statistical characteristics of source symbols; however, VLCs have disastrous effects from bit errors in noisy transmission environments. In order to overcome problems with VLCs, reversible variable-length codes (RVLCs) have been introduced as one of the error resilience tools due to their error recovering capability for corrupted video streams. Still, existing RVLCs are complicated in the design and have some rooms for improvement in coding efficiency. In this paper, we propose a new design method for a symmetrical RVLC from the optimal Huffman code table. The proposed algorithm has a simpler construction process and also demonstrates an improved performance in terms of the average codeword length than other symmetrical RVLC algorithms.

1 Introduction

Most image and video coding schemes have used VLCs, such as the Huffman code [1] or the arithmetic code [2], to improve compression efficiency. However, VLCs are so sensitive to bit errors and packet losses, along with predictive coding. If there is a bit error in the VLC bitstream, we have to discard all the ensuing data until the resynchronization marker.

In recent years, the reversible variable-length code (RVLC) has been introduced in order to reduce the effect of channel errors in the compressed bitstream. In RVLC with the resynchronization marker, we can decode the bitstream both in the forward and backward directions and recover unaffected video data as much as possible from the received bitstream. Thus, RVLC has received extensive attention as one of the error resilience tools, during the development of new video coding standards, such as MPEG-4 and H.263+, which require enhanced error control capabilities for mobile applications.

RVLC can be categorized into two different classes, symmetrical and asymmetrical codes, according to their bit patterns. The symmetrical RVLC employs the same code table for decoding both in the forward and backward directions. Although the asymmetrical RVLC offers better coding efficiency than the symmetrical RVLC owing to more flexible code assignment, the asymmetrical RVLC

A. Yazici and C. Şener (Eds.): ISCIS 2003, LNCS 2869, pp. 675–682, 2003.

requires two separate coding tables. While MPEG-4 includes an asymmetrical RVLC, H.263+ employs a symmetrical RVLC [3], [4].

After Takishima *et al.* [5] proposed the first work for constructing a symmetrical RVLC from a given Huffman code, Tsai *et al.* [6] improved the symmetrical RVLC construction algorithm and reduced the average codeword length. However, these algorithms should find symmetrical codewords on a full binary Huffman tree and precalculate the number of available symmetrical codewords at each level before constructing the symmetrical RVLC. Moreover, the overall design procedure for the symmetrical RVLC requires high complexity due to adapting precalculated values at each level to the given Huffman table, and some restrictions are imposed during this adaptation process. Consequently, symmetrical codewords can be missed at some levels.

In this paper, we propose a new algorithm to construct a symmetrical RVLC based on the optimal Huffman code table. When we find symmetrical codewords on one side of the binary Huffman tree, we simplify the adaptation process to the given Huffman table. In addition, the proposed adaptation process can contribute to reducing the average codeword length since any effective symmetrical codewords cannot be missed.

2 The Proposed RVLC Algorithm

2.1 Search for Symmetrical Codewords

Fig. 1 (a) illustrates an example of the binary Huffman tree. Starting from the root node, we assign '0' and '1' to the left and right branches of the tree. The leaves of the tree are only allowed to be codewords so that the Huffman code satisfies the prefix condition. However, the allocation of '0' and '1' to the left and right branches from the root node is arbitrary. Therefore, we can obtain another Huffman code, as shown in Fig. 1 (b), that has the same average codeword length as that of Huffman code in Fig. 1 (a); this interchange of '0' and '1' is equivalent to the bit inversion operation in the original Huffman codewords.

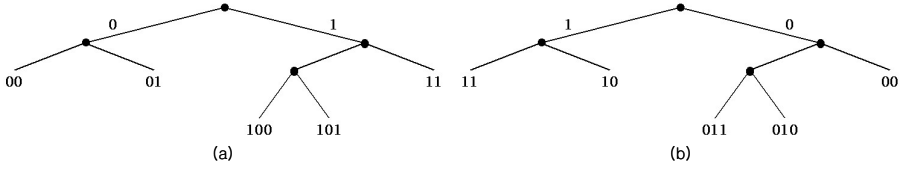

Fig. 1. Binary Huffman trees

RVLCs have to satisfy not only the prefix condition but also the suffix condition to be decoded in both the forward and backward directions instantaneously. For the symmetrical RVLC, however, the prefix condition leads to the suffix condition due to the symmetrical bit-pattern property. This symmetry property

provides another advantage. If all the bits of the chosen symmetrical codeword are inversed, we can obtain another symmetrical codeword which has the same length and satisfies both the prefix and suffix conditions. For example, in Fig. 1 (a) and (b), we can derive symmetrical codewords '11' and '101' from '00' and '010' through bit inversion, respectively.

In order to use this property, we search for symmetrical codewords in the left half region of the binary Huffman tree, especially assigned to '0'. In this region, we find $\lceil S/2 \rceil$, not S, candidates of symmetrical codewords where S is the number of given symbols and $\lceil x \rceil$ is the smallest integer larger than or equal to x. A target symmetrical RVLC for S is composed of the already selected symmetrical codewords on the left half tree and their bit-inversed versions existing on the other side of the binary tree [7].

The number $m_H(l)$ of all symmetrical codewords at level l on the left half tree is

$$m_H(l) = 2^{\lfloor (l+1)/2 \rfloor - 1} \tag{1}$$

Let $C(l)$ denote the set containing all symmetrical codewords at level l in a half region. Then,

$$C(1) = \{0\}, \qquad C(2) = \{00\} \tag{2}$$

$$C(l) = \{\ l\text{-bit codewords } for\ \ 2 \cdot e \ \ and\ \ (2^{l-1} - 2) - 2 \cdot e, \\ where\ \ e \in C(l-2)\ \ \}\ \ \text{for}\ \ l \geq 3 \tag{3}$$

Fig. 2 depicts the distribution of symmetrical candidates on a half binary tree assigned to '0' from (1)-(3) and also shows integer values corresponding to the symmetrical codewords at each level.

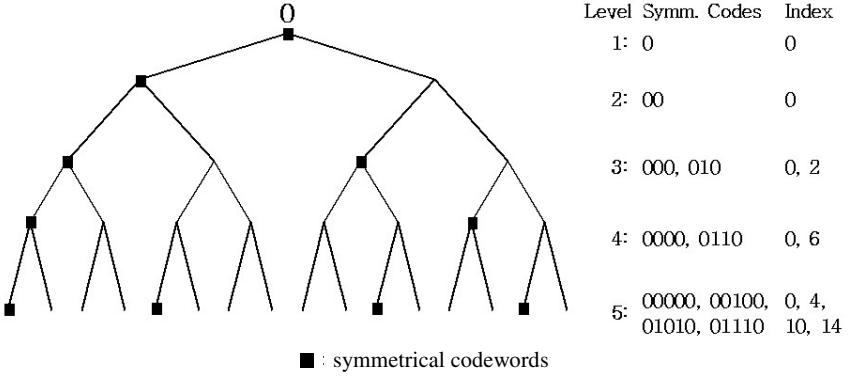

Fig. 2. Distribution of symmetrical codewords on a half binary tree

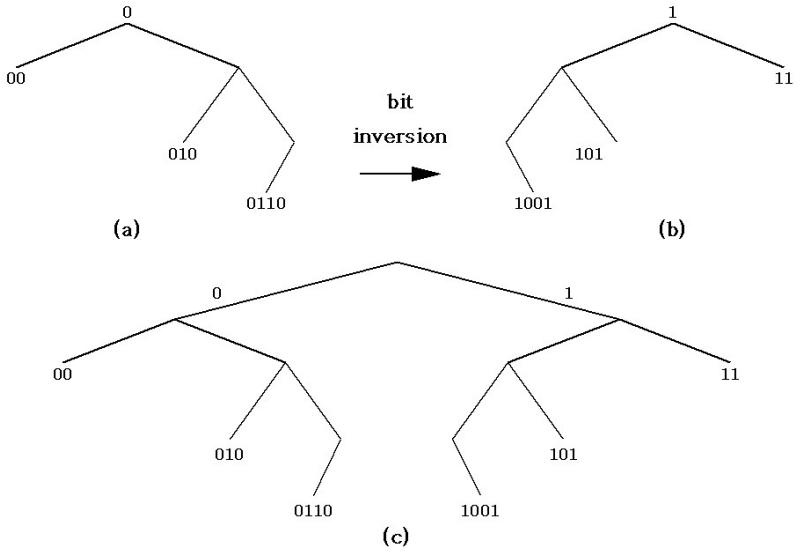

Fig. 3. (a) Unique decodablity of symmetrical codewords already chosen in the left region (b) Unique decodablity of bit inversed vergion in the right region (c) Unique decodable symmetrical RVLC

The prefix condition is the necessary and the sufficient condition for the uniquely docodable code. When we pick up symmetrical codewords on a half binary tree that satisfy the prefix condition by removing codewords that violate the prefix condition, the resulting RVLC is instantaneously decodable. If these codewords are uniquely decoded solely in one side region in Fig. 3 (a), reciprocal codewords generated by bit-inversion can also be uniquely decodable only in the other side region associated with '1' in Fig. 3 (b). When the originally chosen codewords for $\lceil S/2 \rceil$ and bit-inversed codewords for $\lceil S/2 \rceil$ are combined into the symmetrical RVLC at the last stage, the prefix is different from each other as '0' and '1'; therefore, we can obtain a symmetrical RVLC for S to be instantaneously decoded, as shown in Fig. 3 (c).

2.2 $\mathbf{Z_L}$ Adaptation

In order to reduce average length of the generated symmetrical RVLC, we need to find the most suitable RVLC to the probability distribution or the Huffman code for the given symbols, and the available codewords should not be passed from the upper level.

Since we choose symmetrical codewords in the left half region, the prefix and suffix bits of these codewords have at least one '0' bit. In addition, there are symmetrical codewords that consist of all '0' bits at each level, as shown in Fig 2. We define these codewords as $\mathbf{Z_L}$s at level L. For $\mathbf{Z_L}$, we should consider the following two situations.

1) If $\mathbf{Z_L}$ is chosen, the number of '0' bits in the prefix or the suffix of each selected symmetrical codeword in the left half region, except $\mathbf{Z_L}$, is less than L due to the elimination process. For instance, the selection of '000', $\mathbf{Z_3}$, leads that the prefix or the suffix of the other symmetrical codewords are determined to be only '0' or '00'. In other words, the choice of $\mathbf{Z_L}$ can control the number of target symmetrical RVLCs at each level.
2) Since bit-inversed symmetrical codewords are included in the target RVLC, we can obtain the codeword composed of L '1' bits for $\mathbf{Z_L}$. Thus, these two codewords are added to level L in the result. If $\mathbf{Z_L}$ is selected carefully at a proper upper level L to be suitable for the distribution of occurrence probabilities of symbols, the average length can be reduced successfully.

In this paper, we propose $\mathbf{Z_L}$ adaptation to construct a symmetrical RVLC from the optimal Huffman code table [7]. In $\mathbf{Z_L}$ adaptation, L is determined in such a way that the bit length of $\mathbf{Z_L}$ equals to L_{min}, which is defined as the bit length of the shortest codeword in the given Huffman code, opposed to L_{max}, the bit length of the largest one. $\mathbf{Z_L}$ adaptation allocates $\mathbf{Z_L}$ and L '1'bit codeword from the most probable symbol successively so that we can obtain more efficient symmetrical RVLC.

We summarize the proposed construction procedure for the symmetrical RVLC below.

1) In the left half region of the binary Huffman tree, $\mathbf{Z_L}$ is determined. The bit length of $\mathbf{Z_L}$ is selected to be the same as that of the shortest Huffman code, L_{min}. For instance, if a given Huffman code starts from a 2-bit codeword, which means that L_{min} is 2, simply '00' is selected.
2) Until symmetrical codewords are selected as many as $\lceil S/2 \rceil$, all available symmetrical codewords are chosen from the highest level. All the selection processes are followed by the elimination of codewords that violate the prefix condition.
3) Combining the already chosen symmetrical codewords and their bit-inversed codewords, we can obtain a target symmetrical RVLC to be instantaneously decoded.

Hence, the number $m(l)$ of effective symmetrical codewords at level l for $\lceil S/2 \rceil$ on a half binary tree is

$$m(l) = m_H(l) - \sum_{l} v_z(l) - \sum_{i=1}^{L-1} v_{ci}(l) \tag{4}$$

where $v_z(l)$ indicates the number of codewords causing the violation of the prefix condition at level l from the choice of $\mathbf{Z_L}$, and $v_{ci}(l)$ represents the number of codewords to be deleted from the previous selection of symmetrical codewords. In addition, L and i denote the bit length of $\mathbf{Z_L}$ and the number of '0' bits on the prefix or the suffix of the previous chosen codeword, respectively.

In Eq. (4), $v_z(l)$ is given by

(i) $v_z = 1$, when $l < 2L + 1$,
(ii) $v_z = m_H(l - 2L)$, when $2L+1 \leq l$,

and $v_{ci}(l)$ is expressed by

(i) $v_{ci} = 1$, when $2L' + 1 - i < l < 2L' + 1$,
(ii) $v_{ci}(l) = m_H(l - 2L')$, when $2L'+1 \leq l$,

where L' is the bit length of the already selected symmetrical codeword at the upper level.

3 Experimental Results and Analysis

In order to evaluate the proposed algorithm, we compare the performance of the proposed algorithm with that of existing RVLC algorithms in terms of the computational complexity of the construction process and coding efficiency.

3.1 Complexity of the Construction Process

Conventional RVLC schemes [5], [6] construct symmetrical RVLCs based on the Huffman binary tree as many as S. After the precomputation of the number of available symmetrical codewords at each level, from L_{min} to L_{max}, they repeat the adaptation and search process, whose complexity is $O(N^2 \cdot (\sqrt{2})^N) \mid_{N=S}$.

However, the proposed scheme constructs a symmetrical RVLC on a half binary tree as many as $\lceil S/2 \rceil$. After the decision of $\mathbf{Z_L}$, we search for RVLCs until a half of S. Then, we finish the design of the code by bit inversion. The complexity of the proposed RVLC scheme is $O(N^2 \cdot (\sqrt[4]{2})^N) \mid_{N=S}$. The proposed algorithm reduces the number of symbols and the search range by half. This reduction makes a significant difference of complexity between the existing and new design algorithms, which is about $O((\sqrt[4]{2})^N) \mid_{N=S}$ or $(1.19)^S$.

3.2 Coding Performance

Table 1 lists codeword assignments for the English alphabet with symmetrical RVLCs designed by Takishima's [5], Tsai's [6], and the proposed algorithms, and compares their coding performances in terms of the average codeword length. In Table 1, C1, C2, and C3 are generated by the Huffman, Takishima's, and Tsai's algorithms, respectively.

Results of C2 and C3 indicate that their algorithms do not find all available symmetrical codewords at Level 8 and Level 9 despite the existence of other effective candidates at those levels, owing to meet the restriction condition at Level 7. In addition, we observe that the unavoidable limitation occurs at Level 3, the most probable symbol location. There are four effective codewords at Level 3; nevertheless, two available codewords are omitted because the bit length of RVLC should be bounded to that of the Huffman code.

Table 1. Comparison of coding performance for the English alphabet

Occurrence Probability		Huffman C1		Takishima's algorithm : C2		Tsai's algorithm : C3		Proposed algorithm : C4	
		L	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**
E	0.14878570	3	001	3	000	3	010	3	000 ($\mathbf{Z}_3$)
T	0.09354149	3	110	3	111	3	101	3	111
A	0.08833733	4	0000	4	0110	4	0110	3	010
O	0.07245796	4	0100	4	1001	4	1001	3	101
R	0.06872164	4	0101	5	00100	4	0000	4	0110
N	0.06498532	4	0110	5	11011	4	1111	4	1001
H	0.05831331	4	1000	5	01010	5	01110	5	00100
I	0.05644515	4	1001	5	10101	5	10001	5	11011
S	0.05537763	4	1010	5	01110	5	00100	5	01110
D	0.04376834	5	00010	5	10001	5	11011	5	10001
L	0.04123298	5	00011	6	001100	6	011110	6	001100
U	0.02762209	5	10110	6	110011	6	100001	6	110011
P	0.02575393	5	10111	6	010010	6	001100	6	011110
F	0.02455297	5	11100	6	101101	6	110011	6	100001
M	0.02361889	5	11110	6	011110	7	0111110	7	0010100
C	0.02081665	5	11111	6	100001	7	1000001	7	1101011
W	0.01868161	6	011100	7	0010100	7	0010100	7	0011100
G	0.01521216	6	011101	7	1101011	7	1101011	7	1100011
Y	0.01521216	6	011110	7	0011100	7	0011100	7	0111110
B	0.01267680	6	011111	7	1100011	7	1100011	7	1000001
V	0.01160928	6	111011	7	0100010	7	0001000	8	00111100
K	0.00867360	7	1110100	7	1011101	7	1110111	8	11000011
X	0.00146784	8	11101011	8	00111100	8	01111110	8	01111110
J	0.00080064	9	111010101	9	001010100	9	011111110	8	10000001
Q	0.00080064	10	1110101000	10	0010110100	10	0111111110	9	011111110
Z	0.00053376	10	1110101001	10	1101001011	10	1000000001	9	100000001
Average length		4.15572392		4.69655649		4.60728507		**4.46463681**	

In Table 1, C4 is designed by the proposed method. Experimental results show that the average codeword length of C4 is about 5.2% and 3.2% shorter than those of C2 and C3, respectively. Moreover, C4 is composed of pairs of bit inversion, $\mathbf{Z_3}$ and '111', which are the shortest in target symmetrical RVLC, are assigned to the most probable symbols by $\mathbf{Z_L}$ adaptation, and attainable symmetrical candidates are not missed at any levels.

Table 2 shows a comparison of characteristics of the proposed symmetrical RVLC with existing symmetrical RVLCs. In Table 2, the proposed algorithm is superior to the conventional algorithm in relation to three factors: maximum codeword length, minimum codeword length, and the number of symmetrical codewords at the highest level $n_{Huff}(L_{min})$, where $n_{Huff}(i)$ is the number of Huffman codewords having a length i. These factors can affect the average codeword length and the coding performance significantly.

Table 2. Comparison of code characteristics

Type of Code	**Max. Length**	**Min. Length**	**Number of Codewords at L_{min}**
Huffman Code	L_{max}	L_{min}	$n_{Huff}(L_{min})$
Conventional Symm. RVLC	$\geq L_{max}$	L_{min}	$\leq n_{Huff}(L_{min})$
Proposed Symm. RVLC	$\leq L_{max}$	$\leq L_{min}$	$\geq n_{Huff}(L_{min})$

4 Conclusions

In this paper, we have proposed a new algorithm to design a symmetrical reversible variable-length code (RVLC) from the optimal Huffman code table. After we find symmetrical codewords in the left region of the given binary Huffman tree with $\mathbf{Z_L}$ adaptation more efficiently, we use bit inversion to generate a symmetrical RVLC. Bit inversion and $\mathbf{Z_L}$ adaptation resolve the variation problem and simplify the construction process, efficiently reducing the average codeword length. Experimental results demonstrate that the proposed RVLC algorithm improves the performance over existing RVLC algorithms.

Acknowledgments. This work was supported in part by Kwangju Institute of Science and Technology, in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultra-Fast Fiber-Optic Networks (UFON) Research Center at K-JIST, and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK21) project.

References

1. Huffman, D. : A method for the construction of minimum redundancy codes, Proc. Inst. Radio Engr., Vol. 40, Sept. (1952) 1098–1101
2. Rissanen, J. J. and Langdon. Jr., G. G. : Arithmetic Coding, IBM J. Res. Develop., 23, (1979) 149–162
3. ISO/IEC 14496–2 : Information Technology – Coding of audio/video objects, Final Draft Int. Std., Part 2 : Visual, Oct. (1998)
4. ITU-T Rec. H.263 : Video coding for low bit rate communications, Annex V, (2000)
5. Takishima, Y., Wada, M., and Murakami, H. : Reversible variable length codes, IEEE Transactions on Communications, Vol. 43, Feb. (1995) 158–162
6. Tsai, C. W. and Wu, J. L. : A modified symmetrical reversible variable length code and its theoretical bounds, IEEE Transactions on Information Theory, Vol. 47, Sept. (2001) 2543–2548
7. Jeong, W. H. and Ho, Y. S. : Design of symmetrical reversible variable-length codes from the Huffman code, Picture Coding Symposium (2003) 135–138