An Efficient Error Detection Technique for VLC Coded Data

Upendra Singh Rajpurohit and Suneeta Agarwal

Department of Computer Science & Engg, Motilal Nehru National Institute of Technology, Allahabad
E-mail : upendra1984@gmail.com, suneeta@mnnit.ac.in

ABSTRACT

In order to achieve high compression in data, VLC coding is being most promptly used now days. This VLC technique provides a high degree of compression but has very poor error control; even a change in just one bit may lead to the corruption of whole file in the worst case. In this paper we discus a novel approach to solve this problem of error propagation and early detection of occurrence of errors. The algorithm works with out any preconditions and sends small amount of additional information with data. Implementation of algorithm confirms the decrement in erroneous data received and control on error propagation.

Keywords: *Algorithm, VLC, MPEG, Error detection*

1. INTRODUCTION

The VLC is widely used in source compression because of its high compression ratio. A lot of Audio, Video and Text compression algorithms are already being designed based on this coding scheme including H.26x, MPEG, AVI etc. But VLC coding has a major drawback of error propagation; even a single bit of error may lead to the corruption of whole file in the worst case. Thus transmitting a VLC coded file over a noisy channel without using an error detection algorithm is a worthless task. Although many VLC exhibits self synchronization property as discussed in [1], but the synchronization is achieved by a large number of additional data transmitted along with the original data and by assuming many preconditions.

Some algorithms have already been proposed in order to solve the said problem. A Reversible VLC (RVLC) based technique is discussed in [2]. But RVLC requires decompression of data from both ends and thus has a high complexity algorithm. The authors of [3] discuss a block interleaved error resilient coding scheme in which high value data can be prevented from errors by placing it ahead in block but it doesn't deal with the errors once they occur. An another approach to solve the problem as discussed in [5] is by sending some characters as it is in between along with there positions so that they can be cross checked at the receiver end for the synchronization.

In this paper we propose a new approach towards solving the problem of error detection. Our method doesn't require any known preconditions about data or the networks conditions, also requires very less amount of additional data for synchronisation purpose.

The paper is organised in three sections. In section two, we discus the problem of errors in VLC coded data, in section three we propose a solution to the problem and section four represent results which clearly indicates the superiority of our algorithm over other existing techniques.

2. ERRORS IN VLC CODED DATA

Most of the entropy encoding schemes use VLC codes such as Huffman codes, Reverse VLC, Universal VLC etc. all are known to be very sensitive for errors.

The sensitivity of errors lies in the fact that these codes are of variable length and use the prefix property for the decoding purpose. This prefix decoding of variable length codes is not suitable for decoding of erroneous data files, as it not only results in incorrect decoding of code words but also results in the loss of synchronization of prefix and thus leads to the incorrect decoding of subsequent words.

For example, suppose the characters along with their corresponding code words used to code a text as

Table 1

Character	Code Word
A	10
B	010
C	0110
D	1110
E	110

For example consider the given sequence of code words as under,

10011011011101001011100101l0

The above data has the following code sequence

10 0110 110 1110 10 010 1110 010 110

Upon correctly decoding following character sequence is obtained.

ACEDABDBE

Suppose a noise burst leads to two errors each of one bit in data (second and fifteenth bit from left changes from 0 to 1) at the time of transmission. The code sequence now will be

1**1**011011011101**1**0101110010110

The bold characters show the changed bits.

The given sequence will now have the codes like this

110 110 110 1110 110 10 1110 010 110

Upon decoding following character sequence is obtained.

EEEDEADBE instead of ACEDABDBE

Thus even a small amount of error is strong enough to corrupt the whole file.

The errors which can occur in VLC coded data files can be classified into various categories.

i. The errors which lead to invalid code word.

ii. The errors which lead to a valid code word at the moment but may lead to an invalid codeword later.

iii. The errors which propagate but resynchronize itself after some time.

The first two types of errors are considered as fatal errors and can be detected easily. But the problem is with the third type of errors as they are hard to detect. Although in real time applications the third category errors are of very less threat but still they must be detected as early as possible because it may be possible that they may corrupt a big chunk of file before getting resynchronized. Even in the case of fatal errors, we can't be sure whether the error has occurred at this point or had occurred previously. The method proposed here deals with all the three categories of errors.

3. ERROR DETECTION IN VLC CODES

The proposed Algorithm sends the whole data in blocks embedded with small amount of synchronization bits.

The sizes of these blocks are not fixed, they vary according to the detection of errors at the receiver end. In order to report the error back to the sender, the concept of negative acknowledgment is used. After receiving this acknowledgement the size of the block is reduced to half of current size otherwise doubled. This reduces the overhead communication cost of acknowledgment to a considerable extent. The algorithm works as follows:

The given data is first transmitted in blocks with the policy:

Say the largest code among all character is of length 'N', make the first macro block of size greater than or equal to N and transmit, upon successful transmission increase the size of block to double, and continue the process until a threshold is not reached or an error doesn't occur. On occurrence of an error the size of the block will be made equal to the last known good size i.e. the half of the current size, and transmit all data from that point onwards again.

Every block is ended with a specific ending for error detection purpose. It is known that mostly errors disturb the synchronisation of codes and causes error with a difference of at least one bit. This error can be detected at the end of the block.

The ending embedded in block consists of 0s sandwiched between two 1s. The number of 0s can vary from 0 to N-1.

Let BSIZE be the block size, and the last codeword ends at the bit number BSIZE, then the ending of the block will be "11", but if the last code word ends at position (BSIZE-REM), REM are the remaining bits in current block and the next symbol has code word of size greater than REM which cannot be fitted in the current block and thus it will go in the next block and the remaining bits of current block will be filled by extending 11 containing REM number of 0s in between these two 1s.

In general, the embedded ending 'X' will be the M number of zeros sandwiched between two 1s, where M = {0 .... N-1}. Errors can be detected by checking the pattern of X. The X when checked in advance will tell the ending point of last code word. The number of 0s =M in the ending symbol will tell that the ending point of the codeword is at position K-M+1, where K is the size of the block.

For example:

Consider Table 1, and the following string

10011011011101001011100101l0

So the codes were

10 0110 110 1110 10 010 1110 010 110

Here N=4, so size of block must be greater than or equal to 4.

Say Block Size =8, then as per the algorithm only first 2 words will fit into block 1

100110BB

2 bits are remaining (BB), but the next word can not be included into block 1 (110), so it will go into block 2. BB will be filled by 10 ending the block by 01

So the block transmitted is 1001101001

And ending X=1001 denotes that the last word ends at position 6, and thus can be checked for the verification of data received.

On correctly receiving the block the receiver doubles the size of the block to be receive, otherwise it will reset the size to the previous known good size, and sends a negative acknowledgement containing the block no. of corrupted block to the sender, the sender on receiving the acknowledgement resets the size to the previous best size and retransmit the block.

The Algorithm has advantages over the other proposed algorithms, as compared to [6], proposed algorithm preserves the beauty of VLC technique and doesn't loss any data which makes it suitable for any type of data Audio, Video, Text etc.

Also in addition to this we are using one additional bit at the end of each block for parity checking. This can directly detect the odd number of bit changes immaterial of whether the code is getting resynchronised or not.

In comparison to [5] the proposed algorithm requires very less amount of data to be sent and less amount of computation for the purpose of resynchronisation and error detection.

4. ALGORITHM

- Find the size of largest VLC codeword (N)
- Set MINSIZE and MAXSIZE,

  MINSIZE,MAXSIZE>=N
- Make Block size BSIZE=MINSIZE

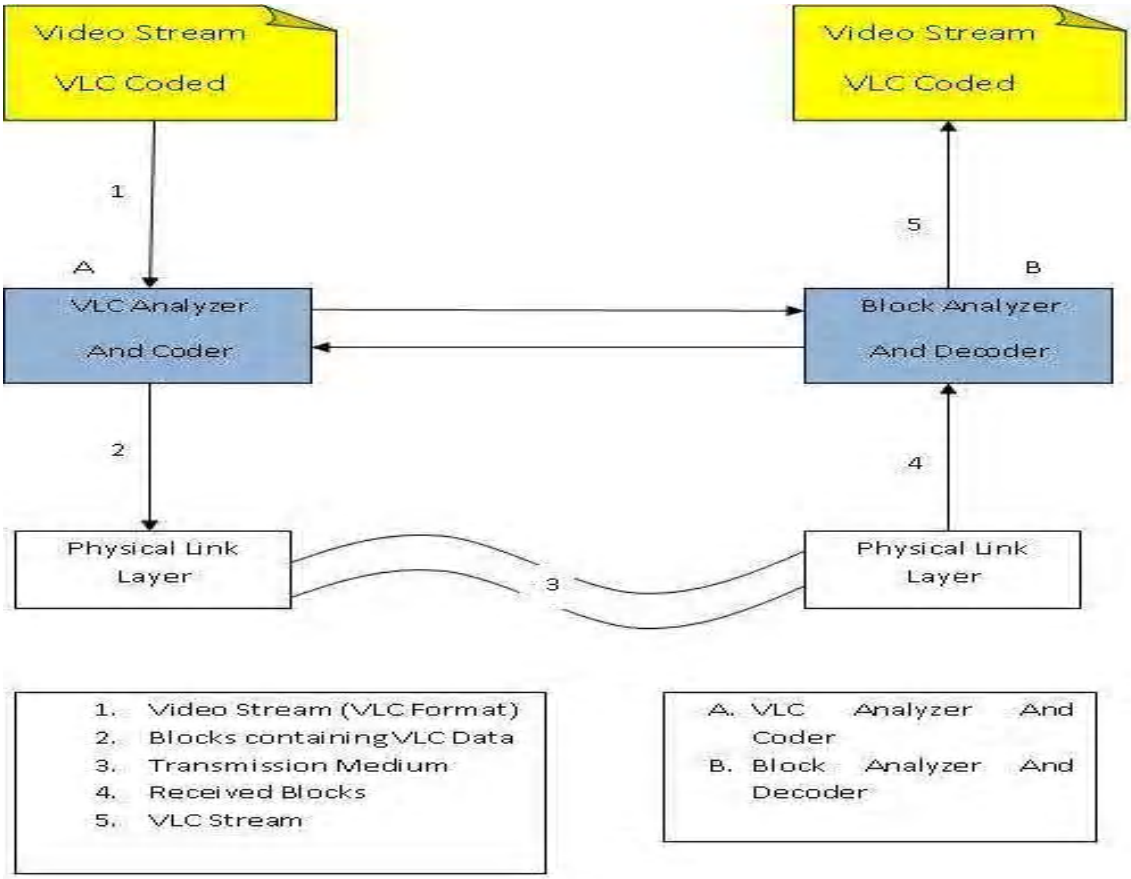

Fig. 1: Block Diagram of the system

- Until the completion of data
  - Make block of size BSIZE and serial number BNO
  - Transmit block over the physical channel
  - Make BSIZE=BSIZE*2
- If an acknowledgement is received
  - Make BSIZE = last known best size

5. IMPLEMENTATION AND RESULTS

In order to evaluate the effectiveness and performance of the proposed Algorithm, a lot of sample dataset have been implemented with different sizes of blocks and Huffman codes. In Figure 2 it is clear that the algorithm requires very less amount of extra data as compared to sending the check marks. In addition to minimizing error propagation effect, it provides high accurate error

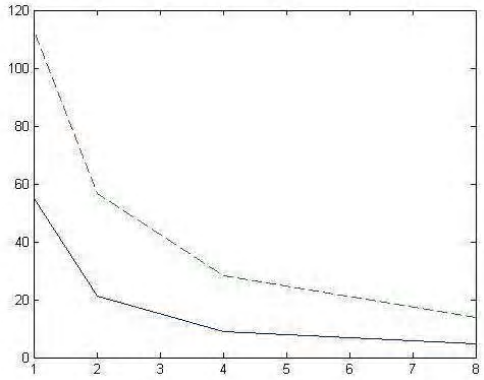

Fig. 2: The Amount of Extra Data Required

X-axis represents block size in the multiple of initial size, where initial size is greater than or equal to 'N'. Y axis represents % of extra data required. Solid line represents the data required in proposed algorithm. Dashed line represents the data required while sending the codeword with there respective positions.

detection capability In Figure 3 the Efficiency is shown in terms of the amount of correctly received data with respect to the block size. Efficiency obtained without implementing algorithm on corrupted data is as low as 36%. But after implementing the proposing algorithm we can obtain efficiency over and above 95%.

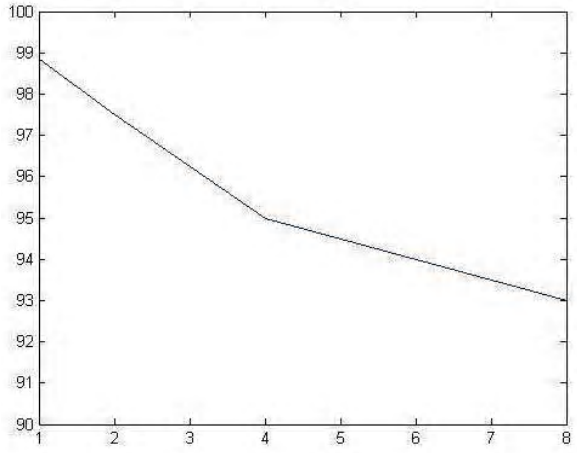

Fig. 3: Percentage of correct data received.

X-axis represents block size in the multiple of initial size, where initial size is greater than or equal to 'N'. Y axis represents % of correct data received

Also the algorithm requires less computing power to reorganize the compressed bitstream, which makes it suitable for real time application. Furthermore the algorithm has small redundancy and excellent error resilient capability.

6. CONCLUSION

In this paper we have proposed a novel error detection technique for VLC coded data over a noisy channel. We have proposed and implemented an algorithm to detect the erroneous blocks in VLC coded data and it was found that the proposed algorithm is sufficient to detect most of the erroneous blocks. The algorithm requires a considerably less amount of extra data for error detection purpose. Further the algorithm is totally lossless in no error condition which is very common in error coding techniques. Since the size of block is taken as variable rather than static thus it improves the transmission rate in no error condition.

REFRENCES

1. T. Ferguson and J. H. Rabinowitz, "Self-synchronizing huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-30, pp. 687–693, Jul. 1984.
2. Qiang Wang, Debin Zhao, Siwei Ma, Wen Gao, "An Enhanced Robust Entropy Coder for Video Codecs Based on Context-Adaptive Reversible VLC", *IEEE Data Compression Conference (DCC'07)*, 2007
3. Y. Fang, L. Yu, "Block-Interleaved Error-Resilient Entropy Coding", *IEEE transaction on Circuits and Systems*, pp. 53-56, May-2007
4. E. Khan, S. Lehmann, H. Gunji, and M. Ghanbari, "Iterative error detection and correction of h.263 coded video for wireless networks", *IEEE transactions on circuits and systems for video technology*, vol. 14, no. 12, Dec. 2004
5. H. Cai, B. Zeng, "A new, efficient, and flexible error detection approach for compressed visual contents", *IEEE transaction on Circuits and Systems*, vol. 03, no. 04, pp. III-909 – III-912, May-2004
6. H Nguyen, J Brouet, P Duhamel, "Robust and adaptive transmission of compressed video streams over EGPRS", *IEEE transactions on Consumer Communications and Networking Conference*, Page(s):320-324 5-8 Jan. 2004