Improving LZW image compression

Article · August 2010

CITATIONS
8

READS
509

4 authors, including:

Sawsan Abu Taleb
Al-Balqa' Applied University
4 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Asma'a Khtoom
Al-Balqa' Applied University
5 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Islah Gharaibeh
Al-Balqa' Applied University
5 PUBLICATIONS **18** CITATIONS
SEE PROFILE

European Journal of Scientific Research
ISSN 1450-216X Vol.44 No.3 (2010), pp.502-509

http://www.eurojournals.com/ejsr.htm

Improving LZW Image Compression

Sawsan A. Abu Taleb
Prince Abdullah Bin Ghazi Faculty of Science & Information Tech
Al-Balqa' Applied University
E-mail: swsn202@gmail.com

Hossam M.J. Musafa
Computer Center, Al-Balqa' Applied University
E-mail: hasa.mustafa@gmail.com

Asma'a M. Khtoom
Prince Abdullah Bin Ghazi Faculty of Science & Information Tech
Al-Balqa' Applied University
E-mail: asmakhtoom@yahoo.com

Islah K. Gharaybih
E-mail: islahgh@yahoo.com

Abstract

Previous lossless image compression techniques seek the smallest possible image storage size for a specific level of image quality; in addition, dictionary-based encoding methods were initially implemented to reduce the one-dimensional correlation in text. This paper proposes improving compression method which uses a bit plane slicing and adaptive Huffman with a Lempel-Ziv-Welch (LZW) dictionary. When LZW compression used to compress image, the limitation of the type of image and the number of colors must be considered and LZW compression will be used for some cases. The limitation will be decreased by using bit plan slicing for colored and gray scale images. The compression ratio of proposed method is better than the standard LZW method by 55.6% for colored image, and 102% for gray scaled images. Experimental results for the proposed method outperform the existing methods for a large set of different type of images.

Keywords: (LZW) Dictionary compression, Adaptive Huffman, Image Compression, lossless image coding, Bit Plane Slicing.

1. Introduction

First of all; there are two kinds of compression coding schemes: lossless and lossy compression. Lossless compression requires that original data be reconstructed without any distortion after inverse operation. On the other hand, lossy compression does not guarantee that the original and recovered data are identical, but it often provides better performance (in terms of compression ratio) than lossless methods. Lossy compression can be applied to voice, image, and video media applications because

they do not necessarily require perfect recovery if the reconstructed quality is good enough for human perception [2][11][13].

Lossless data compression algorithms mainly include Lempel and Ziv (LZ) codes [6], Huffman codes [12], and others such as [5] and [11].However, in this paper, we will focus on the topic of LZW and adaptive Huffman lossless compression. The LZW data compression algorithm is a powerful technique for lossless data compression that gives high compression efficiency for text as well as image data [4]. In this paper, we don't consider the Huffman code due to its inherent feature of being required to know a priori probability of the input symbols. Instead, we consider a variant of the Huffman code called the adaptive Huffman code or the dynamic Huffman code [3], which does not need to know the probability of the input symbols in advance.

1.1. LZW Compression

LZW is named after Abraham Lempel, Jakob Ziv and Terry Welch [4], the scientists who developed this compression algorithm [6][14]. It is a lossless 'dictionary based' compression algorithm. Dictionary based algorithms scan a file for sequences of data that occur more than once. These sequences are then stored in a dictionary and within the compressed file, references are put where-ever repetitive data occurred. LZW compression replaces strings of characters with single codes. The code that the LZW algorithm outputs can be of any arbitrary length, but it must have more bits in it than a single character. The first 256 codes (when using eight bit characters) are initially assigned to the standard character set. The remaining codes are assigned to strings as the algorithm proceeds. The sample program runs as shown with 12 bit codes [6].

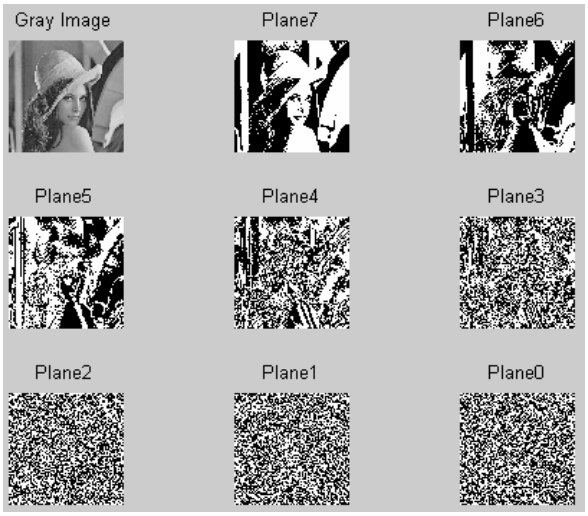
LZW is an adaptive technique. As the compression algorithm runs, a changing dictionary of (some of) the strings that have appeared in the text so far is maintained. Because the dictionary is pre-loaded with the 256 different codes that may appear in a byte, it is guaranteed that the entire input source may be converted into a series of dictionary indexes. If "A" and "B" are two strings that are held in the dictionary, the character sequence "AB" is converted into the index of "A" followed by the index of "B". "A" greedy string matching algorithm is used for scanning the input, so if the first character of "B" is "x", then "Ax" cannot be an element of the dictionary. The adaptive nature of the algorithm is due to that fact that "A" "x" is automatically added to the dictionary if "A" is matched but "A" "x" is not matched [13].

This means codes 0-255 refer to individual bytes, while codes 256-4095 refers to substrings. Thus, there are Advantages and disadvantages of LZW compression; the size of files usually increases to a great extent when it includes lots of repetitive data or monochrome images. LZW compression is the best technique for reducing the size of files containing more repetitive data [6] [13]. LZW compression is fast and simple to apply. Since this is a lossless compression technique, none of the contents in the file are lost during or after compression. The decompression algorithm always follows the compression algorithm. LZW algorithm is efficient because it don't need to pass the string table to the decompression code. The table can be recreated as it was during compression, using the input stream as data. This avoids insertion of large string translation table with compression data.

1.2. Bit-Plane Slicing

Highlighting the contribution made to the total image appearance by specific bits. Assuming that each pixel is represented by 8-bits, the image is composed of eight 1-bit planes. Plane (0) contains the least significant bit and plane (7) contains the most significant bit as you see in figure (1). Only the higher order bits (top four) contain the majority visually significant data. The other bit planes contribute the more subtle details. It is useful for analyzing the relative importance played by each bit of the image [8] [13].

Figure 1: An 8-bit gray-scale lena image of size 256 X 256 pixels. Each bit plane is a binary image



1.3. Adaptive Huffman Coding

The Huffman algorithm requires both the encoder and the decoder to know the frequency table of symbols related to the data being encoding, so it requires the statistical knowledge which is often not available. Even when it is available, it could be a heavy overhead especially when many tables had to be sent .To avoid building the frequency table in advance, an alternative method called the Adaptive Huffman algorithm [3] that was first conceived independently by Faller (1973) and Gallager (1978). It allows the encoder and the decoder to build the frequency table dynamically according to the data statistics up to the point of encoding and decoding.

In Adaptive Huffman Coding, the statistics are gathered and updated dynamically as the data stream arrives [3][13].

```
Encoder
Initial_code();
while not EOF
{
get(c);
encode(c);
update_tree(c);
}
```

Initial code assigns symbols with some initially agreed upon codes, without any prior knowledge of the frequency counts. Update tree constructs an Adaptive Huffman tree. It basically does two things:

a) Increments the frequency counts for the symbols (including any new ones).
b) Updates the configuration of the tree.

The encoder and decoder must use exactly the same initial code and update tree routines.

2. The Problem Definition

LZW compression works best for files containing lots of repetitive data. This is often the case with text and monochrome images. Files that are compressed but that do not contain any repetitive information at all can even grow bigger. Therefore; LZW compression can not be used in variant color images or gray scale image or natural images that contain shadows or gradient, this is due to the start of The code table is initialized with all single-character strings (256) and more in variant color images. LZW places

longer and longer repeated entries into a dictionary, and then the code of the string will be larger than two bytes. So the compression ratio is near to zero.

3. Related Works

Before presenting the improvement of LZW for image compression, we will discuss both the features and limitations of the previous algorithms. Akimov, Kolesnikov and Fränti[1] worked in lossless compression of color map images by context tree modeling, they propose an n-ary context tree model with incomplete tree structure for the lossless compression of color map images. The proposed n-ary incomplete context-tree-based algorithm outperforms the competitive algorithms (MCT, PWC) by 20%, and by 6% in the case of full context tree (CT) algorithm. The proposed algorithm has some disadvantages:

- Apply on map images that have few colors.
- The compression method was successfully applied to raster map images up to 67 colors. If the overwhelming memory consumption can be solved in the case of images with a larger number of colors.

Horspool [9] have selected two ways of improving LZC (the UNIX compress command). One, a method of loading the dictionary at a faster rate, has not been used before. The other, a method to phase in increased lengths of binary numbers gradually, is not original but is not currently used with LZC. The Disadvantages the algorithm; apply on English text files and C source code file that have high redundancy.

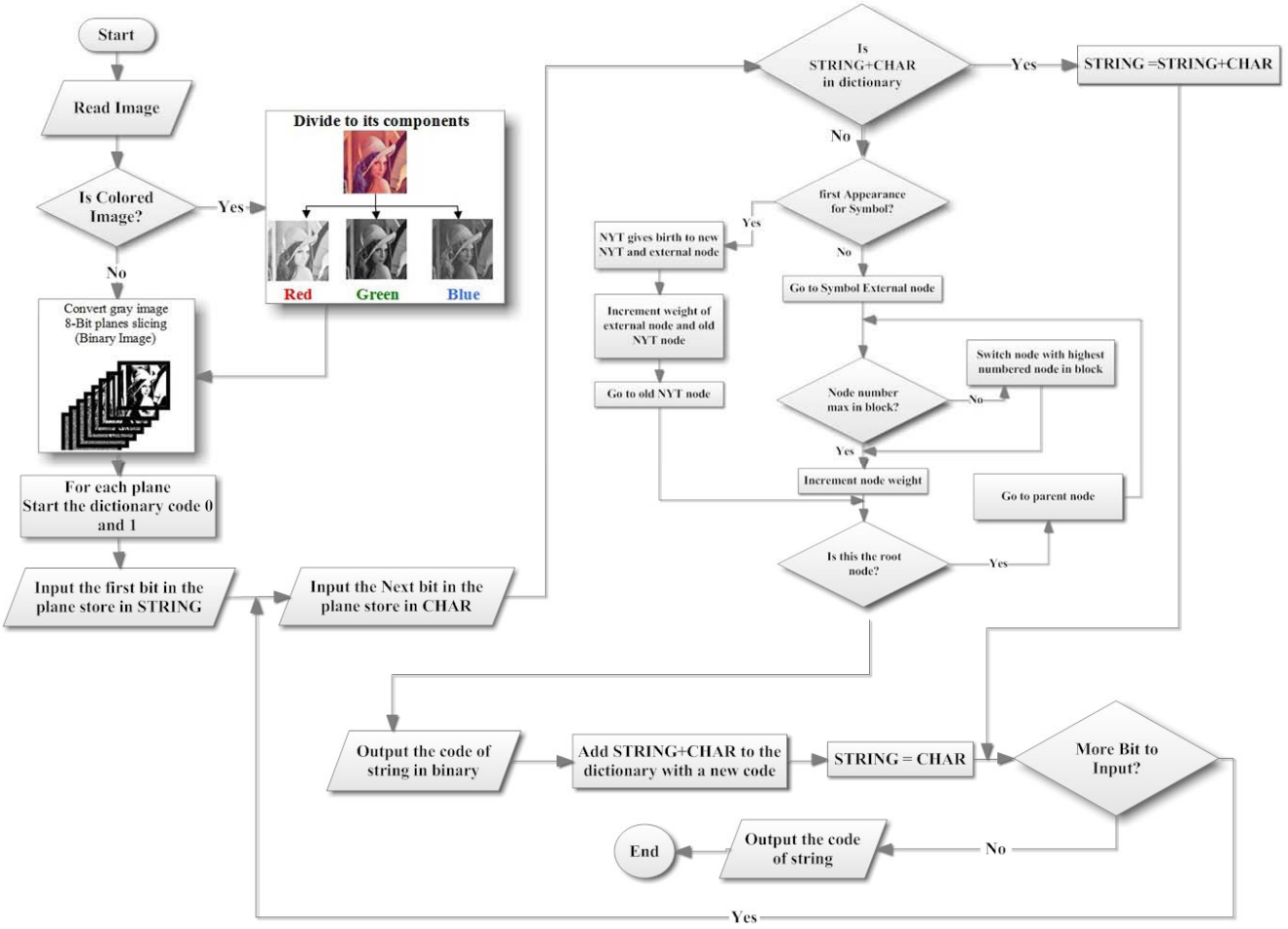
Cui [15] presented a new LZW compression algorithm that increases the throughput and improves the compression ratio simultaneously. The key of the proposed algorithm is designing adaptive preprocessor which decreases correlation between original input data block. A parallel VLSI architecture for the new LZW compression processor is proposed. The architecture is based on a parallel dictionary set that has the capability of parallel search. A test suite consisting of six text sources and six image sources is applied to the proposed architecture. The hardware cost is the first disadvantage. Moreover; the algorithm tested in six images only, which have high redundancy and the compression ratio using standard LZW is greater than two.

4. Improving LZW

At each step the dictionary is searched to find the longest matching phrase which is a prefix of the input data. The index of the phrase is an output of the compressed encoding, and a new entry, consisting of the next symbol from the input concatenated to the end of the phrase just matched, is inserted into the dictionary. Coding continues in this sequence, restarting each time from the last unmatched symbol in the input. Expansion of LZW coded data starts with the initial dictionary used for compression and increase redundant data in image and decrease number of bits for output code.

The proposed method improves LZW compression in the following ways. First, slice the gray-scale images into eight binary (monochrome) images by using bit-plane slicing. The colored images are typically represented by the tristimulus red, green, and blue signals each of which is gray scale image will be sliced into eight binary (monochrome) images by using bit-plane slicing. The separation of the input image can be done through color separation or through semantic separation [7][10]. The generated binary images contain redundant bits, which will increase the compression ratio. Because the number of color decrease to two colors black (the value is 0) and white (the value is 1).

Figure 2: Improving LZW compression flowchart



Second, we initialize LZW dictionary with two characters "0" that represent zero values for black color and "1" that represent one value for white color in monochrome image instead of (256) characters of the underlying character set. Third, each output code in the dictionary associates a frequency counter to phase in binary codes progressively using Adaptive Huffman algorithm to decrease the number of bits. This way a continuous adaption will be achieved and local variations will be compensated at run time. The three methods are shown in the figure (2).

As you see in figure (2) the first step in improve LZW compression read the image then check if color or gray scale image because the color image will be divided into three components: red, green and blue before dividing it to (8) binary images. For each of binary image 2D matrix will be converted to vector, and deal with the two values (0 and 1) as a character, then, standard LZW compression is applied. The output code of string using Adaptive Huffman Algorithm to achieve improvement from the binary code instead of decimal number that produced by the standard LZW algorithm.

5. Results

In Order to show the effect of 8-bit planes slicing of image on compression ratio, some experimental results are given in Table (1) and Table (2). The "File Size1" column corresponds to the file size after implementing standard LZW. Also the "Compression ratio1" column corresponds to the ratio when applying standard LZW compression. The two columns "File Size2" and "Compression Ratio2"

correspond to improvement method of LZW compression using 8-bit planes slicing and Adaptive Huffman algorithm. The results show that compression ratio of all images are improved. The amount of improvement depends greatly on the nature of the image; for files with very little short-range correlations between intensities, such as coin image, the improvement is non-existent or negligible, but for images containing variant of colors; the improvement are significant.

As another means of displaying the compression ratio improvement, the compression ratio for gray scale images was approximately 102% over standard LZW algorithm, and for colored image it was 55.6%. Experimental results indicate that compression ratio is improved, depending on the nature of the image file. Figure (1) and figure (2) show that compression with 8-Bits planes slicing and adaptive Huffman has a consistent advantage over standard LZW. Note that Compression ratio =B0/B1, where B0 is the number of bits before compression, and B1 is the number of bits after compression.

Table 1: experiment results for gray-scale images

Gray Scale Image					
Image Name	**File Size (no.bytes)**	**File Size1 (no.bytes)**	**File Size2 (no.bytes)**	**Compression Ratio 1**	**Compression Ratio 2**
CameraMan	16384	14522	7642	1.128219	2.143941377
Lena	16384	15767	8923	1.039132	1.83615376
Xray	16384	13812	6987	1.186215	2.344926292
Storm	16384	15815	5564	1.035979	2.944644141
Coin	16384	7802	4790	2.099974	3.42045929
Head	16384	15705	7877	1.043235	2.079979688
Barbara	65536	62782	33730	1.043866	1.94295879
Boat	65536	61764	30668	1.061071	2.136950567
Trees	65536	58026	26750	1.129425	2.449943925
Peppers	65536	61146	27001	1.071795	2.427169364
Lena	65536	59013	30943	1.110535	2.117958828
Average	**38725.8**	**35104.9**	**17352.2727**	**1.177222**	**2.349553275**

Figure 3: Improving LZW algorithm for gray-scale images

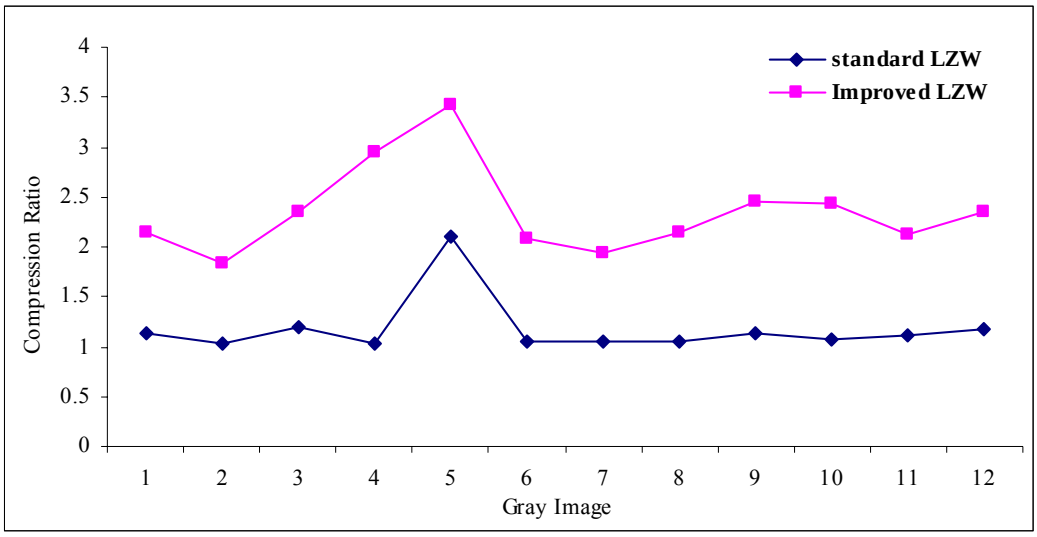
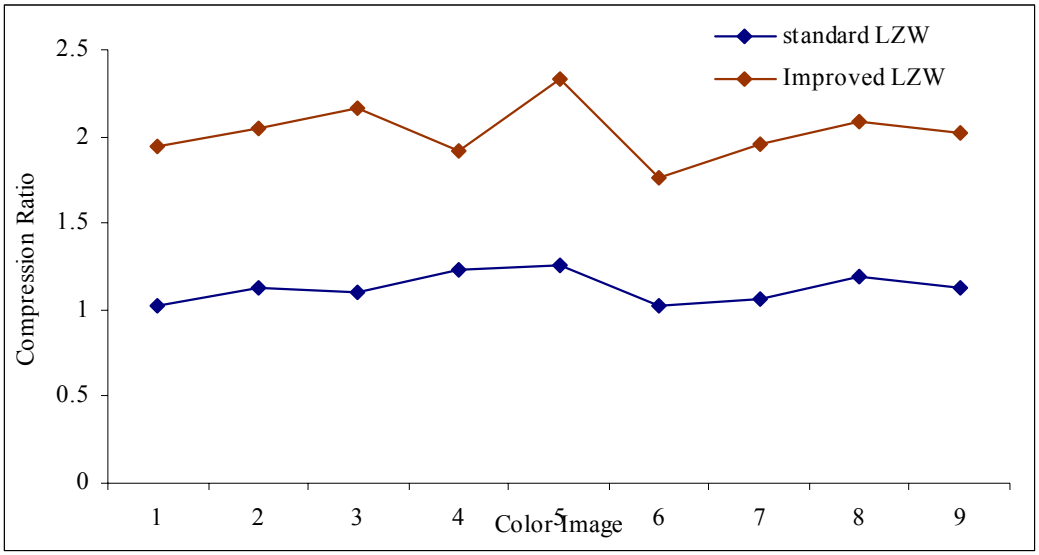


Table 2: Experiment results for colored images

True Color Scale Image					
Image Name	**File Size (no.bytes)**	**File Size1 (no.bytes)**	**File Size2 (no.bytes)**	**Compression Ratio 1**	**Compression Ratio 2**
lenacolor	196608	180739	101508	1.0236	1.936871971
Treescolor	196608	175095	96032	1.122864731	2.047317561
pepperscolor	196608	179179	91004	1.097271444	2.160432508
Housecolor	196608	159373	102322	1.233634304	1.921463615
Girlcolor	196608	157099	84290	1.251491098	2.332518685
Natural	135090	131443	76876	1.027745867	1.757245434
Boatcolor	196608	185072	100734	1.062332498	1.951754125
Barbara	196608	165118	94346	1.190712097	2.083903928
Average	**188918.25**	**166639.75**	**93389**	**1.126206505**	**2.023938478**

Figure 4: Improving LZW algorithm for colored images



6. Summary and Conclusions

Obviously, the compression ratio is enhanced by coding technique. It's not easy to find implementable methods for improving the compression ratio of LZW, especially when we deal with images that have variant color and large size. We have selected 8-bits planes slicing to increase the redundant data and adaptive Huffman to decrease the number of bits of output code are the way of improving LZW. In addition, it initialize LZW dictionary with two characters (0) and (1) instead of (256) characters.

References

[1] A. Akimov, A. Kolesnikov, and P. Fränti, "Lossless Compression of Color Map Images by Context Tree Modeling", IEEE Trans. Image Processing, Vol. 16, No. 1, January 2007.

[2] Bell, T.G., Cleary, J.G., and Witten, I.H. "Text Compression", Prentice-Hall, Englewood Cliffs, NJ (1990).

[3] D. E. Knuth, “Dynamic Huffman coding", J. Algorithms, vol. 6, pp.163–180, 1985.

[4] H. K. Reghbati, “An Overview of data compression techniques”, Computer, Vol.14, No. 4, pp.71-76, July 1981.

[5] J. L. Núñez and S. Jones, “Gbit/s lossless data compression hardware”, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 11, no. 3, pp. 499–510, Jun. 2003.

[6] J. Ziv and A. Lempel, “A universal algorithm for sequential data compression", IEEE Trans. Inf. Theory, vol. IT-23, no. 3, pp. 337–343, Mar.1977.

[7] P. Kopylov and P. Fränti, “Compression of map images by multilayer context tree modeling”, IEEE Trans. Image Process., vol. 14, no. 1, pp. 1–11, Jan. 2005.

[8] Rafael C. Gonzale and ,Richard E. Woods ,Digital Image Processing ,3rd edition, Prentice Hall, Upper Saddle River, New Jersey , 2008.

[9] R. Nigel Horspool ,"Improving LZW", Dept. of Computer Science, University of Victoria P.O. Box 3055, Victoria, B.C., Canada V8W 3P6

[10] S. Forchhammer and O. Jensen, “Content layer progressive coding of digital maps", IEEE Trans. Image Process., vol. 11, no. 12, pp. 1349–1356, Dec. 2002.

[11] S. Khalid," Introduction to Data Compression", 2nd edition, San Mateo, CA: Morgan Kaufmann, 2000.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms", 2nd ed. New York, McGraw-Hill, 2001.

[13] Ze-Nian Li and Mark S. Drew, Fundamentals of Multimedia, Prentice Hall, Upper Saddle River, NJ, 2004.

[14] Ziv, J, and Lempel, A. “Compression of Individual Sequences via Variable-Rate Coding", IEEE Trans. On Inf. Theory IT-24, 5 (Sept. 1978), pp. 530-536.

[15] Wei Cui, "New LZW Data Compression Algorithm and Its FPGA Implementation", School of Information Science and Technology, Beijing Institute of Technology, Beijing, 100081, China.