Design of Robust Reversible Variable-Length Codes using the Property of Free Distance

Wook-Hyun Jeong[1], Young-Suk Yoon[2], and Yo-Sung Ho[2]

[1]*Samsung Electronics Co., LTD*
416, Maetan3-dong, Paldal-gu, Suwon-si, Gyeonggi-do, Korea

[2]*Kwangju Institute of Science and Technology* (*K-JIST*)
1 *Oryong-dong, Buk-gu, Kwangju, 500-712, Korea*

wookhyon.jeong@samsung.com, {*ysyoon, hoyo*}*@kjist.ac.kr*

Abstract

In recent years, reversible variable-length codes (RVLC) have been proposed to recover correct information from erroneous bitstreams. The free distance of a binary code indicates the strength of the code to transmission errors. Since the minimum free distance of RVLC is one, we cannot assure that encoded data are little exposed. In this paper, we propose a new code design algorithm for robust RVLC by enhancing the free distance. In the proposed algorithm, we exploit properties of the Huffman code and the average length function satisfying the distance condition in order to improve the code performance.

1. Introduction

Compressed video streams are very sensitive to bit errors. Since most image and video coding standards include variable-length codes (VLC), such as Huffman codes [1] or arithmetic codes [2], they are so sensitive to bit errors that the decoder may lose synchronization.

Recently, the reversible variable-length codes (RVLC) have been introduced to reduce the effect of channel errors. In RVLC, we can decode the bitstream both in the forward and backward directions and recover correct information as much as possible from received bitstreams.

Opposed to the hard-decoding operation based on the look-up table, soft-decoding schemes based on the conditional probabilities of states and the Trellis structure have a lot of advantages in decoding corrupt VLC [3, 4]. In the Trellis approach, the free distance is used as a measure of error-correcting property of VLC.

Laković *et al.* [5] proposed a code design method for RVLC with a good error-correcting property. Although RVLC can reduce discarded data at the decoder, the minimum value of the free distance of RVLC is equal to that of VLC $d_{free} \geq 1$. Laković increased the minimum value of d_{free} as $d_{free} \geq 2$ and reduced the sensitivity of RVLC to transmission errors. However, Laković's algorithm is based on Tsai's algorithm [6], which is based on the Huffman code. Thus, existing robust RVLC algorithms have some room for improvement in code efficiency [7].

In this paper, we propose a new algorithm for robust RVLC with a good distance property. Not only affix but also free distance conditions imposes more redundancy to robust RVLC. We try to improve code efficiency of robust RVLC by exploiting properties of the Huffman code and the average length function.

2. Distance Property of RVLC

The free distance d_{free} is derived from the minimum Hamming distance between two different codewords of the same length [5]. Since the minimum free distance of VLC is 1, all VLCs are equally exposed to bit errors. In order to enhance robustness of RVLC, we increase the free distance of RVLC ($d_{free} \geq 2$). However, since the distance property imposes more redundancies on RVLC, we focus on how to reduce code inefficiency.

3. Property of the Huffman Code

3.1. Starting Bit Length of RVLC

Since the asymmetrical RVLC does not concern bit patterns of codewords, it is similar to the Huffman code. Thus, we adopt the shortest bit length L_{min} of a given

0-7803-8603-5/04/$20.00

Huffman code as the starting bit level of the asymmetrical RVLC. However, for the source data of the near-uniform distribution, $L_{min}-1$ rather than L_{min} provides better performance for the symmetrical RVLC. On the other hand, in case of the highly skewed distribution, $L_{min}+1$ can be a better choice empirically. Therefore, we should choose one of three values ($L_{min}-1$, L_{min}, and $L_{min}+1$) as the starting level.

In addition, the choice of $\mathbf{Z_L}$ at the starting level is useful [9]. $\mathbf{Z_L}$ is composed of only '0' bits and it should be selected once along with the codeword, consisting of all '1' bits, over whole bit levels.

3.2. The Number of RVLCs

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 the bit length vectors of the Huffman code and the RVLC, respectively. Since $n_{RVLC}(i)$ is usually smaller than $n_{Huff}(i)$ at a lower level due to the affix condition and the distance condition, the more efficient RVLC exists on $n_{RVLC}(L_{min})$ that is larger than or equal to $n_{Huff}(L_{min})$. The distance condition ($d_{free} \geq 2$) makes the maximum number of available candidates at level L denotes 2^{L-1}, which can be proved out using the binomial theorem. Thus, the range of $n_{RVLC}(L_{min})$ is given by :

$$n_{Huff}(L_{min}) \leq n_{RVLC}(L_{min}) \leq 2^{L_{min}-1} \qquad (1)$$

3.3. Level-Skipping Adaptation

The Huffman code for highly skewed sources does not assign any VLCs to a few levels and we define these levels as the skipped levels L_{skip}. In addition, we separate whole levels into two levels, the upper levels and the lower levels, where the upper levels contain a half of the number of symbols. In order to obtain more efficient robust RVLCs, we should assign more codewords to shorter bit levels in the upper levels. Simultaneously, we provide lower levels more available candidates. Clearly, this situation is contractive. We figure out this contraction using the level-skipping adaptation. In the level-skipping adaptation, we do not assign any reversible codewords to L_{skip}.

4. Average Length Function

We define a set $R(L;M)$ whose components are all RVLCs that have the same reversible codewords in upper levels over the level L and $n_{RVLC}(L) = M$. We define an average length function $f_{\bar{L}}$ whose domain is $n_{RVLC}(L)$ and the range is the minimum average codeword length in the set R for a given source distribution.

$$f_{\bar{L}} : n_{RVLC}(L) \rightarrow \min(\bar{L}(R)) \qquad (2)$$

The average length function is a convex function that has the minimum value in a certain interval. Through the average length function, we can select more efficient codewords at each level.

In upper levels, we select available robust codewords at each level as follows:

1) Find all available candidates at the current level L_{curr}. The distance property is $d_{free} \geq 1$.
2) Separate candidates into two sets, O_{even} and O_{odd}, according to the number of '1' bits, and select one set that has more components than the other. Moreover, select all codewords satisfying the distance condition in the other set. The distance property is $d_{free} \geq 2$.
3) Generate robust $\begin{pmatrix} n_{RVLC}(L_{curr}) \\ n_{RVLC}(L_{curr}) - l \end{pmatrix}$ RVLCs and calculate the average length function $f_{\bar{L}}$, where $\begin{pmatrix} a \\ b \end{pmatrix}$ is the number of combinations of size b from a set of size a and l is increased by 1 from 0. In other words, $f_{\bar{L}}$ is a function of l.
4) Due to the convex property, if the current value of $f_{\bar{L}}$ is larger than that of the previous one, select the previous codewords.

The proposed design procedure for robust RVLCs is as follows:

1) Determine available codewords at L_{min} for asymmetrical RVLCs applying the level-skipping adaptation and the choice of $\mathbf{Z_L}$. For symmetrical RVLCs, the starting bit level can be one of the above three values based on existence of skipped levels.
2) In upper levels, search for robust reversible codewords at each level with the average length function and the distance property.
3) In lower levels, select all available codewords satisfying both the affix condition and the distance condition at each level.

5. Experimental Results

Table I lists the average codeword lengths of robust symmetrical and asymmetrical RVLCs that are designed by Tsai's [6], Tseng's [7], Lin's [8], Laković's [5], and our proposed algorithms, for the file set from Canterbury Corpus [10].

Table I shows that our proposed robust RVLCs have higher code efficiency than Laković's robust RVLC. In particular, our robust asymmetrical RVLC has better code efficiency than Lin's RVLC, except for F2, F5, and F6, in spite of the distance condition.

Table II and Table III list the codeword assignment for the English alphabet with robust symmetrical and asymmetrical RVLCs. Since the Laković's method succeeds existing problems, restriction problems are observed at Level 3, 9, and 10. For the robust symmetrical RVLC, more efficient RVLC of a good distance property exists on $L_{min}-1$. Average lengths of the proposed robust symmetrical and asymmetrical RVLCs are about 1.2% and 2.5% shorter than those of the existing robust RVLCs.

6. Conclusions

In this paper, we have proposed how to construct more efficient error-correcting RVLCs using properties of the Huffman code and the average length function. In order to enhance the robustness of RVLCs to transmission errors, we increase the free distance of RVLCs. Although more redundancies are imposed on robust RVLCs, we significantly reduce the average codeword length of RVLCs of a good distance property. Experimental results demonstrate that the proposed algorithm has improved coding performance over existing robust RVLC algorithms successfully.

Acknowledgement

This work was supported in part by K-JIST, in part by the Ministry of Information and Communication (MIC) through the Realistic Broadcasting Research Center at K-JIST, and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK21) project.

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. Rissanen and G.G. Langdon. Jr., "Arithmetic coding," *IBM J. Res. Develop.*, 23, pp. 149-162, March 1979.

[3] S. Kaiser and M. Bystrom, "Soft decoding of variable length codes," *Proc. IEEE Int. Conf. Commum.*, pp. 1203-1207, June 2000.

[4] R. Bauer and J. Hagenauer, "On variable length codes for iterative source/channel-decoding," *Proc. IEEE Data Compression Conf.*, pp. 273-282, June 2000.

[5] L. Laković and J. Vallasenor, "On design of error-correcting reversible variable length codes," *IEEE Communs. Letters*, vol. 6, pp. 337-339, Aug. 2002.

[6] C.W. Tsai and J.L. Wu, "On constructing the Huffman-code-based reversible variable length codes," *IEEE Trans. Comm.*, vol. 49, pp. 1506-1509, Sept. 2001.

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

[8] C.W. Lin, Y.J. Chuang, and J.L. Wu, "Generic construction algorithms for symmetric and asymmetric RVLCs," *Proc. IEEE Int. Conf. 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," *LNCS 2869*, pp. 675-682, Nov. 2003.

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

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

Files		No. of symbols	Huffman code	Symmetrical RVLC				Asymmetrical RVLC			
				Tsai's method	Tseng's method	Laković's method ($d_{free} \geq 2$)	Proposed method ($d_{free} \geq 2$)	Tsai's method	Lin's method	Laković's method ($d_{free} \geq 2$)	Proposed method ($d_{free} \geq 2$)
			Average length	Average length	Average length	Average length	Average length	Average length	Average length	Average length	Average length
F1	asyoulik.txt	68	**4.84465**	5.27886	5.21025	5.41179	**5.31476**	5.01142	5.00954	5.20114	**4.99592**
F2	Alice29.txt	74	**4.61244**	5.01398	4.93155	5.21075	**5.02562**	4.80326	4.68871	5.01147	**4.73161**
F3	Xargs.l	74	**4.92382**	5.39863	5.33996	5.52851	**5.44342**	5.07334	5.16087	5.34651	**5.08761**
F4	grammar.lsp	76	**4.66434**	5.04757	5.01774	5.24805	**5.13455**	4.85461	4.78581	5.04473	**4.76816**
F5	Plabn12.txt	81	**4.57534**	4.94473	4.89527	4.98592	**4.98433**	4.80659	4.64910	4.83863	**4.69002**
F6	lcet10.txt	84	**4.69712**	5.12232	5.01682	5.22562	**5.10563**	4.87868	4.74177	4.95115	**4.81642**
F7	cp.html	86	**5.26716**	5.85839	5.81173	6.09414	**5.98710**	5.37113	5.77080	5.42775	**5.28917**
F8	Fields.c	90	**5.04090**	5.47596	5.46332	5.69381	**5.22125**	5.26987	5.20278	5.39520	**5.17480**
F9	Ptt5	159	**1.66091**	1.77735	1.75992	1.80165	**1.79499**	1.71814	1.70401	1.79911	**1.67945**
F10	Sum	255	**5.36504**	6.10683	6.03917	6.32367	**6.27025**	5.49767	6.01870	6.30885	**5.49070**
F11	kennedy.xls	256	**3.59337**	4.25681	4.27209	4.37120	**4.32269**	3.89401	3.85384	3.96247	**3.82626**

(: applying the level-skipping adaptation)

Table II. Coding performances of robust symmetrical RVLCs for the English alphabet

Occurrence Probability		Huffman : S1		Tsai's algorithm : S2		Tseng's algorithm : S3		Laković's algorithm : S4 ($d_{free} \geq 2$)		Proposed algorithm : S5 ($d_{free} \geq 2$)	
		L	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**
E	0.14878570	3	001	3	010	3	000	3	010	2	00 ($\mathbf{Z}_2$)
T	0.09354149	3	110	3	101	3	111	3	101	3	010
A	0.08833733	4	0000	4	0110	3	010	4	0110	3	101
O	0.07245796	4	0100	4	1001	3	101	4	1001	4	1111
R	0.06872164	4	0101	4	0000	4	0110	4	0000	4	0110
N	0.06498532	4	0110	4	1111	4	1001	4	1111	4	1001
H	0.05831331	4	1000	5	01110	5	00100	5	01110	5	11011
I	0.05644515	4	1001	5	10001	5	11011	5	10001	5	01110
S	0.05537763	4	1010	5	00100	5	01110	5	00100	5	10001
D	0.04376834	5	00010	5	11011	5	10001	5	11011	6	110011
L	0.04123298	5	00011	6	011110	6	001100	6	011110	6	011110
U	0.02762209	5	10110	6	100001	6	110011	6	100001	6	100001
P	0.02575393	5	10111	6	001100	6	011110	6	001100	7	1110111
F	0.02455297	5	11100	6	110011	6	100001	6	110011	7	1100011
M	0.02361889	5	11110	7	0111110	7	0010100	7	0111110	7	0111110
C	0.02081665	5	11111	7	1000001	7	1101011	7	1000001	7	1000001
W	0.01868161	6	011100	7	0010100	7	0011100	7	0011100	8	11100111
G	0.01521216	6	011101	7	1101011	7	1100011	7	1100011	8	11000011
Y	0.01521216	6	011110	7	0011100	7	0111110	7	0001000	8	01111110
B	0.01267680	6	011111	7	1100011	7	1000001	7	1110111	8	10000001
V	0.01160928	6	111011	7	0001000	8	00111100	8	01111110	9	111000111
K	0.00867360	7	1110100	7	1110111	8	11000011	8	10000001	9	110000011
X	0.00146784	8	11101011	8	01111110	8	01111110	8	00111100	9	110101011
J	0.00080064	9	111010101	9	011111110	8	10000001	9	011111110	9	011111110
Q	0.00080064	10	1110101000	10	0111111110	9	011111110	10	0111111110	9	100000001
Z	0.00053376	10	1110101001	10	1000000001	9	100000001	10	1000000001	10	1110000111
Average length		4.15572392		4.69655649		4.60728507		4.46463681		**4.567250**	

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

Occurrence Probability		Huffman : A1		Tsai's algorithm : A2		Lin's algorithm : A3		Laković's algorithm : A4 ($d_{free} \geq 2$)		Proposed algorithm : A5 ($d_{free} \geq 2$)	
		L	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**	**L**	**codeword**
E	0.14878570	3	001	3	000	3	000	3	000	3	000 ($\mathbf{Z}_3$)
T	0.09354149	3	110	3	111	3	100	3	111	3	011
A	0.08833733	4	0000	4	0101	3	101	4	0101	3	101
O	0.07245796	4	0100	4	1010	4	0010	4	1010	3	110
R	0.06872164	4	0101	4	0010	4	0011	4	0110	4	0010
N	0.06498532	4	0110	4	1101	4	0110	4	1001	4	0100
H	0.05831331	4	1000	4	0100	4	0111	4	0011	4	1001
I	0.05644515	4	1001	4	1011	4	1110	4	1100	4	1111
S	0.05537763	4	1010	4	0110	4	1111	5	00100	5	00111
D	0.04376834	5	00010	5	11001	5	01001	5	11011	5	01010
L	0.04123298	5	00011	5	10011	5	01010	5	01110	5	10001
U	0.02762209	5	10110	5	01110	5	01011	5	10001	5	11100
P	0.02575393	5	10111	5	10001	5	11001	6	010010	6	001100
F	0.02455297	5	11100	6	001100	5	11010	6	101101	6	010111
M	0.02361889	5	11110	6	011110	5	11011	6	011110	6	100001
C	0.02081665	5	11111	6	100001	6	010001	6	100001	6	111010
W	0.01868161	6	011100	7	1001001	6	110001	6	001011	7	0011010
G	0.01521216	6	011101	7	0011100	7	0100001	6	110100	7	0101100
Y	0.01521216	6	011110	7	1100011	7	1100001	7	0100010	7	1000001
B	0.01267680	6	011111	7	0111110	8	01000001	7	1011101	7	1110111
V	0.01160928	6	111011	7	1000001	8	11000001	7	0010100	8	00110111
K	0.00867360	7	1110100	8	00111100	9	010000001	7	1101011	8	01011010
X	0.00146784	8	11101011	8	11000011	9	110000001	8	10111101	8	10000001
J	0.00080064	9	111010101	9	100101001	10	0100000001	9	010000010	8	11101100
Q	0.00080064	10	1110101000	10	0011101001	10	1100000001	10	0100000010	9	001101100
Z	0.00053376	10	1110101001	10	1001011100	11	01000000001	10	1011111101	9	010110111
Average length		4.15572392		4.30677804		4.18734808		4.18734808		**4.236589**	