DESIGN OF ASYMMETRICAL REVERSIBLE VARIABLE-LENGTH CODES AND THE COMPARISON OF THEIR ROBUSTNESSES

Wook-Hyun Jeong, *Young-Suk Yoon*, *and Yo-Sung Ho*

Dept. of Information and Communications, Kwangju Institute of Science and Technology (K-JIST)
1 Oryong-dong, Buk-gu, Kwangju, 500-712, Korea (Asia)
phone: +82-062-970-2258, fax: +82-062-970-2247, email: {whjeong, ysyoon, hoyo}@kjist.ac.kr
web: visual.kjist.ac.kr

ABSTRACT

The reversible variable length code (RVLC) has been proposed as one of the error resilience tools owing to recovery capability for corrupted video streams. Although the asymmetrical RVLC requires separate two code tables, it shows better coding efficiency due to more flexible codeword assignment. Still, existing design methods for asymmetrical RVLCs are inefficient. In this paper, we propose a new design algorithm for the asymmetrical RVLC using the property of the Huffman code and the average length function. In addition, comparing the robustness of RVLCs shows that proposed efficient asymmetrical RVLC does not lose error-correcting property significantly.

1. INTRODUCTION

Compressed video stream are very sensitive to bit error and packet loss. In particular, all the still image and video coding schemes have used variable-length codes (VLCs) such as the Huffman code[1] and arithmetic code[2] at the entropy coding stage, which is so weak to bit error that it can lead to loss of synchronization at the receiver, which may result in loss of several video frames, even though VLC is able to provide the high compression efficiency.

In recent days, the reversible variable-length code (RVLC) has been introduced to reduce the effect of channel errors in the coded bitstream. In RVLC with resynchronization, we can decode the bitstream both in the forward and backward directions and recover unaffected video data as much as possible from the received erroneous bitstream, while the video data between two resynchronization markers should be discarded in VLC with resynchronization.

RVLC can be categorized into two different classes; symmetrical and asymmetrical RVLC, according to their bit patterns. Although the asymmetrical RVLC requires two separate coding tables, it offers better coding efficiency than the symmetrical RVLC due to more flexible code assignment. While MPEG-4 includes an asymmetrical RVLC, H.263+ employs a symmetrical RVLC[3, 4].

Takishima *et al.*[5] proposed the first work which specified the design method for asymmetrical RVLCs based on a given Huffman code to make their average codeword length close to that of the optimal Huffman code. Tsai *et al.*[6] improved this algorithm and reduced the average codeword length. Recently, Tseng *et al.*[7] introduced a new design method using an exhaustive search scheme with a bounding function for the symmetrical RVLC. Lin *et al.* [8] extended Tseng's method to the asymmetrical RVLC.

However, these existing design algorithms show some room for improvement in coding efficiency. Both Takishima's and Tsai's algorithms can miss available reversible codeword candidates in spite of the existence of those codewords. In both Tseng's and Lin's algorithms, since the basic assumption is that the sum of local optimization can lead to the global optimization, which is not always true, naturally their proposed backtracking algorithms are limited. In addition, this scheme is not clear in many critical factors addition, this scheme is not clear in many critical factors to design a RVLC, such as the starting bit length and codeword selection mechanisms.

In this paper, we propose a new construction algorithm for asymmetrical RVLCs using properties of the Huffman code and the average length function. In order to obtain efficient asymmetrical RVLC, we employ essential information from the Huffman code. Moreover, we define and exploit the average length function that search for efficient reversible asymmetrical codewords at each level. In addition, we compare the coding performance between the existing algorithm and the proposed algorithm in terms of average codeword length and the robustness of each RVLC in terms of the error-correcting capability after passing through the normally distributed Gaussian noise.

2. PROPOSED ALGORITHM

2.1 Property of the Huffman code

While the Huffman code[1] starts from the least frequent symbol, the proposed construction procedure for RVLCs employs the top-down scheme like existing RVLC algorithms. In the top-down scheme, we starts assigning the shortest codeword to the most probable symbol and constructs a target RVLC until all symbols are allocated to reversible codewords. The best way to construct the most optimal RVLC is to choose the RVLC with the shortest average length after generating all available RVLCs; however, the number of all available RVLCs is infinite in the top-down approach.

We should determine several elements organizing a target asymmetrical RVLC, such as the shortest bit length in the RVLC, the number of RVLCs at each level, and corresponding codewords. If we apply the codeword assignment of the given Huffman code, we can concrete the region, where the more efficient asymmetrical RVLC exists.

In the optimal Huffman code, we assign the shortest codeword to the most frequent symbol, whose bit length is determined by the source distribution and the number of symbols. However, the bit length L_{min} of the shortest code is critical to build the optimal code under the given distribution. Since asymmetrical RVLCs do not concern bit-pattens of codewords, they are close to the Huffman code. Thus, we adopt a critical description L_{min} of a given Huffman code as the starting bit level of an asymmetrical RVLC.

Let the bit length vector $n(i)$ denote the number of codewords with the bit length i, and $n_{Huff}(i)$ and $n_{RVLC}(i)$ be bit length vectors of the Huffman code and RVLC, respectively. In the proposed algorithm, we determine $n_{RVLC}(l)$ and codewords at each level l using the exhaustive search method. When deciding these elements, however, we try to find the fast approach to the more optimal RVLC applying L_{min}, $n_{Huff}(L_{min})$, codeword assignments of the given Huffman code.

Usually, since $n_{RVLC}(l)$ is smaller than $n_{Huff}(l)$ al lower level l due to the suffix condition, the more efficient RVLC exist on

$n_{RVLC}(L_{min})$ larger than or equal to $n_{Huff}(L_{min})$. The maximum number of codewords at level L_{min} is $2^{L_{min}}$ and the range of $n_{RVLC}(L_{min})$ is given by :

$$n_{Huff}(L_{min}) \le n_{RVLC}(L_{min}) \le 2^{L_{min}} \quad (1)$$

Moreover, the choice of $\mathbf{Z}_L$ at the starting level is helpful. In $\mathbf{Z}_L$ adaptation, the bit length of $\mathbf{Z}_L$ is equal to L_{min}[9]. $\mathbf{Z}_L$ is composed of only L '0' bits and $\mathbf{Z}_L$, along with the codeword consisting of all '1' bits, is necessarily once selected over whole bit levels from L_{min} to L_{max}. Thus, assigning and guaranteeing $\mathbf{Z}_L$ at the average codeword length. Therefore, in the asymmetrical RVLC, the bit length of $\mathbf{Z}_L$ is L_{min}..

In case of highly skewed distributions, such as the Laplacian distribution, starting from bit level '1' or '2', the Huffman code does not assign any VLCs to several certain levels. These levels that do not have any codewords are defined as the skipped levels L_{skip}. The L_{skip} pays an important role in assuring many candidate codewords at lower levels in the Huffman code.

The solution to obtain more efficient RVLCs is related to how many codewords are allocated to each level. We separate whole levels from L_{min} to L_{max} into two levels, the upper levels and the lower levels, where the upper levels contain $\lceil S/2 \rceil$ codewords where $\lceil x \rceil$ is the smallest integer larger than or equal to x. In order to design more efficient RVLCs, we should consider following situations.

1) In the upper levels, we assign more available codewords to shorter bit levels. Simultaneously, we provide lower levels more available candidates.
2) In the lower levels, we select codewords satisfying the affix condition.

Obviously, the first idea is contractive. However, we resolve this problem using level-skipping adaptation. In the level-skipping adaptation, we do not assign any RVLC candidates to L_{skip}, where the codeword assignment of the proposed RVLCs can follow that of the Huffman code.

2.2 Average Length Function

Takishima's[5] RVLC is sensitive to the choice of codewords called the variation problem. Tsai[6] improved the variation problem using fixed codeword selection schemes based on his own codeword selection mechanism. The backtracking algorithm also seems to overcome this sensitivity. However, if the variation occurs with a particular direction and forms a useful pattern, it would be better to generate a lot of variations.

Table I. Set $R(L;M)$

Symbols	$R(3;2)$					
A	2	00	2	00	2	00
B	3	010	3	011	3	101
C	3	011	3	101	3	111
D	4	1001	4	1001	4	0110
E	4	1101	4	1110	4	1001
F	4	1110	4	1111	5	01010
G	4	1111	5	01010	5	01011

The average codeword length $\overline{L}(X)$ of a VLC of the source X is used as the measurement of coding performance and $\overline{L}(X)$ is the convex function (∪)[10].

We define a set $R(L;M)$ whose components are all RVLCs that have the same reversible codewords in upper levels over level L and $n_{RVLC}(L) = M$ as follows.

$$R(L;M) = \{RVLC \mid \Psi_{L_{min}}^{L-1} \text{ and } n_{RVLC}(L) = M\} \quad (2)$$

where Ψ_m^n is a set whose components are already chosen all reversible codewords with bit length from m to n for $m \le n$.

Table I shows an example of $R(L;M)$ containing several variations at Level 3.

In addition, we define the average length function $f_{\overline{L}}$ of which the domain denotes $n_{RVLC}(L)$ and the range is the minimum value of the average lengths in the set R for the given source distribution. In Table I, the RVLC in the first column may have the minimum value.

$$f_{\overline{L}} : n_{RVLC}(L) \to \min(\overline{L}(R)) \quad (3)$$

where $\min(\overline{L}(R))$ is the minimum value of average codeword lengths of all components in the set $R(L;M)$ for a given source distribution.

The average length function $f_{\overline{L}}$ is a convex function (∪) that has the minimum value in a certain interval. We assume that this function gradually and monotonically decreases or increases on the basis of the minimum value. For $n_{RVLC}(L_{min})$ ranging from $n_{Huff}(L_{min})$ to $2^{L_{min}}$, the average length function has the minimum value in $[n_{Huff}(L_{min}),\ 2^{L_{min}}]$. Since $f_{\overline{L}}$ is convex, local minimum in this interval related to $R(L;M)$ can represent the global minimum value solely at level L. Thus, more efficient RVLC is located on the minimum value of $f_{\overline{L}}$. For instance, $n_{RVLC}(L_{min})$ is determined to be 3 in Fig. 1. After finding $n_{RVLC}(L)$, corresponding codewords at level L are included in the set $\Psi_{L_{min}}^{L-1}$ and the set Ψ is updated as $\Psi_{L_{min}}^{L}$.

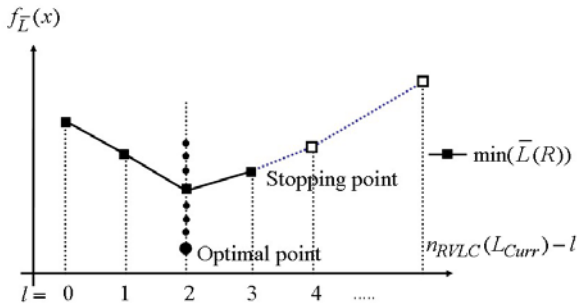

Fig. 1 The number of reversible codewords at current level

In upper level including $\lceil S/2 \rceil$ codewords, we search for appropriate codewords at each level L ($L > L_{min}$) as follows.

1) Find all available candidates at current level L_{curr} from the $\Psi_{L_{min}}^{L_{curr}-1}$. The number of candidates is $n_{RVLC}(L_{curr})$.
2) After generating $\binom{n_{RVLC}(L_{curr})}{n_{RVLC}(L_{curr})-l}$ RVLCs, calculate the average length function $f_{\overline{L}}(x)(x = n_{RVLC}(L_{curr} - l))$, where $\binom{a}{b}$ is the number of combinations of size b from a set of size a and l is increased by 1 from 0 for $0 \le l < n_{RVLC}(L_{curr})$.
3) Due to the convexity (∪) of the average length function, if the current value of $f_{\overline{L}}(x)$ is greater than that of the previous one, select the previous value of l and add corresponding codewords to the set $\Psi_{L_{min}}^{L_{curr}-1}$, as shown in Fig. 1. The number of reversible codewords at current level is

$$n_{RVLC}(L_{curr}) = n_{RVLC}(L_{curr}) - l \quad (4)$$

We summarize the proposed design procedure for asymmetrical RVLCs below.

1) Determine the starting bit level of a target RVLC and available codewords at this level, as shown in Fig. 1. The starting bit level is L_{min} that is the shortest bit length in the given Huffman code. When generating RVLCs, apply the level-skipping adaptation and $\mathbf{Z}_L$ adaptation.
2) In upper levels, search for reversible codewords at each level based on the average length function.
3) In lower levels, select all available codewords, that satisfy the affix condition, at each level until all symbols have been assigned to reversible codeword.

3. EXPERIMENTAL RESULTS

3.1 Coding Performance

Table II lists average codeword lengths of asymmetrical RVLCs for file sets from Canterbury Corpus[11] designed by Tsai's[6], Tseng's[7], Lin's[8], and the proposed algorithms. Various file sets from Canterbury Corpus are used to measure and compare the compression performance. As shown in Table II, we reduce average codeword lengths for asymmetrical RVLCs significantly over all existing methods. Overall, the average lengths of proposed asymmetrical RVLCs are about 3% shorter than those of Lin's asymmetrical RVLCs. There are skipped bit levels in Huffman codes for F2, F4, F8, F9, F10, and F11 and we have applied the level-skipping adaptation to RVLCs for these files. In Table II, we note that the proposed $\mathbf{Z}_L$ adaptation, the level-skipping adaptation, and the average length function can increase coding efficiency successfully. In addition, we observe that the proposed algorithm is superior to existing algorithms for a source that has a lot of symbols with highly skewed distributions.

Table III lists codeword assignments for the English alphabet with asymmetrical RVLCs designed by existing algorithms and the proposed algorithm, and compares their coding performances in terms of the average codeword length. The average codeword length of our RVLC is about 0.34% shorter than that of Lin's RVLC. Since any bit level is not skipped, we have not employed the level-skipping adaptation to asymmetrical RVLCs for English alphabet.

3.2 Robustness

Based on the English alphabet source, a source file with 100MBytes was generated and encoded in the Huffman code and several RVLCs. In addition, these bitstreams were transmitted over the AWGN channel with normal distribution whose mean is '0' and variance is '1'. In order to maintain bit error rate approximately as 10^{-3}, '0' bit and '1' bit were mapped to BPSK channel symbols, '–3' and '3', respectively. In the received channel string, each channel signal was converged in hard decision in which the signal larger than '0' was mapped to '1'; otherwise, '0'.

The robustness of RVLCs was measured by recovering ratio $R_{recover}$ of the redundancy over the Huffman code as follows.

$$R_{recover} = \frac{R_{correct}(\%)}{R_{redundancy}(\%)} \tag{22}$$

where $R_{correct}$ is the difference of the ration calculated from (100–$R_{discard}$) in which $R_{discard}$ is the ratio for discarded data and $R_{redundancy}$ is the ratio for overheads of bitstreams encoded in RVLCs, respectively. That is to say, $R_{recover}$ represents error-correcting ratio per the increased redundancy ratio.

Table IV lists recovering ratios of asymmetrical and symmetrical RVLCs, respectively. Herein, we can note that the proposed RVLCs do not lose the robustness significantly.

4. CONCLUSIONS

In this paper, we propose new design algorithms for more efficient asymmetrical RVLCs. In order to design more efficient RVLCs under any statistics of sources, we employ essential information from the Huffman code and the average length function. The property of the Huffman code is used for determining elementary components of RVLCs. The proposed average length function search for efficient codewords at each level effectively. We have obserbed better coding performance of the proposed schemes with various source files and English alphabet. Experimental results demonstrate that the proposed algorithm improve the coding performance over all existing algorithms successfully. In addition, we compare the error-correcting capabilities of RVLCs and we observe that more efficient RVLCs do not lose their robustness to the transmission error.

REFERENCES

[1] D. Huffman, "A method for the construction of minimum redundancy codes," *Proc. Inst. Radio Engr*., vol. 40, pp. 1098-1101, Sept. 1952.

[2] J.J. Risannen and G.G. Langdon. Jr., "Arithmetic coding," *IMB J. Res. Develop*., 23, pp. 149-162, 1979.

[3] ISO/IEC 14496-2, "Information technology – coding of audio/visual objects," Final Draft International Standard, Part 2 : Visual, Oct. 1998.

[4] ITU-T Recommendation H.263, "Video coding for low bit rate communications," Annex V, 2000.

[5] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Trans. Communications*, vol. 43, pp. 158-162, Feb. 1995.

[6] C.W. Tsai and J.L. Wu, "A modified symmetrical reversible variable length code and its theoretical bounds," *IEEE Trans. Information Theory*, vol. 47, pp. 2543-2548, Sept. 2001.

[7] H.W. Tseng and C.C. Chang, "Construction of symmetrical reversible variable length codes using backtracking," *Computer Journal*, vol. 46, no. 1, Jan. 2003.

[8] C.W. Lin, Y.J. Chuang, and J.L. Wu, "Generic construction algorithm for symmetric and asymmetric RVLCs," *Proc. IEEE International Conference on Communication Systems,* vol. 2, pp. 968-972, Nov. 2002.

[9] W.H. Jeong and Y.S. Ho, "A new construction algorithm for symmetrical reversible variable-length codes from the Huffman code," *Lecture Notes in Computer Science 2869*, pp. 675-682, Nov. 2003.

[10]T.M. Cover and J.A. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.

[11]http://corpus.canterbury.ac.nz

Table II. Coding performances of symmetrical and asymmetrical RVLCs for various source file sets

Files		No. of symbols	Huffman code	Asymmetrical RVLC		
				Tsai's method	Lin's method	Proposed method
			Average length	Average length	Average length	Average length
F1	asyoulik.txt	68	**4.84465**	5.01142	5.00954	**4.85262**
F2	Alice29.txt	74	**4.61244**	4.80326	4.68871	**4.68856**
F3	Xargs.1	74	**4.92382**	5.07334	5.16087	**4.93577**
F4	grammar.lsp	76	**4.66434**	4.85461	4.78581	**4.72571**
F5	Plabn12.txt	81	**4.57534**	4.80659	4.64910	**4.63477**
F6	lcet10.txt	84	**4.69712**	4.87868	4.74177	**4.72741**
F7	cp.html	86	**5.26716**	5.37113	5.77080	**5.28112**
F8	Fields.c	90	**5.04090**	5.26987	5.20278	**5.08843**
F9	Ptt5	159	**1.66091**	1.71814	1.70401	**1.67630**
F10	Sum	255	**5.36504**	5.49767	6.01870	**5.41683**
F11	kennedy.xls	256	**3.59337**	3.89401	3.85384	**3.78413**

Table III. Coding performances of symmetrical and asymmetrical RVLCs for the English alphabet

Occurrence Probability		Huffman code		Asymmetrical RVLC					
				Tsai's algorithm		Lin's algorithm		Proposed algorithm	
		L	codeword	L	codeword	L	codeword	L	codeword
E	0.14878570	3	001	3	000	3	000	3	000 ($\mathbf{Z}_3$)
T	0.09354149	3	110	3	111	3	100	3	101
A	0.08833733	4	0000	4	0101	3	101	3	110
O	0.07245796	4	0100	4	1010	4	0010	4	0010
R	0.06872164	4	0101	4	0010	4	0011	4	0011
N	0.06498532	4	0110	4	1101	4	0110	4	0100
H	0.05831331	4	1000	4	0100	4	0111	4	0111
I	0.05644515	4	1001	4	1011	4	1110	4	1001
S	0.05537763	4	1010	4	0110	4	1111	4	1111
D	0.04376834	5	00010	5	11001	5	01001	5	01010
L	0.04123298	5	00011	5	10011	5	01010	5	01011
U	0.02762209	5	10110	5	01110	5	01011	5	01100
P	0.02575393	5	10111	5	10001	5	11001	5	10001
F	0.02455297	5	11100	6	001100	5	11010	5	11100
M	0.02361889	5	11110	6	011110	5	11011	6	011010
C	0.02081665	5	11111	6	100001	6	010001	6	011011
W	0.01868161	6	011100	7	1001001	6	110001	6	100001
G	0.01521216	6	011101	7	0011100	7	0100001	6	111010
Y	0.01521216	6	011110	7	1100011	7	1100001	6	111011
B	0.01267680	6	011111	7	0111110	8	01000001	7	1000001
V	0.01160928	6	111011	7	1000001	8	11000001	8	10000001
K	0.00867360	7	1110100	8	00111100	9	010000001	9	100000001
X	0.00146784	8	11101011	8	11000011	9	110000001	10	1000000001
J	0.00080064	9	111010101	9	100101001	10	0100000001	11	10000000001
Q	0.00080064	10	1110101000	10	0011101001	10	1100000001	12	100000000001
Z	0.00053376	10	1110101001	10	1001011100	11	01000000001	13	1000000000001
Average length		4.15572392		4.30677804		4.18734808		**4.172804**	

Table IV. Robustness of asymmetrical RVLCs

Remarks	Huffman code	Asymmetrical RVLC		
		Tsai's algorithm	Lin's algorithm	Proposed algorithm
Average length	4.15	4.34	4.18	4.17
$R_{discard}$(%)	18.58	6.30	9.19	13.02
100– $R_{discard}$(%)	81.41	93.70	90.81	86.98
$R_{correct}$(%)	-	12.29	9.40	5.57
$R_{redundancy}$	-	3.63	0.76	0.41
$R_{recover}$	-	3.39	12.37	13.59