ERROR RESILIENT CODING OF DCT COEFFICIENT USING ALTERNATING CODING OF UVLC

Shang Xue and Bengt Oelmann

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

ABSTRACT

Video coding standard H.26L uses a specially designed Variable Length Code (VLC), called Universal Variable Length Code (UVLC) to perform entropy coding for the DCT coefficients. In this work we apply a coding method under the name "Alternating Coding" (ALT coding) to the UVLC in the coding of DCT coefficients. The ALT coded UVLC yields same high coding efficiency as the original UVLC, whereas ALT coding also enables immediate decoding and error detection. It also improves the error robustness by applying an "error speculation" to the packet. For the UVLC coded RUN LEVEL pairs, we modify the ALT coding by further separating the sign bits of the "LEVELs" in each packet. Results show that the ALT coded UVLC always provides better PSNR as well as visual quality than UVLC when both are subjected to the same error environment.

1. INTRODUCTION

The Universal Variable Length Code (UVLC) is used in H.26L to perform entropy coding. In [3], UVLC is suggested to be used in the coding of DCT coefficients for H.26L. It 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, the variable code lengths of UVLC still limit the decoding throughput and extending the UVLC to bi-directional decodable codes demands dual decoder structures to perform decoding in both directions. Moreover, errors in a bi-directionally decodable UVLC packet cannot be detected or located immediately, this also limits the decoding throughput. In this paper, the coding method called "Alternating Coding" method (ALT coding) [1] is applied to the UVLC in the coding of DCT coefficients. ALT coding enables the extraction of the code length information of UVLC, which facilitates the code packet with immediate decoding as well as immediate error detection and location without performing decoding bi-directionally. Thus the dual decoder structures is no longer needed. ALT coding enables an "error speculation" to be applied which improves the error resiliency and error robustness. Moreover, the simple code pattern also simplifies the decoder structure and enables the design of smaller, faster and more power-saving UVLC decoder [2]. In H.26L, the UVLC uses one infinite-extent codeword set rather than designing a different code for each element of the H.26L syntax, only the mapping to the single UVLC code table is customized to the probabilistic behavior of the data. However, extra bits need to be added to indicate the signs of each LEVEL. To apply the ALT coding to DCT coefficients, we make further separation of the UVLC coded "LEVELs", which keeps the codeword of RUNs and LEVELs in accordance. This also helps to simplify the decoding scheme.

2.UVLC FOR DCT CODING

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 codes (FLC) whose length is determined with respect to the length of the symmetric code. 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. RUN UVLC

Coarse code	Additional code	UVLC		Value of RUN
		Codeword	Length	
1	None	1	1	0
00	x_0	$0x_00$	3	if x_0=0, EOB if x_0=1, RUN=1.
010	x_1x_0	$0x_11x_00$	5	'x_1x_0'+ 2[2:5]
... ...	... ...	... ...	... ...	... ...
0111110	$x_5x_4x_3x_2x_1x_0$	$0x_50x_40x_31x_21x_11x_00$	13	'$x_3x_2x_1x_0$'+ 62[62:125]

Table 2. LEVEL UVLC

Coarse code	Additional code	UVLC		Absolute value of LEVEL
		Codeword	Length	
1	None	1s	2	1
00	0	000	3	EOB
00	x_0	010s	4	2
010	x_1x_0	$0x_11x_00s$	6	'x_1x_0'+ 3[3:6]
... ...	... ...	... ...	... ...	... ...
0111110	$x_5x_4x_3x_2x_1x_0$	$0x_50x_40x_31x_21x_11x_00s$	14	'$x_5x_4x_3x_2x_1x_0$' + 63[63:126]

In Table 2, 's' is the sign bit.

In the coding of DCT coefficients, UVLC codes the RUNs and LEVELs separately. Table 1 and Table 2 give examples of RUN UVLC and LEVEL UVLC [3]. We see that the code tables of RUNs and LEVELs are actually identical except for the sign bits at the end of each codeword in the LEVEL table.

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 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 causes error propagation therefore causes the loss of synchronization.

3.ALT CODING FOR UVLC

ALT coding for UVLC involves coding the OIBs and the EIBs separately. We apply two codeword tables 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 we maintain the existing EIBs. Then the OIBs and EIBs of the ALT coded UVLC packet are transmitted separately as shown in Figure 1.

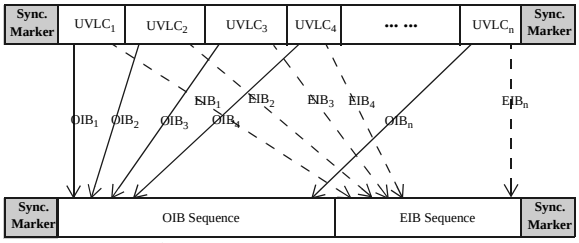


Fig. 1: ALT coding for UVLC

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. We have: $l_{EIB} = l_{OIB} - N$. and $L = l_{EIB} + l_{OIB} = 2l_{EIB} - N$. As long as L and N is known, the packet can be easily partitioned.

Errors occurring in the EIB do not propagate and will therefore be impossible to be detected, whereas errors which occur in the OIB will propagate, only OIB sequence is considered in detecting the errors.

Suppose we have an ALT coded UVLC packet that consists of N codewords of L bits length. 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:
   $l_1 - 1 + l_2 - 1 + \ldots + l_f - 1 \geq \frac{L-N}{2}$, and
   $l_N - 1 + l_{N-1} - 1 + l_{N-2} - 1 + \ldots + l_b - 1 \geq \frac{L-N}{2}$.
   Then the probable correctly decoded OIB set A is:
   $A = \{x | x \in (l_1, \ldots, l_{b-1}) \cup (l_{f+1}, \ldots, l_N)\}$

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

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:

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

The decoded OIB set will then be:

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

Here no dual decoder structures are needed. Once the OIBs are decoded, the EIBs can be calculated so that the complete UVLC is decoded.

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-propagation 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, we speculate 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. We assume 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.

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. APPLYING ALT CODING TO THE CODING OF DCT COEFFICIENTS

To apply ALT coding to DCT coefficients, we further separate each packet to a package of ALT coded UVLCs and a package of sign bits as the code tables of RUNs and LEVELs are identical except for the sign bits. Figure 2 shows the separation. By doing such a separation, the codewords in the "ALT coded UVLC packet" are then kept in accordance and therefore can be decoded as described in section 3.

After the ALT coded UVLC packet is decoded, the sign bits can then be imposed to the LEVELs as the positions of each LEVEL are then known.

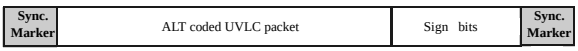


Fig. 2: Further separation of ALT packet in DCT coding

For the DCT coefficients, RUNs and LEVELs appear pairwisely, so the number of codewords between two EOBs must be even. However, the error speculation as well as ALT decoding itself, may result in an incorrect partition of the code packet and therefore the number of codewords between two EOBs may be odd. When the number of codewords between EOBs are detected to be odd, we always discard one codeword to make it even. This results in the absence of some high frequency components, which influence only the details of the block.

In DCT coding, the EOB plays a very important role as an error in the EOB results in an error propagation to the next block. The number of EOBs in the image is also a key factor in reconstructing the image.

Assume there are *X* EOBs in a packet, and *Y* EOBs are detected. We perform the following to guarantee the reconstruction of the image.

1. *X<Y.* Discard the extra ones at the end of the packet.
2. *X>Y.* Put zeros at the end of the packet to fill up the absent EOBs.

After the above are performed, the sign bits will then be matched to the decoded codewords. Due to the error speculation, we may have inserted or deleted some LEVELs in the packet, therefore, the sign bits may turn out to be too many or too few. For simplicity, if the sign bits are too many, we simply discard the extra bits; if the sign bits are too few, we deem the remaining LEVELs to be positive.

5. SIMULATION RESULTS

Several images are transformed using 8*8 DCT, zig-zag scanned and then run-length coded. The RUNs and LEVELs are then coded using UVLC and ALT.

These coded images are then subjected to a Binary Symmetric Channel (BSC) with a Bit Error Rate (BER) of 10^{-3}. The PSNR of the reconstructed images are then compared in Table 3.

Table 3. Comparison of PSNR

Image	PSNR of UVLC (dB)	PSNR of ALT (dB)
Lena	21.92	27.50
Cameraman	24.23	26.06
Monkey	17.81	22.38
House	27.67	30.07

From Table 3 we see that the ALT coded images are always better than the UVLC coded ones. The PSNR increases 2 ~ 5 dB approximately.

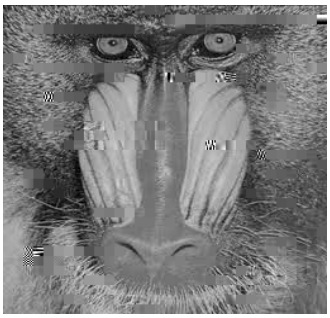
Figure 3 shows the comparison of the visual qualities of the images in Table 3. The qualities of the ALT coded images are evidently better.

(a) The reconstructed Lena using UVLC

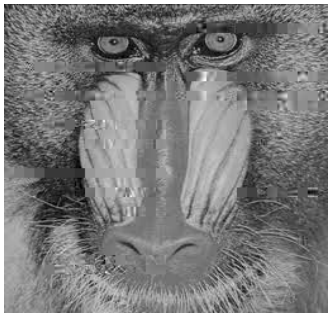
(b) The reconstructed Lena using ALT

(c) The reconstructed Cameraman using UVLC

(d) The reconstructed Cameraman using ALT



(e) The reconstructed Monkey using UVLC



(f) The reconstructed Monkey using ALT

(g) The reconstructed House using UVLC

(h) The reconstructed House using ALT

Fig. 3: Comparison of the reconstructed images

5.CONCLUSIONS

In this paper we apply the ALT coding to the coding of DCT coefficients using UVLC. ALT coded UVLC packet is immediately bi-directionally decodable and immediately error detectable. An “error speculation” helps to improve the error resiliency and error robustness. In addition, the ALT coded UVLC has a simple pattern that enables efficient decoder structure [2]. For DCT coefficients, we modified the packet for ALT coding by separating the sign bits in order to further simplify the decoding. Results show that the ALT coded images always yield higher PSNR as well as better visual qualities than the original UVLC under the same error environment.

6. REFERENCES

[1] S.Xue and B.Oelmann, “Alternating coding for Universal Variable Length Code,” to appear in the proceedings of *IEEE International Conference in Image Processing*, September 2003.

[2] 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.

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