U.P.B. Sci. Bull., Series C, Vol. 71, Iss. 2, 2009 ISSN 1454-234x

LOSSLESS COMPRESSION TOOL FOR LIMITED NUMBER OF COLORS

Radu RĂDESCU[1]

Obiectivul utilitarului ICompress, descris în acest articol, este studiul în condiții reale al compresiei secvențelor masive de date având un număr limitat de culori. Datorită faptului că standardele actuale nu oferă suficientă flexibilitate, precum şi a implementării unor rutine de compresie şi decompresie pentru aceste standarde, care nu diferă foarte mult față de aplicațiile existente, articolul propune crearea unor noi formate de fişiere în scopul implementării compresiei fără pierderi. Algoritmii de compresie studiați sunt LZW şi RLC, pentru fiecare în parte introducându-se un format propriu şi studiindu-se performanțele în cazul unor imagini medicale.

The objective of the ICompress tool is to study the compression behavior in real situations with large data sequences for images with limited number of colors. Because the existing standards do not offer enough flexibility, and the implementation of some compression and decompression routines for these standards would not differ very much from the existing applications, it was chosen the creation of new file formats in order to apply the lossless compression. The studied compression algorithms are LZW and RLC, for each one introducing a file format and studying the performance in the case of some medical images.

Keywords: lossless compression, statistical algorithms, dictionary-based methods

1. Introduction

The ICompress application carries out the encoding for images with at most as 255 colors. The LZW coding [1]-[4] is inspired by the one used for the GIF standard [3], attaining a better packing of the factors compared to the GIF format. The RLC encoding [1]-[4] is a common algorithm, which aims the replacement of the repeated character sequences. Details of the coding method will be provided in the following. As output, the application produces two file types, named after the encoding type used to make them. The LZW type file is aligned at 32 bits; it has a header with the coding and image parameters, and a color palette [5]. The same idea is found at the RLC format, except the 32 bits alignment, which is not necessary [6].

[1] Reader, Applied Electronics and Information Engineering Department, University POLITEHNICA of Bucharest, Romania, rradescu@gmail.com

As source, the program accepts BMP files, represented on 8, 16 or 32 bits, with the condition that these files will contain less than 255 colors. If only images with a large number of colors, ICompress has an algorithm to reduce the color depth [7]. In addition, the coded files can be loaded and viewed, being converted automatically to the BMP format for memory storage.

2. Implementing the compression

A general problem regarding the image processing is the image run through to find the exact number of different colors.

This is the algorithm: the colors are stored in a table, which is built as the image is run through pixel by pixel. If the color of the pixel exists in the table, it is ignored, and if not, it is added to the table. This kind of simple algorithm has a major disadvantage from the speed point of view. A simple computation shows that for an image with 50,000 colors, in a common 1024×768 pixels format, a 50,000-positions vector will be made which needs to be run through more or less entirely, for the majority of the 786,432 pixels. The problem can be solved using advanced methods to run through the image or data storage, so that the search time for the color table can be reduced substantially.

For the implementation of the ICompress application, the second solution was chosen. There are many methods of fast search, like the binary trees or map type structures. The MFC CMap class was used, which implements a map structure and has a very efficient search algorithm like *Hash Table*.

The image is first run through and brought to a brute form and the color table is made up. The encoder runs over this brute form, coding the color table identifiers. The pointers are ASCII characters with codes form 1 to 255. The problems related strictly to the LZW encoding have been the dictionary search speed and the optimization for writing of factors in file. If the first problem was solved using a map dictionary implemented with CMap, the second one is more complicated. Characteristic to the LZW coding is the fact that it provides a series of words that can be binary represented with different lengths, assigned by the current number of inputs in the dictionary for building of the code words.

If a fixed format for the writing of data would be used, adequate from point of view of the maximum inputs in the dictionary (e.g., for 4.096 WORD positions for 16 bits), the coding would not be efficient. Thus for a word written at the beginning of the process, when the dictionary is not full, many bits would by redundant. From this situation, the necessity of binary writing the code words emerges.

In the application, the factors are first binary written using consecutive moves in a DWORD (double word with 32 bits). When the DWORD space is finished, the current DWORD is written in the file, and the writing of code words continues in another DWORD. At the decoding, the information is read DWORD by DWORD, and the code words are extracted with the use of some bit wise masks.

A problem related to the string storing in the CMap of the code words has appeared because of the impossibility of using the character with the ASCII code 0 – character used for the end of strings. For this reason, it was accepted that the images would have at most 255 colors and the indexes from the color table would begin with 1. A possible solution is to store the exact string length, but this would lead to a performance decrease.

For RLC, the encoding is made at character level, and the 0 character is used as an indicator for the encoding presence, so it cannot take part in the source symbols. A coded sequence is composed of the 0 character, indicator of the encoding, and then a character whose ASCII code represents the number of repetitions in the source sequence of the third character, which is the encoded character. Only sequences for which the compression is effective are coded (larger than 3 characters and shorter than 255 characters, the maximum ASCII code).

To have an encoding flexibility, in the case of LZW, a variable size dictionary is used, with 2,048, 4,096 or 8,192 positions, and in the case of RLC – two ways to run through the image: up and down or left to right, which exploits in two different ways the image correlation. The algorithm to reduce the number of colors is sufficiently effective without high expectations, its intentions being strictly for use. The algorithm is based on the nearest color method, computed based on the mean square method. The generated color palette is a joint one, including 128 standard colors, allocated equally in the color space, and other 127 colors calculated based on the bar graph of the image.

Because the search of the nearest color for every pixel in a 255-position map is time consuming (even when the Cmap is used), 3 maps with balanced colors were used. Therefore, the colors are divided between the three maps, depending on the distance to the three base colors: R, G, and B. When the picture run through is performed, the nearest value is searched in the nearest map. It is also specified an acceptable step for the difference of two colors, in order to avoid the exhaustively search for every pixel in the map.

3. Experimental results

A study was made concerning the effect of implemented algorithms on images with different correlations and different number of colors. Two pairs of images were chosen (one with a large color dispersion, with different sizes, and the other with large areas of the same color, also with different sizes) and their number of colors was reduces gradually from 255 to 128 and finally to 16 colors.

In order to have an accurate image of the performances, the original BMP file and the same file compressed with GIF have been compared to the files obtained with the ICompress application. A usual compression software was used, RAR, with the best compression method offered by it.

The experimental results are presented in Tables 1, 2 and 3.

Table 1

The dimensions of graphic compressed formats (B) for images with 255 colors

255 colors	NMR 600×600	Angiographies 600×600	Tomographies 600×600	Ultrasounds 540×405	X rays 540×405
BMP	361,078	361,078	361,078	219,778	219,778
LZW	150,951	258,532	130,190	62,608	100,527
RLC	272,950	356,454	201,309	96,037	212,335
RAR	120,595	209,811	110,156	53,667	84,009

Table 2

The dimensions of graphic compressed formats (B) for images with 128 colors

128 colors	NMR 600×600	Angiographies 600×600	Tomographies 600×600	Ultrasounds 540×405	X rays 540×405
BMP	180,118	180,118	180,118	110,278	110,278
LZW	81,094	101,070	62,770	27,782	49,582
RLC	203,886	268,667	149,215	62,007	140,527
RAR	61,001	78,773	47,664	22,802	38,536

Table 3

The dimensions of graphic compressed formats (B) for images with 16 colors

16 colors	NMR 600×600	Angiographies 600×600	Tomographies 600×600	Ultrasounds 540×405	X rays 540×405
BMP	180,118	180,118	180,118	110,278	110,278
LZW	81,094	101,070	62,770	27,782	49,582
RLC	203,886	268,667	149,215	62,007	140,527
RAR	61,001	78,773	47,664	22,802	38,536

4. Conclusions

One can see that the ICompress program gives a superior compression compared to GIF, mostly because of a better saving of coded words in files and because of the fact that it builds a color palette with the size obtained by the exact colors in the image.

The ICompress application has an advantage before the GIF compression as the file size is growing and the color number is decreasing. The particular case of 16 color files accentuates the major deficiency of the ICompress program. Because the 0 index is not used – by the above-mentioned reasons – the compression is affected and the primary indexes of the colors from the table need an extra bit comparing to the indexes used by the GIF format.

For images with 15 colors or less, the ICompress application has better results than GIF, but the above-mentioned problem appears for black and white (1 bit) images.

The usual archiver (RAR) proved to be the best from the point of view of the compression rate. The reason is the combination of statistic and adaptive methods (LZW, Huffman).

One can observe (see Figures 1 and 2) that the result does not depend very much on the number of colors from the input image (an effect of the general character of the encoding) but on the size of the source file. From the point of view of the compression time, this one is the slowest.

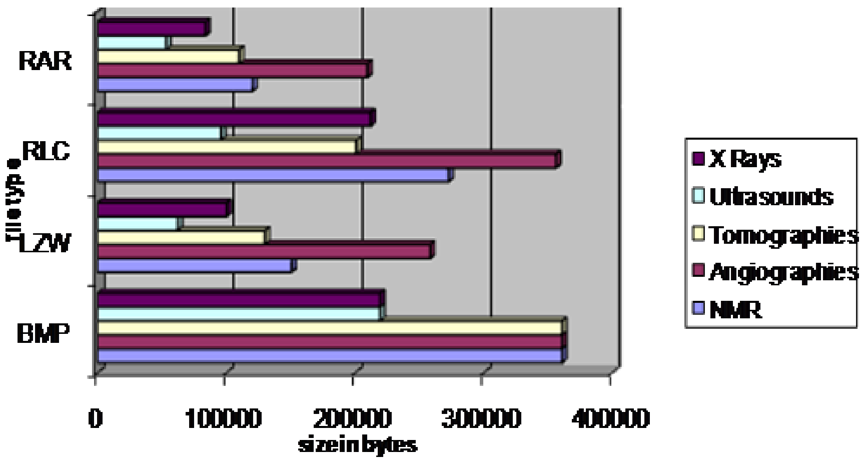

Fig. 1. The performance of coding for 255 colors

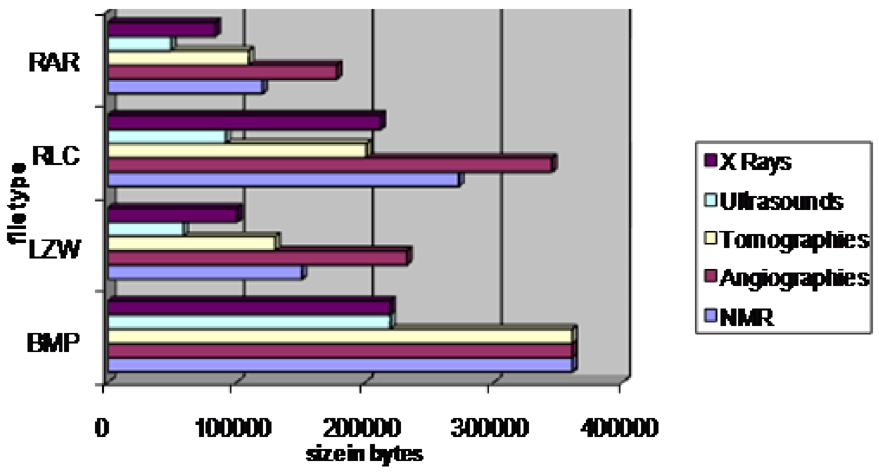

Fig. 2. The performance of coding for 128 colors

For the RLC compression, one can see that it does not depend very much on the number of colors from the image, depending more on the correlation within the image. In the case of the 16 color images, the RLE compression for the BMP format, with the combination of two pixels on the same byte, is superior to the RLC encoding for The ICompress format. Exceptions are images that, after decreasing the number of colors, have become more correlated, in advantage of the RLC encoding.

REFERENCES

[1] *G. Held*, Data Compression, J. Wiley, New York, 1984.
[2] *A.T. Murgan*, The Principles of Information Theory in Information and Communication Engineering, Romanian Academy Press, Bucharest, 1998.
[3] *R. Rădescu*, Lossless Compression – Methods and Applications, Matrix Rom Press, Bucharest, 2003.
[4] *R. Rădescu*, Digital Transmission of Information – Applications and Practical Works, UPB Press, Bucharest, 2006.
[5] *A.T. Murgan*, *R. Rădescu*, Comparison of Algorithms for Lossless Data Compression Using the Lempel-Ziv-Welch Type Methods, Proceedings of 1994 IEEE-IMS Workshop on Information Theory and Statistics, pp. 105, Alexandria, Virginia, USA, October 1994.
[6] *R. Rădescu*, *C. Şindelaru*, An Application in Image Compression Using the RLC and LZW Algorithms, EEA Revue on Electronics, Electro-technique and Automatics, Electra Press, Bucharest, **Vol. 51**, No. 4, pp. 39-42, 2003.
[7] *R. Rădescu*, *Şt. Olteanu*, Text and Image Compression with Derived LZW Algorithms, EEA Revue on Electronics, Electro-technique and Automatics, Electra Press, Bucharest, **Vol. 53**, Nr. 4, pp. 7-10, Oct.-Dec. 2005.