METHODS FOR OBTAINING REVERSIBLE VARIABLE LENGTH CODES FROM HUFFMAN CODES

Aamir Saeed Malik and Yo-Sung Ho
Kawangju Institute of Science and Technology
1 Oryong-dong, Puk-gu, Kwangju, 500-712, Korea

ABSTRACT

Reversible Variable Length Coding (RVLC) can be regarded as an extension of Variable Length Coding (VLC) with the unique suffix property as well as the unique prefix property of VLC. This paper proposes two methods for RVLC which are based on the VLC. Huffman coding is modified to obtain the RVLC.

1. INTRODUCTION

Recent advances in technology have resulted in a rapid growth in mobile communications. So there is a need for reliable transmission of multimedia information, i.e., audio, video, text, graphics and speech data. Typically, the bandwidth requirements for the raw data are very high. Therefore, video compression techniques are used to reduce the bandwidth requirements and enable the transmission of video information over band-limited wireless channels.

Wireless channels are generally noisy and suffer from a number of channel degradations, such as bit errors and burst errors due to fading and multipath reflections. When compressed video data is sent over these channels, it is subject to these degradations. Effect of channel errors on compressed video can be very severe. Variable length coding schemes also render the compressed bitstream very brittle to channel errors.

As a result, the video decoder that is decoding the corrupted video bitstream loses synchronization with the encoder. Predictive coding techniques, such as motion compensation, applied in the current video compression standards, make matters worse by quickly propagating the effects of channel errors across the video sequence and rapidly degrading video quality. Unless the video encoder and the decoder take proper remedial steps, the video communication system totally breaks down.

The ISO MPEG [1] standards and other image coding standards (like JPEG, H.26x family) utilize coding in the form of Variable Length Codes (VLCs) [2]. Variable length codes are well known to give efficient compression, but are very vulnerable in the noisy environments because of synchronization losses that can accompany bit errors [3]. In most applications, these standards have utilized reliable media, rendering vulnerability of the compressed bitstream to noise less important.

As stated earlier, there is a growing level of interest in wireless exchange of compressed image and video signals. For applications where delay is less critical, the wireless channel can be made extremely reliable by use of suitable retransmission protocols. However, retransmission becomes problematic in delay constrained real-time applications, and the issue of how to design and use VLCs in error prone environments takes on high importance.

These considerations have led to a growing level of interest in Reversible Variable Length Codes (RVLCs).

2. REVERSIBLE VARIABLE LENGTH CODES

Reversible Variable Length Codes (RVLCs) are designed such that they can be instantaneously decoded both in the forward and reverse directions. A part of the bitstream that cannot be decoded in the forward direction due to the presence of errors, can often be decoded in the backward direction. Therefore, the number of discarded bits can be significnatly reduced.

Due to the extensive use of VLC tables, even after error detection, typically the decoder can only isolate the error to be somewhere between the resynchronization points, but it cannot pinpoint its exact location. Hence, all data that corresponds to the macroblocks between these two resynchronization points needs to be discarded, as shown in Fig. 1. The effects of displaying an image reconstructed from erroneous data can cause highly annoying visual artifacts.

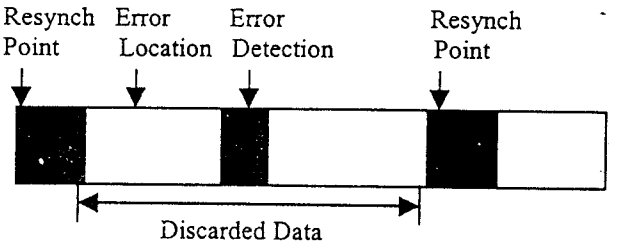

Figure 1: Decoding with VLC Table

RVLCs can provide a solution to some extent. They enable the decoder to salvage some data between two resynchronization points. By comparing the forward and the reverse decoded data, we can localize the exact location of the error in the bitstream more precisely and we can salvage some data between the two resynchronization points. This operation is illustrated in Fig. 2. The use of RVLCs is a part of the MPEG-4 standard [1].

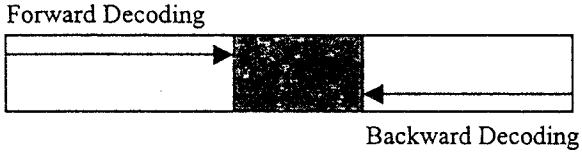

Figure 2: Decoding with RVLC table

RVLCs are special VLCs that have the prefix property when decoding them in both the forward and the reverse directions. Hence, they can be uniquely decoded in both directions. The advantage of RVLCs is that when a decoder detects an error while decoding the bitstream in the forward direction, it jumps to the next resynchronization marker and decodes the bitstream in the backward direction until it encounters an error.

Therefore, RVLCs can provide error resilient transmission with little overhead. Another potential use of RVLC is in the random access or in the searching of an entropy encoded stream of data. The ability to search in both directions should halve the amount of indexing overhead or the number of index points necessary to achieve the same average search time, as compared to a one directional VLC.

RVLC must satisfy the suffix condition for instantaneous backward decoding as well as the prefix condition for instantaneous forward decoding. The suffix condition is sufficient for instantaneous decoding in the backward direction just as the prefix condition is in the forward direction. It is worth noting that a VLC already satisfies the prefix condition.

3. PREVIOUS WORKS

Takishima, Wada and Murakami [4] were one of the first who studied RVLC and proposed algorithms for their construction. They defined two main classes of RVLC algorithms, namely symmetrical and asymmetrical RVLCs. They reported that the symmetrical and asymmetrical RVLCs are 13% and 4.9% longer than the corresponding VLC code, respectively.

Wen and Villasenor [5] presented the ideas behind two general classes of RVLCs and discussed the results by applying the codes in the framework of the H.263+ and MPEG-4 video coding standards. They described the reversible codes that have the same length distribution as the Golomb-Rice (GR) codes and exp-Golomb (EG) codes. They also presented a more general class of RVLC [6]. In summer of 1997, they presented a RVLC proposal in collaboration with Ericsson to the ITU-T low-bit-rate coding group, charged with developing H.263+ [7]. The ITU document proposed the use of RVLCs based on the construction method in [6], and was accepted by the ITU and included as a part of Annex D of H.263+.

Chujoh and Watanabe [8] described the RVLCs and their application to error tolerant video coding. They introduced four methods for RVLCs. However, they did not implement all the methods.

4. TWO METHODS FOR RVLCs

4.1 METHOD 1

In the first method, we start with a Huffman code. From the Huffman table, all the codewords with non-unique suffixes are identified and marked. Then the shortest bit length of the non-unique suffix codewords is denoted as 'm'.

The first stage of forming the RVLC table is to define a combination of bits, denoted as 'S'. The length of 'S' is m-1 bits and it is formed according to the procedure described below. Once we obtain 'S', we append 'S' to all the non-unique suffix codewords of m-bit length, hence making them unique. This process is repeated until all the codewords are dealt with.

STEP 1: Let 'n' be the shortest bit length of the non-unique suffix codewords in the Huffman table. We select a combination of 'n' bits, 'Z', which is not used in the Huffman table. Let

$$S = Z \tag{1}$$

STEP 2: For the n-bit length codeword 'S', we check each of the bit locations. Bit locations are identified in each n-bit length codeword where the values are the same to the corresponding bit locations in all the n-bit length codewords. Then, the corresponding locations in 'S' are replaced with the opposite values.

STEP 3: Check if

$$n = m - 1 \tag{2}$$

If Eq. (2) is satisfied, the construction of 'S' is completed; otherwise, we affix one more bit to the MSB location of 'S', and we repeat STEP 2 after increasing 'n' by 1.

However, this scheme has few problems. If no bit location of codewords has the same bit in STEP 2, the solution is to take a codeword that is not used in the Huffman table. Another problem occurs if more than one

bit location has the same value in all n-bit length codewords. The solution to this problem is to keep track of all the bits changed for the first time. If the same bit has to be changed again, it is replaced while the other changed bits are made constant in the worst case.

This method has low computational complexity compared to Method 2, but it has more overhead than Method 2.

4.2 METHOD 2

In the second method, we also start with a Huffman code. All the codewords with non-unique suffixes are identified. The shortest bit length codewords are found which are not used in the Huffman table. Let 'A' be a set containing all the unused codewords:

$$A = \{l_1, l_2, \ldots, l_i, \ldots, l_n\} \quad (3)$$

where l_i is a codeword, not used in the Huffman table.

Now, the smallest codeword in 'A' is considered, i.e., l_1. x-LSB's (x is initialized to 1) of l_1 are taken and appended to the non-unique suffix codewords, identified earlier. If it results in a unique suffix codeword, the codeword is removed from the list of the non-unique suffix codewords.

The same process is repeated for all non-unique suffix codewords by incrementing 'x' until 'x' becomes equal to the total bit length of the codeword, as defined in Eq. (3). If there are still some codewords with non-unique suffixes, the whole process is repeated with the next codeword in the set 'A'.

This method has high computational complexity compared to Method 1, but results in a low overhead.

5. EXPERIMENTAL RESULTS

Two methods have been tested and compared with the Huffman code. Method 1 results in a RVLC code, which is 12.8% longer than the corresponding Huffman code. Method 2 produces better results. The RVLC code, generated by Method 2, is approximately 5% longer then the corresponding Huffman code.

As we can notice in Table 1, the entropy of the source remained the same, since it only depends on the probability of occurrences. Table 1 shows that the average length of codewords of Method 2 is very close to that of the Huffman code, as compared with Method 1. Therefore, Method 2 outperforms Method 1 in terms of code efficiency. In addition, code efficiencies of Method 1 and Method 2 are very close to symmetrical and asymmetrical RVLCs [4], respectively.

An example for obtaining RVLCs from a Huffman code by Method 1 and Method 2, is given in Appendix.

Table 1: Comparison of Coding Efficiency

	Method 1	Method 2	Huffman
Entropy	4.192	4.192	4.192
Average Length	4.87	4.53	4.3175
Code Efficiency	86%	92%	97%
Bit Increase over Huffman	12.8%	4.9%	

6. CONCLUSIONS

The proposed methods are heuristic approaches based on the simple notion that VLCs can be modified to obtain RVLCs. Method 2 is better than Method 1 in terms of coding efficiency. Method 2 has about 5% overhead with respect to the corresponding Huffman codes.

ACKNOWLEDGEMENTS

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultra-Fast Fiber-Optic Networks (UFON) Research Center at Kwangju Institute of Science and Technology (K-JIST).

REFERENCES

[1] R. Talluri, "Error-Resilient Video Coding in the ISO MPEG-4 Standard", IEEE Communications Magazine, pp. 112-119, June 1998.

[2] Y. Takishima, M. Wada and H. Murakami, "Variable length code with High Resynchronization Capability", Journal of the Institute of Television Engineers of Japan, vol. 43, No. 8, pp. 844-846, 1989.

[3] J. Maxted and J. Robinson, "Error Recovery for Variable Length Codes", IEEE Trans. IT-31, 6, pp.794-801, Nov, 1985.

[4] Y. Takishima, M. Wada and H. Murakami, "Reversible Variable Length Codes", IEEE Trans. Comm., vol. 43, No. 2/3/4, pp. 158-162, 1995.

[5] J. Wen and J. Villasenor, "Reversible Variable Length Codes for Efficient and Robust Image and Video Coding", IEEE Data Compression Conference, pp. 470-480, March, 1998.

[6] J. Wen and J. Villasenor, "A class of Reversible Variable Length Codes for robust image and video coding", Proc. 1997 IEEE Int. Conf. Image Proc., vol. 2, pp. 65-68, Oct 1997.

[7] T. Einarsson, J. Wen and J. Villasenor, "Proposal for longer motion vectors in H.263+", Document Q15-B-21, Sunriver, OR, Sep 8-11, 1997.

[8] T. Chujoh and T. Watanabe, "Reversible Code and Its Application to Error Tolerant Video Coding", Electronics Information and Communication Institute, vol. J-80A, No.3, pp. 532-541, March 1997.

APPENDIX: Example of Obtaining RVLC

(1) Identify the codewords with non-unique suffixes (the same for Method 1 and Method 2)

Alphabet	Probability	Huffman	Method 1/2
E	0.13	000	000
T	0.09	0010	0010
A	0.08	0011	0011
O	0.08	0100	0100
R	0.07	0101	0101
N	0.065	0110	0110
H	0.065	0111	0111
I	0.06	10000	xx
S	0.06	10001	10001
D	0.04	10010	xx
L	0.035	10011	xx
U	0.03	10100	xx
P	0.03	10101	xx
F	0.03	10110	xx
M	0.02	10111	xx
C	0.02	11000	xx
W	0.02	11001	11001
G	0.015	11010	11010
Y	0.015	11011	11011
B	0.015	11100	11100
V	0.01	11101	11101
K	0.005	111100	xx
X	0.005	111101	xx
J	0.005	111110	111110
Q	0.0025	1111110	xx
Z	0.0025	11111111	11111111

xx: Codewords with non-unique suffix

(2) Modify the non-unique suffix codewords to unique suffix codewords

Alphabet	Prob.	Huffman	Method1	Method2
E	0.13	000	000	000
T	0.09	0010	0010	0010
A	0.08	0011	0011	0011
O	0.08	0100	0100	0100
R	0.07	0101	0101	0101
N	0.065	0110	0110	0110
H	0.065	0111	0111	0111
I	0.06	10000	100001100	100001
S	0.06	10001	10001	10001
D	0.04	10010	100101100	xx
L	0.035	10011	xx	xx
U	0.03	10100	101001100	101001
P	0.03	10101	xx	101011
F	0.03	10110	101101100	101101

xx: Codewords with non-unique suffix

(2) (Continued): Modify the non-unique suffix codewords to unique suffix codewords

Alphabet	Prob.	Huffman	Method1	Method2
M	0.02	10111	xx	101111
C	0.02	11000	110001100	xx
W	0.02	11001	11001	11001
G	0.015	11010	11010	11010
Y	0.015	11011	11011	11011
B	0.015	11100	11100	11100
V	0.01	11101	11101	11101
K	0.005	111100	xx	xx
X	0.005	111101	xx	xx
J	0.005	111110	111110	111110
Q	0.0025	1111110	xx	xx
Z	0.0025	11111111	11111111	11111111

xx: Codewords with non-unique suffix

(3) Get the final RVLC table

alphabet	Prob.	huffman	Method1	Method2
E	0.13	000	000	000
T	0.09	0010	0010	0010
A	0.08	0011	0011	0011
O	0.08	0100	0100	0100
R	0.07	0101	0101	0101
N	0.065	0110	0110	0110
H	0.065	0111	0111	0111
I	0.06	10000	100001100	100001
S	0.06	10001	10001	10001
D	0.04	10010	100101100	1001001
L	0.035	10011	1001101100	1001101
U	0.03	10100	101001100	101001
P	0.03	10101	1010101100	101011
F	0.03	10110	101101100	101101
M	0.02	10111	1011101100	101111
C	0.02	11000	110001100	1100001
W	0.02	11001	11001	11001
G	0.015	11010	11010	11010
Y	0.015	11011	11011	11011
B	0.015	11100	11100	11100
V	0.01	11101	11101	11101
K	0.005	111100	111100001100	11100010
X	0.005	111101	111101011100	111101010
J	0.005	111110	111110	111110
Q	0.0025	1111110	1111110001100	11111100011
Z	0.0025	11111111	11111111	11111111
Av.code Length		4.3175	4.87	4.53