A High Compression Deflate Algorithm for Video Stream

Mrs. T. Arumuga Maria Devi[1] Mr. S.Senthil Arumugam[2] Mr.K.Dinesh[3]

[1,2,3]Centre for Information Technology and Engg, Manonmaniam Sundaranar University, Tirunelveli.

Abstract
To reduce the redundancies in video frame sequence for the similarity between the current and the prior image, a new video coding algorithm based on zero-tree wavelet with deflate compression algorithm is presented in this paper. This algorithm is the combination of Huffman coding and LZ77 used in video streams. With this algorithm, we obtain a new compression algorithm which increases the compressed ratio than arithmetic coding.

Keywords:
LZ77, Huffman coding, deflate, Corner detection and Matching algorithm

1. Introduction

Video compression is the main application of compression of video frames efficiently which is efficient for the transmission and storage in databases. According to the features of star-field video, LIU[8] exploited a zero-tree wavelet algorithm with extremely high compression. In fact, LIU's simple difference method is not always successful for there is more complicate content in most video sequence. This paper improved LIU's motion detecting method using corner detection and matching algorithm [7]. This corner detection and matching algorithm gives an efficient result. After detecting the moving objects we are going to apply the deflate coding framework is presented which employs different strategy based on different video frames. Finally, the zero-tree wavelet coding method is used to for still image compression. With the algorithm we can realize the real-time deflate compression of video streams with high compression rate.
In section 2, the framework of the coding system is described. Then we introduce the motion matching method and compression scheme in section 3. And in section 4, the still image compression and edge modifying strategy are presented. Some experimental results and discussion are given in section 5 & 6.

2 System framework

The flow chart of coding and decoding is showed in Figure 1.
The input of the system is the color video frames captured by camera. The image is transformed from RGB color space to YUVcolor space, and then coded in YUV color space. To improve the compression the weight of YUV is assigned as 4:2:2 considering the features of human vision. For the first frame of the video sequence, code the whole image with zero-tree wavelet transform and arithmetic coding algorithm. For the latter frames, find the target areas and the motion parameters by motion detecting from the current frame and the prior frame, code the whole image or only code the areas with zero-tree wavelet transform where motion parallax exists, and then send the arithmetic coding stream and motion parameters.
The decoding process is adverse to the coding process.

3. Motion detecting and coding strategy

Wei-feng LIU et.al employed the corner detection and matching method [7] to calculate the motion parameters between current and prior frames. Then the current frame can be estimated combining the motion parameters and the prior frame. Comparing the current frame and its estimated frame, the target area that has distinct movement can be detected. Figure 2 shows the motion detecting example

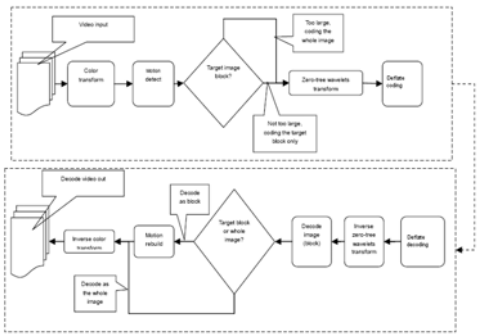


Figure 1. System Framework

If the total target area is small, the real-time compression can be improved highly by just coding the areas only. In fact, the total target area is not always small. When the total target area is large, the performance improves little by only coding the areas than coding the whole image. Sometimes coding the target area too large has more time cost than coding the whole image does. Therefore it is no need to always code the target area.

Manuscript received August 5, 2012
Manuscript revised August 20, 2012

We defined a ratio γ and a threshold $\gamma_{threshold}$ to decide when to code the target area or the whole image. Here

$$\gamma = \frac{the\_total\_target\_area}{the\_whole\_image\_area}$$

When γ is less than $\gamma_{threshold}$ code the target area only to get high compression. Otherwise, code the whole image. Figure 3 shows the different examples of the target area.

According to the definition $\gamma_{threshold}$ and γ above, $0 \le \gamma$ ($\gamma_{threshold}$) ≤ 1 .When $\gamma_{threshold}$ =0,that means code the whole image for every frame, the performance plays the same as the still image compression. When $\gamma_{threshold}$ =1that means always code the target area. The compression will be improved most but there may be some more time cost for some frames. So the $\gamma_{threshold}$ should be set based on different requirement. In experiment, we found that the compression can be improved 30%~50%without more time cost when we set $\gamma_{threshold}$ = 0.3.

4. Zero-tree wavelet and edge Modification

As an advanced transform technique wavelet transform is a method about temporal-frequency analysis, and is superior in local quality both in spatial field and frequency field. The wavelet coefficients of image have some merits on statistical quality (Figure 4. There are some coding methods using wavelet, such as DPCM [5] and VQ [6].

Zero-tree is a data structure, it is based on the hypothesis that if a wavelet coefficient at coarse scale is insignificant with respect to a given threshold T, then all wavelet coefficients of the same orientation in the same spatial location at finer scales are likely to be insignificant with respect to T. Empirical evidence suggests that this hypothesis is often true [3]. Consider the distributing character of the coefficients, the right scanning order should bring as many zero-tree roots as possible to compressing the code. Then the scanning order starts from coarse scale to finer scale (Figure 4). In other words, the child can't come into being before his parent.

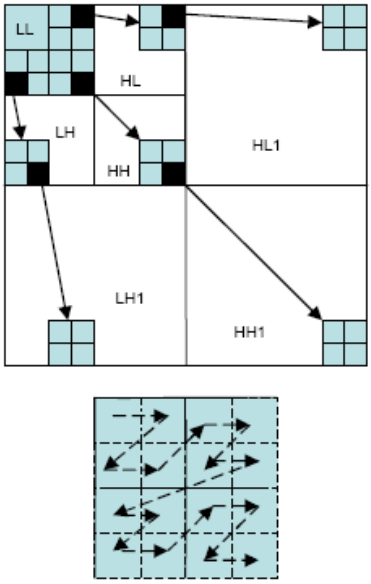


Figure 4. Parent-child dependencies of the sub-bands and the scanning order of the sub-bands

We think that the zero-tree method accord more with the character of the wavelet coefficients. Therefore the zero-tree code method [2][3] is employed to improve high compression. The area of target area detected is usually small, the decoding image block may have a little edge artifacts. It can be ignored in large images, but not allowed in small images. Here we adjust the target area before it is coded. Take off the edge artifacts when decoding the image block (Figure 5), then it can achieve the high fidelity.

Figure 5. The modifying of the edge: outer rectangle is the coding field; the inner is for rebuilding.

5. Deflate Compression algorithm

Deflate is a lossless data compression algorithm that uses a combination of the LZ77 algorithm and Huffman coding. Deflate is widely thought to be free of any subsisting patents, and at a time before the patent on LZW (which is

used in the GIF file format) expired, this has led to its use in gzip compressed files and PNG image and video files, in addition to the ZIP file format for which Katz originally designed it. [9]

A. Stream format

A Deflate stream consists of a series of blocks. Each block is preceded by a 3-bit header:
(1) 1-bit: Last block in stream marker:
(a) 1: this is the last-block in the stream.
(b) 0: there are more blocks to process after this one.
(2) 2-bits: Encoding method used for this block type:
(a) 00: a stored/raw/literal section follows, between 0 and 65,535 bytes in length.
(b) 01: a static Huffman compressed block, using a pre-agreed Huffman tree.
(c) 10: a compressed block complete with the Huffman table supplied.
(d) 11: reserved, don't use.
Most blocks will end up being encoded using method `10`, the *dynamic Huffman* encoding, which produces an optimized Huffman tree customized for each block of data individually. Instructions to generate the necessary Huffman tree immediately follow the block header.
Compression is achieved through two steps

- The matching and replacement of duplicate strings with pointers.
- Replacing symbols with new, weighted symbols based on frequency of use.

B. Duplicate string elimination

Within compressed blocks, if a duplicate series of bytes is spotted (a repeated string), then a back-reference is inserted, linking to the previous location of that identical string instead. An encoded match to an earlier string consists of a length (3–258 bytes) and a distance (1–32,768 bytes). Relative back-references can be made across any number of blocks, as long as the distance appears within the last 32 kB of uncompressed data decoded (termed the *sliding window*).

C.Bit reduction

The second compression stage consists of replacing commonly-used symbols with shorter representations and less commonly used symbols with longer representations. The method used is Huffman coding which creates an unprefixed tree of non-overlapping intervals, where the length of each sequence is inversely proportional to the probability of that symbol needing to be encoded. The more likely a symbol has to be encoded, the shorter its bit-sequence will be.
A tree is created which contains space for 288 symbols:

- 0–255: represent the literal bytes/symbols 0–255.
- 256: end of block – stop processing if last block, otherwise start processing next block.
- 257–285: combined with extra-bits, a match length of 3–258 bytes.
- 286, 287: not used, reserved and illegal but still part of the tree.

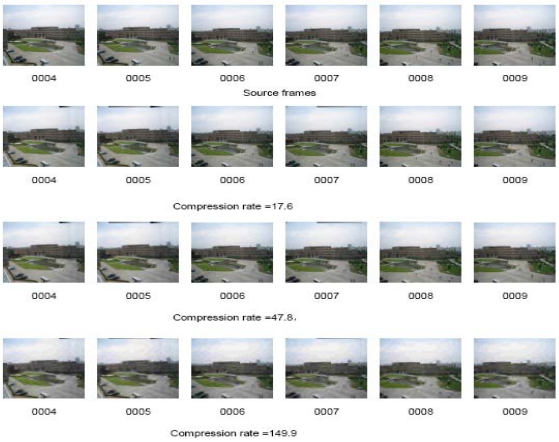
A match length code will always be followed by a distance code. Based on the distance code read, further "extra" bits may be read in order to produce the final distance. The distance tree contains space for 32 symbols:

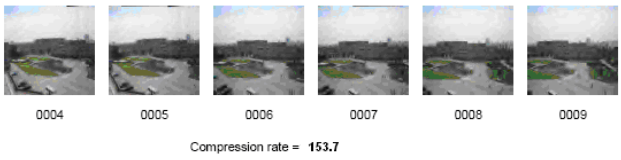
- 0–3: distances 1–4
- 4–5: distances 5–8, 1 extra bit
- 6–7: distances 9–16, 2 extra bits
- 8–9: distances 17–32, 3 extra bits
- 26–27: distances 8,193–16,384, 12 extra bits
- 28–29: distances 16,385–32,768, 13 extra bits
- 30–31: not used, reserved and illegal but still part of the tree.

D. Encoder/compressor

During the compression stage, it is the *encoder* that chooses the amount of time spent looking for matching strings. The zlib/gzip reference implementation allows the user to select from a sliding scale of likely resulting compression-level vs. speed of encoding. Options range from -0 (do not attempt compression, just store uncompressed) to -9 representing the maximum capability of the reference implementation in zlib/gzip.
Other Deflate encoders have been produced, all of which will also produce a compatible bitstream capable of being decompressed by any existing Deflate decoder. Differing implementations will likely produce variations on the final encoded bit-stream produced. The focus with non-zlib versions of an encoder has normally been to produce a more efficiently compressed and small encoded stream.

Some Experimental Results





6. Conclusion

It is presented to reduce the redundancies in video frame sequence for the similarity between the current and the prior image, a new video coding algorithm based on zero-tree wavelet with deflate compression algorithm is presented in this paper. This algorithm is the combination of Huffman coding and LZ77 used in video streams. With this algorithm, we obtain a new compression algorithm which increases the compressed ratio than arithmetic coding and the performance is more effective. In[8],compression rate-149.9 was achieved. Here, proposed technology achieved compression rate-153.3.Thus the proposed technique is achieved the high compression rate.

References

[1] A. Murat Tekalp, Digital Video Processing, Electronic Industries & PRENTICE HALL publishing company, Beijing, 1998

[2] Jerome M. Shapiro, Embedded Image Coding Using Zerotrees of Wavelet Coefficients, IEEE TRANSACTIONS ON SIGNAL PROCESSING. VOL 41. NO. 12. DEC. 1993

[3] Embedded Zerotree Wavelet Encoding C C. Valens1999 c.valens@mindless.com

[4] Howard, P.G.; Vitter, J.S., Arithmetic coding for data compression , Proceedings of the IEEE, Volume: 82 , Issue: 6 , June 1994

[5] Liu Bin, Tian Jinwen, Zhang Tianxu, An Image Lossless Compression Algorithm Based on Integer Haar Wavelet Transform and DPCM, Journal of Huazhong University of Science & Technology, Sep. 1999

[6] He Li, Wang Yanping, Image Compression Using Wavelet Transform and Constraint Matrix, Acta Electronica Sinica, Vol.23, No.4, Apr. 1995

[7] Zhou Peng, Tan Yong, Xu Shoushi, A New Method of Image Registration Based on Corner Detection, Journal of University of Science and Technology of China, vol.32, No.4, Oct. 2002

[8] LIU Weifeng and WANG Zengfu, " A high compression algorithm for video stream" , 2008 Congress on Image and Signal Processing.

[9] P. Deutsch, DEFLATE Compressed Data Format, Aladdin Enterprises Category: Informational May 1996

[10] Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the Institute of Radio Engineers, September 1952, Volume 40, Number 9, pp. 1098-1101.

[11] Ziv J., Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

[12] Gailly, J.-L., and Adler,M., ZLIB documentation and sources, available in ftp://ftp.uu.net/pub/archiving/ zip/doc/

[13] Gailly, J.-L., and Adler, M., GZIP documentation and sources, available as gzip *.tar in ftp:// prep.ai.mit.edu/pub/gnu/

[14] Schwartz, E. S., and Kallick, B. "Generating a canonical prefix encoding." Comm. ACM, 7,3 (Mar. 1964), pp. 166-169.

[15] Hirschberg and Lelewer, "Efficient decoding of prefix codes," Comm. ACM, 33,4, April 1990, pp. 449- 459.

T. Arumuga Maria Devi received B.E. Degree in Electronic and Communication Engineering from Manonmaniam Sundaranar University, Tirunelveli India in 2003, M.Tech degree in Computer and Information Technology from Manonmaniam Sundaranar University, Tirunelveli, India in 2005. Currently, she is doing Ph.D in Computer and Information Technology and also the Assistant Professor of Centre for Information Technology and Engineering of Manonmaniam Sundaranar University. Her research interests include Signal and Image Processing, Multimedia and Remote Communication.

S.Senthil Arumugam received B.E Degree in Computer Science Engineering from Anna University, Chennai, India in 2007, Currently he is doing M.Tech degree in Computer and Information Technology from Manonmaniam Sundaranar University. His research interests include Signal and Image Processing and Networking..

K.Dinesh received B.Tech Degree in Information Technology and Management from SASTRA University, Tanjavoor, India in 2009, Currently he is doing M.Tech degree in Computer and Information Technology from Manonmaniam Sundaranar University. His research interests include Signal and Image Processing and networking