ALTERNATING CODING FOR UNIVERSAL VARIABLE LENGTH CODE

Shang Xue and Bengt Oelmann

Department of Information Technology and Media, Mid Sweden University
SE-851 70 Sundsvall, Sweden
xue.shang@mh.se

ABSTRACT

Variable length code (VLC) is used in a large variety of lossless compression applications. A specially designed VLC, called "Universal Variable Length Code" (UVLC), is utilized in the latest video coding standard H.26L under development. In this work we propose a new coding method under the name "Alternating Coding" (ALT coding) and apply it to the UVLC. The ALT coded UVLC yields the same high coding efficiency as the original UVLC, but ALT coding also enables immediate decoding, immediate error detection and location in a UVLC packet. It improves the error resiliency and error robustness of the UVLC packet by applying an "error speculation". Results show that the ALT coded UVLC packet always outperforms the original UVLC packet in terms of error resiliency and robustness. Moreover, the simple pattern of ALT coded UVLC can greatly simplify the UVLC decoder architecture and therefore enables small, fast, and low-power decoder design.

1. INTRODUCTION

The video coding standard H.26L uses a unique variable length code pattern (VLC) which is called Universal Variable Length Code (UVLC) to perform entropy coding. It was first proposed in [2]. In [2,3,4] it is suggested that it be used in the coding of motion vectors as well as DCT coefficients for H.26L. In [6] a similar code is also suggested for use in MPEG 4. UVLC is claimed to be able to provide good performances in terms of coding efficiency, configurability to various applications, and error resiliency. The error resiliency of UVLC is achieved by extending the UVLC to a bi-directionally decodable mode. However, UVLC still belongs to the VLC family and the variable code lengths of UVLC limit the decoding throughput as the decoding of a codeword depends recursively on those previously decoded. The bi-directional decodability provides UVLC with better error resiliency whereas demands decoding to be performed in both directions. This requires dual decoder structures. Moreover, errors in a bi-directionally decodable UVLC packet cannot be detected or located unless decoding is performed till almost the end of the packet. This, again, limits the decoding throughput to a large extent.

In this paper, we propose a new coding method called "Alternating Coding" method (ALT coding) to the UVLC. It enables the extraction of the code length information of UVLC, which facilitates the code packet with immediate decoding ability as well as immediate error detection and location ability without performing decoding bi-directionally. This improves the decoding throughput without the expense of dual decoder structures. The ALT coded UVLC has a very simple code pattern. This enables an "error speculation" to be applied to the ALT coded UVLC packet which improves its error resiliency and error robustness. Simulation results show that the ALT coded UVLC almost always yields a data loss of less than 20% when the sequence is subjected to errors, whereas the data loss of the original UVLC can exceed 60%. Moreover, the simple code pattern also enables the simplification of the decoder structure and the design of smaller, faster and low-power UVLC decoder [1].

2. UVLC AND ITS ERROR RESILIENCY

The bi-directionally decodable UVLC, which from now on will be referred to as UVLC, is constructed by interleaving symmetric VLC code with fixed length code (FLC) whose length is determined with respect to the length of the symmetric code. The coding efficiency of UVLC has been proven to be nearly optimal in the coding of many video signals due to their statistical properties [6]. The symmetric VLC is denoted coarse code, while the FLC is named additional code. After interleaving, the bits in the coarse code become odd-indexed bits (OIBs) in the UVLC, whereas the bits in additional code become even-indexed bits (EIBs). Table 1 gives an example of UVLC [2].

Table 1. An example of UVLC

Class k	Coarse code	Additional code	UVLC Codeword	UVLC Length	Value to be expressed
1	1	None	1	1	1
2	00	x_0	$0x_00$	3	'x_0'+ 2[2:3]
3	010	x_1x_0	$0x_1 1x_0 0$	5	'x_1x_0'+ 4[4:7]
4	0110	$x_2x_1x_0$	$0x_2 1x_1 1x_0 0$	7	'$x_2x_1x_0$'+ 8[8:15]
5	01110	$x_3x_2x_1x_0$	$0x_3 1x_2 1x_1 1x_0 0$	9	'$x_3x_2x_1x_0$'+ 16[16:31]
...	... ...	... ...	... ...	...	... ...

Transmission errors in a packet containing codewords of a VLC can be classified into propagating errors and non-propagating errors. As the EIB of UVLC can be any binary combination, an error occurring in the EIB will not propagate and decoding performs continuously without

0-7803-7750-8/03/$17.00 ©2003 IEEE

noticing the error. Therefore, errors in the EIB can never be detected or located. The OIB of a UVLC is of a fixed pattern, and an error which occurs in the OIB will cause the error to propagate and therefore cause the loss of synchronization.

When the number of source symbols contained in a packet is known to be N (such as for motion vectors and DCT coefficients in H.263), the error resiliency of UVLC is gained through decoding the packet from both directions. Errors can be detected when one of the following cases occurs:

1. Upon decoding the last bit of the packet does not coincide with the end of a codeword.
2. When the number of decoded symbols accumulates to a larger total than N.
3. When an illegal codeword is reached in the decoding procedure.

When decoding is interrupted by one of the above, errors must have occurred in one of all the previously decoded codewords. The location of the error cannot be made more precisely until the backward decoding is completed. This slows down the decoding and demands extra decoder structures. Moreover, even backward decoding the packet guarantees neither a more precise error location nor the resynchronization and it is still highly probable that the packet of codes is wasted.

3. ALT CODING FOR UVLC

ALT coding for UVLC involves coding the OIBs and the EIBs separately. The OIBs can actually be thought of as a set of unary expression. Two codeword tables are applied to the OIBs, one is {1, 11, 111, 1111, ...}, the other is {0, 00, 000, 0000, ...}. They are alternated in the coding procedure so that the codeword boundaries and codeword lengths can be easily determined by detecting the value changes in the OIB sequence. EIBs can be any binary combination but their code lengths can be determined by decoding the corresponding OIB. So the existing EIBs are maintained. Then the OIBs and EIBs of the ALT coded UVLC packet are transmitted separately as Figure 1 shows.

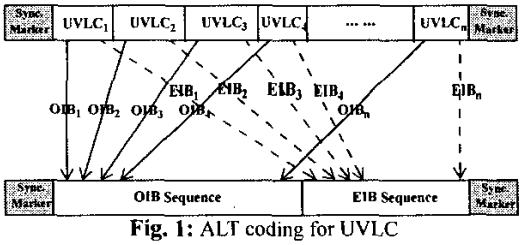

Fig. 1: ALT coding for UVLC

For example, the UVLC packet 000 00110 010 01110 after ALT coding becomes an OIB sequence 11 000 11 000 and an EIB sequence 0 01 1 11. The ALT coded packet is transmitted as:

11 000 11 000 0 01 1 11.

ALT coding does not change the code length of each UVLC, so it is able to achieve exactly the same coding efficiency as the original UVLC.

To perform decoding, the ALT packet needs to be firstly partitioned into an OIB sequence and an EIB sequence. Let N be the number of codewords in the packet, L be the length of the packet, l_{OIB} be the length of the OIB sequence and l_{EIB} be the length of the EIB sequence. Then $l_{EIB} = l_{OIB} - N$, $L = l_{EIB} + l_{OIB} = 2l_{EIB} - N$. With the ready information of L and N, the packet can be easily partitioned. The decoding procedure is described in Figure 2.

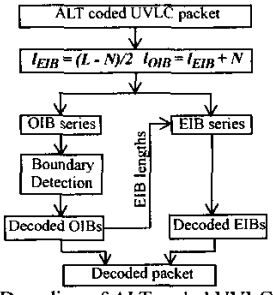

Fig. 2: Decoding of ALT coded UVLC packet

The OIB sequence is just a combination of alternating all-one codes and all-zero codes, therefore, decoding of the OIB can be done immediately and in parallel by checking the value changes in the OIB sequence (this could be done by performing a row of XOR operations to the OIB sequence). Once the OIB sequence is decoded, the lengths of EIBs can be calculated so that the complete codewords can be decoded.

As errors occur in the EIB does not propagate and will therefore be impossible to be detected. It is useless to consider the EIB sequence in terms of error detection. Whereas errors which occur in the OIB will propagate and thus only OIB sequence will be considered.

Suppose an ALT coded UVLC packet consists of N codewords of L bits long. The decoded OIBs are denoted as $l_1, l_2, ..., l_M$. The lengths of the corresponding EIBs are then l_1-1, l_2-1, ..., l_M-1. M is the number of codewords detected. Errors will be detected when one or more of the following cases are encountered:

1. $M<N$.
   Let f and b satisfy:
   $l_1 - 1 + l_2 - 1 + ... + l_f - 1 \geq \frac{L-N}{2}$, and
   $l_N - 1 + l_{N-1} - 1 + l_{N-2} - 1 + ... + l_b - 1 \geq \frac{L-N}{2}$.
   Then the probable correctly decoded OIB set A is:
   $A = \{x | x \in (l_1, ..., l_{b-1}) \cup (l_{f+1}, ..., l_N)\}$

2. $M>N$.
   Then the probable correctly decoded OIB set B is:
   $B = \{x | x \in (l_1, ..., l_{M-N-1}) \cup (l_{N+2}, ..., l_M)\}$

3. OIBs longer than the longest possible OIB are detected.

Assume that the OIBs which exceeds the longest OIB are OIBs number $x_1, x_2, ..., x_k$.

Then the probable correctly decoded OIB set C is:

$$C = \left\{ x | x \in (l_1, ..., l_{x_1 - 1}) \cup (l_{x_k + 1}, ..., l_N) \right\}$$

The decoded OIB set will then be:

$$\text{OIB set} = \begin{cases} A, & \text{case 1,} \\ A \cap C, & \text{case 1 and case 3,} \\ B, & \text{case 2,} \\ B \cap C, & \text{case 2 and case 3,} \\ C, & \text{case 3.} \end{cases}$$

It can be seen that the simple pattern of the OIB greatly simplifies the bi-directional decoding of the ALT coded UVLC and no dual decoder structures are needed.

Error resiliency of ALT coded UVLC can be improved by applying a simple "error speculation" to the error-infected packet. To simplify the analysis, we assume that only one bit error occurs in an OIB sequence. The bit error will have four types of influences on the OIB sequence.

1. An error occurring on the boundary of the OIB sequence causes an insertion or a deletion of one codeword. For example, the first codeword 1111 becomes 0111 or the first two codewords 0111 become 1111.
2. An error infects the shortest OIB (i.e. one-bit OIB) which sits in between two codewords. This results in a deletion of two codewords. For example, 1110111 becomes 1111111. Then three OIBs become only one.
3. An error occurs in the middle of an OIB whose length is greater than two bits. This results in the insertion of two codewords. For example., 1111111 becomes 1110111. Then one OIB becomes three.
4. An error occurs on the boundary of two OIBs. This is a non-propagating error. For example, 1110000 becomes 1111000. This will not influence synchronization.

When case 1, 2 or 3 occurs, the number of OIBs detected will not be equal to N. When one of these cases is detected, we speculate where the error bit occurs by the "error speculation".

If the number of OIBs is $N-1$ or $N+1$, then case 1 has occurred. The error is then speculated to have occurred on the first or the last bit.

If the number of OIBs is $N-2$, then case 2 has occurred. If there exists an OIB that has a length longer than the longest possible OIB length, this OIB must have been infected by a bit error. In this case, the error can be located precisely. Otherwise, it is speculated that the location of the error is within the longest OIB (As longer OIBs have less probability of occurrence.) in the OIB sequence, and randomly change the value of one bit in this OIB. By doing so, resynchronization is achieved and many correct codewords can be resumed.

If the number of OIBs is $N+2$, then case 3 has occurred. It is assumed that a one-bit OIB in between the two shortest OIBs is the error bit (this is reasonable as the shorter the codeword is, the more probable it occurs in a sequence and hence more probable to be infected by an error). Again, resynchronization and error recovery can both be achieved.

Actually, no more than 4 codewords will be ruined even in the worst speculation when only one bit error exists in the sequence. This makes the ALT coded packet perform robustly when subject to a bit error.

Table 2 shows an example of how the error speculation works. Only the OIB sequence is considered and the maximum OIB length is 5 bits. The speculated bit is underlined.

Table 2. Examples the decoding of error infected ALT coded UVLC sequence

Correct sequence	**00, 1, 00, 1, 0, 111, 000, 1, 00**		
Case 1	error seq	00, 1, 00, 1, 1, 111, 000, 1, 00	6 codes resumed
	9-2 codes detected	00, 1, 00, 11111, 000, 1, 00	
	guess on long code	*00, 1, 00,* 11, 0, 11, *000, 1, 00*	
Case 2	error seq	00, 1, 00, 1, 0, 101, 000, 1, 00	6 codes resumed
	9+2 codes detected	00, 1, 00, 1, 0, **1, 0,** 1, 000, 1, 00	
	guess on short code	*00, 1, 00,* 1**1**, **0**, 1, *000, 1, 00*	
Case 3	error seq	00, 1, 00, 1, 0, 110, 000, 1, 00	7 codes resumed
	9 codes	00, 1, 00, 1, 0, 11, 0000, 1, 00	
	sync not influenced	*00, 1, 00, 1, 0,* 11, 0000, *1, 00*	
Case 4	error seq	00, 1, 00, 1, 0, 111, 000, 1, 01	all codes resumed
	10 codes	00, 1, 00, 1, 0, 11, 1000, 1, 0,1	
	guess on last bit	*00, 1, 00, 1, 0,* 11, 1000, *1, 00*	

In [7], the authors show that for a Binary Symmetric Channel (BSC), the crossover probability is below 10^{-3}, the possibility of having more than one bit errors in a packet is thus very small. Therefore, the error speculation is able to efficiently improve the error resiliency and robustness of a UVLC packet.

To complete the error resilient decoding of an ALT packet, two steps are involved. The first is the error speculation, and if this fails, normal ALT decoding is then performed.

4. SIMULATION RESULTS

The UVLC used in the simulations has maximum codeword length of 13 bits, which is adequate for most image/video applications [2,3]. The probability distribution of the symbols to be coded is optimized for the UVLC used in the simulation. 100 runs of simulations are performed for each result obtained.

The UVLC packet and the ALT coded UVLC packet are both subjected to corruption using a BSC with bit error rate (BER) of 10^{-4} and 10^{-3}. Figure 3 shows a comparison of the Correct Rate (CR) of ALT coded UVLC and UVLC. The number of codewords in a packet varies from 8 to 1024.

$$CR = \frac{\text{No. of correctly decoded codes}}{\text{Total No. of codes in the packet}}$$

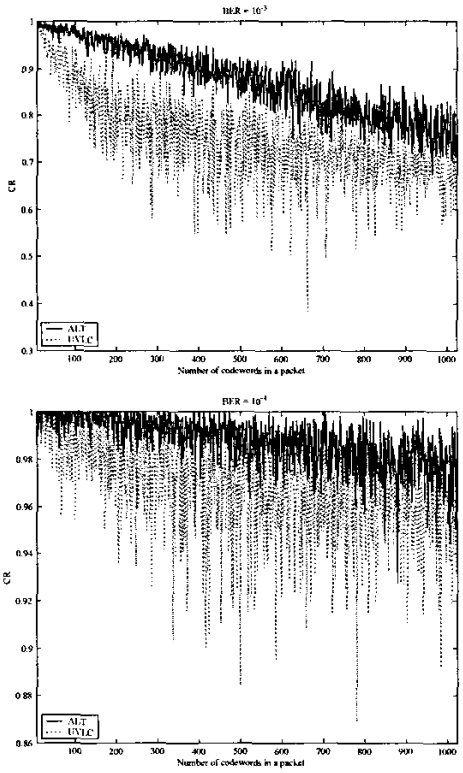

Fig. 3: Comparison of CR of ALT and UVLC under different BERs

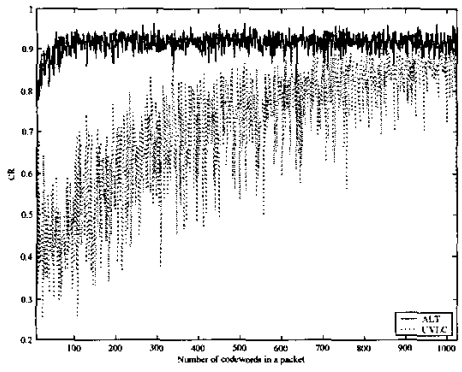

Fig. 4: CR of ALT and UVLC when packet is infected by single bit error

From Figure 3 it can be seen that the ALT packet always outperforms the UVLC packet. The CR of the ALT packet is almost always over 80% and has a much smaller variance. However the CR of the UVLC packet goes below 60% for large packet sizes and bigger BER and its variance is much bigger. In addition, the advantage of the ALT packet is more evident when the packet is subjected to higher BER. It can be seen that the CR of ALT packet under 10^{-3} BER is comparable to that of the UVLC under 10^{-4} BER when the packet size is less than 500.

In order to show the effectiveness of the "error speculation", the CR of the ALT packet and the UVLC packet are also compared when only one bit error occurs in the packet. The number of codewords in the packet varies from 8 to 1024. The results are shown in Figure 4.

From the results it can be seen that the ALT coded UVLC always achieves a CR of nearly 90%, yet that of the UVLC packet can even go below 30%. Moreover, the CR of the ALT packet shows much smaller variance than that of the UVLC packet and is therefore much more robust.

5. CONCLUSIONS

In this paper we propose the ALT coding method and apply it to UVLC. ALT coding yields the same high coding efficiency as UVLC, but ALT coded UVLC packet is immediately bi-directionally decodable and immediately error detectable. By applying an "error speculation", resynchronization may be achieved, thus better error resiliency and error robustness are gained. In addition, the ALT packet has a simple pattern that enables the design of efficient decoder [1].

6. REFERENCES

[1] S. Xue and B. Oelmann, "A coding method for UVLC targeting efficient decoder architecture," to appear in the proceedings of the *3rd International Symposium on Image and Signal Processing and Analysis.*, Sep. 2003.

[2] Y. Itoh, "Bi-directional motion vector coding using universal VLC," *Signal Processing: Image Communication*, vol. 14, pp. 541-557, May 1999.

[3] Y. Itoh, Ngai-Man Cheung, "Universal variable length code for DCT coding," in *International Conference on Image Processing*, vol. 1, pp. 940-943, 2000.

[4] N.-M. Cheung, Y. Itoh, "Configurable variable length code for video coding," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1805-1808, 2001.

[5] ITU-T, *H.26L TML8 Document from http://standard.pictel.com*, Sep. 2001.

[6] J. Wen, J.D. Villasenor, "A class of reversible variable length codes for robust image and video coding," *International Conference on Image Processing*, vol. 3, Set-Volume 2, October 1997.

[7] M. Rahman and S. Misbahuddin, "Effects of a binary symmetric channel on the synchronization recovery of variable length code," Computer J., vol. 32, pp. 246-251, 1989.

[8] J. H. Jeon et al, "A fast variable-length decoder using plane separation," in *IEEE. Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 806-812, Aug. 2000.