A New Lossless Compression Algorithm for Static Color Images - INA

Juan Ignacio and Larrauri Villamor
Universidad de Deusto
Bilbao, Spain

Abstract

We present a new algorithm for compression of static color images. It allows to get higher compression ratios than the universal methods. Most of these traditional Lossless methods use techniques based on the elimination or reduction of the existent redundancy in the data (pixels), using mainly methods based on statistical models (e.g. Huffman Coding, Arithmetic Coding), dictionary models, pattern substitution (LZW...) or predictive coding of the adjacent pixel or near symbols (FELICS, JPEG...). This way, they lead to ratios which oscillate from 1:2 to 1:4 being considered these last ones as well acceptable results.

Alternatively, we propose a new universal method based mainly on three sequential processes: first, segmentation of the image in fixed blocks, second, application of a compression algorithm based on the structure of data in form of binary tree and last, coding in order traversal of the binary tree. This method guarantees as minimum a ratio of compression of 1:3.

Finally, we have applied the proposed method to a group of images of different sources and nature (photographic, satellite, medical etc.) and we have compared the experimental results with those given by the universal methods, among which is included JPEG Lossless proposed as an standard.

Introduction

The compression ratios acquired when compressing Lossless images depend directly on the inside structure of the data, that is, the relation among the pixels and their adjacent pixels. Because of this, a universal algorithm which guarantees the same results with different kinds of images (scientific, scanned images, photographs,...) does not exist.

One method can achieve high ratios with photographs and little efficient results with a different type of image and, even a contradiction may happen: an output compressed file of greater size than the input one can be got.

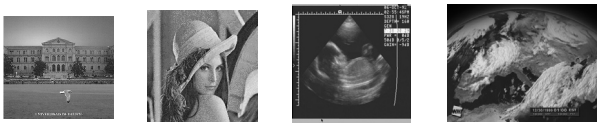

Fig.1 RLE-10% Fig2 RLE:33% Fig.3 RLE:50% Fig. 4RLE>Image

Frequently, the inside structure is defined by the nature of the source image. In this case, we can select the algorithm which best fits to this source. Nevertheless, in practice, two images coming from the same origin source, can have different inside structure.

The theoretical solution lies in examining the image a priori getting, through a histogram or probabilistic calculation, the characteristics of the image and determining afterwards the optimal model to use. However, this procedure is not viable in practice, because it means a former reading of the image which increases the time and CPU resources.

We present a new approach based on ensuring a compression ratio of 1:3 or 66% reduction in size of the source of the image without loss of data, no matter the provenance of the original image. From this ratio, the greater the redundance in the image is, the greater the achieved compression ratio will be.

Description of the Method

The compression method consists of three sequential processes which are represented in the following figure described in detail below:

I. Source Segmentation Binary tree Encoding I.Compressed

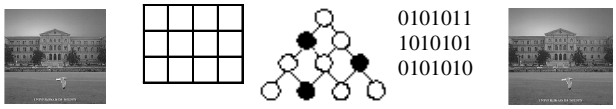

Figure 5. The sequential steps of the method.

The segmentation process is the key for the success of the pixel treatment in form of binary tree structure. The algorithm is the core of the method and its main novelty. It consists in representing the pixels of each block by means of the structure of complete binary tree or pyramid. This structure is built from couples of pixels which are the nodes, which are ordered according to what we call the "distance". The process of traversal coding allows to reduce the codeword or path (number of bits used to represent a node from the root to the node itself) through of a system of lineal code. Next we describe each of the processes.

Image Segmentation

Initially, the complete image is divided in blocks of a fixed size which are not significant. Each block is composed of 4x4 pixels, no matter the characteristics of the image The blocks represent the sequential order of execution of the image until its finalisation. The sequential order is of up-below and of left to right. If the image is RGB or true colour or more than 24 bits per pixel it requires to define the colour, each colour plane will be processed as an independent plane until the three planes that define the image are completed.

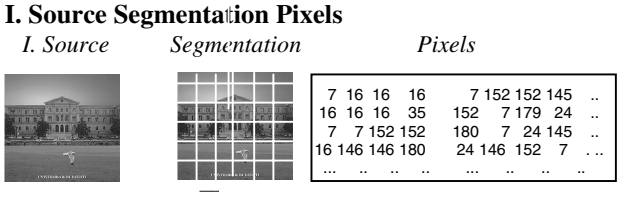

Figure 6. Segmentation process

The image segmentation does not generate data compression. However, it gives the following benefits: it increases the execution speed when operating with blocks of 4 x 4 pixels, it reduces the storage space or memory, speed up the data visualization when compressing and decompressing the image, and it adapts the data to a binary tree data structure to be treated in the next process.

Data Structure: Binary Tree

Some universal methods as Shanon-Fano o Huffman Coding use the binary tree structure to code the data basing it on the probability of symbol ocurrance. These methods use much CPU resources in building up and coding the binary tree.

We present a new approach in which the pixels are deal with as ordered couples. Each ordered couple generates a node in the binary tree structure. However, we use an intelligent coding system that lets us know which pixels form the node without having to travel along the binary tree.

The block is the unit for data treatment. It provides the data adaptation for its treatment as a binary tree structure. The pixels being different (p_j) in each block (b_i) are placed in growing order determining the base of the complete binary tree

$$Bi = \{ p_1, p_2, p_3, p_4, \ldots p_{16} \}$$

Each couple genarates a node into the binary tree. This is, each binary tree will contain only eight nodes. The eight resultant nodes do not represent any relationship among pixels, but only a value or "distance " which allows us to simplify the treatment of binary tree structure reducing the level or height of the tree to a maximum of 9 levels.

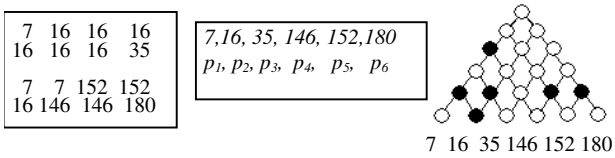

Figure 7. Generation of the binary tree.

The entropy is the number of necessary bits to code the block of 4x4 pixels, H(i)
$P(i)$ = probability of appearance of the pixel.
$n(i)$ = number of bits of code node.

$$H(i) = \sum_{i=1}^{16} P(i)n(i) \quad (1)$$

We propose a new algorithm based on the treatment of the blocks as a data structure in the form of a compact tree by reduction of the number of levels or height of the binary tree. Given a block of 16 elements (matrix of 4 x 4 pixels), we represent the block as a data structure in form of a binary tree, which satisfies the property of being compact, because although we have 16 elements or pixels different to treating him as relationship of orderly couples as maximum we will have 8 nodes which are formed by each couple. Therefore, we establish that the binary tree as maximum will have nine levels, and then the total number of necessary bits to code the block, T(Bi):

$n(i)$ = number of bits of code node.

$$T(Bi) = \sum_{i=1}^{n<=9} n(i) \quad (2)$$

Formal Description of Data Compression Algorithm

To encode an image, we treatment the data blocks and repeat the following step by each block. (B_1, B_2, B_3, ... , B_n):

1. Initially we generate a sequence of positive integer numbers corresponding to the values of the pixels of the block Bi .

$$Bi = \{ p_1, p_2, p_3, p_4, \ldots p_{16} \}$$

where

$$p_1 < p_2 < p_3 < \ldots < p_n$$

2. The pixels group in orderly couples as it is described in the following example, in order at their distances in absolute value.

$$\begin{bmatrix} \binom{p_1}{p_5} & \binom{p_2}{p_6} & \binom{p_3}{p_7} & \binom{p_4}{p_8} \\ \binom{p_9}{p_{13}} & \binom{p_{10}}{p_{14}} & \binom{p_{11}}{p_{15}} & \binom{p_{12}}{p_{16}} \end{bmatrix} \begin{bmatrix} \binom{7}{16} & \binom{16}{16} & \binom{16}{16} & \binom{16}{35} \\ \binom{7}{16} & \binom{7}{146} & \binom{152}{146} & \binom{152}{180} \end{bmatrix} \begin{bmatrix} 1001 \\ \\ 1311 \end{bmatrix}$$

Figure 8. Distance

3. Built the binary tree. The distances represent the level or height that it occupies the node in the binary tree.

Therefore, the "distance= 0" correspond at the level base or more low of the tree. The "distance= 1" correspond at the first level, and so forth. To build the structure of the data in form of binary tree we begin with the inferior levels.

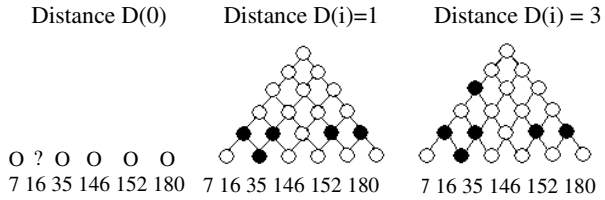

Figure 9. Generation of the binary tree.

4. Each child node generated believes two parents nodes, except if the child node is located in the left or right branch of the tree. In this case, it generates an only father node.

Properties of the Binary Tree:

a) The binary tree is complete, that is to say each child node belongs to a father node.
b) All the pixels of the block are contained in at least in a node, that is to say, all the pixels are represented.
c) In the left branch and another in the right branch of the tree a node always exists as minimum. This property still allows to reduce more the bits number.
d) Possibility of compacting of the binary tree only representing the levels or existent distances.
e) Each node is represented by an unique and inferior code to 9 bits.

Theorem:

When the proposed algorithm is applied to a data structure in form of compact binary tree and which is composed by the eight resulting nodes of the orderly couples generated by each block Bi, the total number of bits (T) employees to code the binary tree,

$$T(Bi) = \sum_{i=1}^{8} p_{low} + distance + 1 \qquad (3)$$

plow = the first pixel of the node.
B(i) = Block (4x4 pixels).

Coding

The methods based in the coding of a binary tree structure need to travel along the path between a root node and the value to code in order to get the data coding or codeword.
Our method simplifies this task. Needing no pointers and without travelling along the tree, we get the same codeword as we do with a traverse coding. Codeword generation is very fast and without CPU resources consumption. The codeword structure of each ordered couple will be made of a sequence of consecutive ones and zeros. The sequence of ones corresponds to the number of order that the least of the couple pixel has in the series of integer numbers and the number of zeros is defined by the "distance" plus one

$$Codeword\begin{pmatrix} p_{low} \\ p_{high} \end{pmatrix} = low + (high - low + 1)$$

where:
low : number of order that the least of the couple pixel has in the series.
high: number of order that the least of the couple pixel has in the series.

Applying the example of the figure 4. (Block 1)

1. Sequence of positive integer numbers

$$\begin{bmatrix} \binom{7}{16} & \binom{16}{16} & \binom{16}{16} & \binom{16}{35} \\ \binom{7}{16} & \binom{7}{146} & \binom{152}{146} & \binom{152}{180} \end{bmatrix}$$

$p_1 < p_2 < p_3 < .... < p_n$
$7 < 16 < 35 < 146 < 152 < 180$
N. of order:
1 , 2, 3, 4, 5, 6

Table 1. Coding of each couple of pixels

Pixels	Order Low	Order high	Code-word "1"	Codeword "0" (highlow+1)	Coding
(7,16)	1	2	1	2-1+1 = 2 (00)	100
(16,16)	2	2	11	2-2+1=1 (0)	110
(16,16)	2	2	11	2-2+1=1 (0)	110
(16,35)	2	3	11	3-2+1=2 (00)	1100
(7,16)	1	2	1	2-1+1=2 (00)	100
(7,146)	1	4	1	4-1+1=4 (0000)	10000
(152,146)	4	5	1111	5-4+1=2 (00)	111100
(152,180)	5	6	11111	6-5+1=2 (00)	1111100
Code(100-110-110-1100-100-10000-111100-1111100)=34 bits					

Our coding system rapidly gets the codewords with no need to travel along the binary tree. In fact, we prove that the resulting coding is the same as that got when applying a traverse coding. In the latest, the root node corresponds to the pixel of least order $p_1$which is coded with symbol "10".

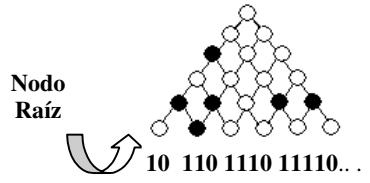

Figure 10. Traversal coding

The results are the same as those we get when applying a traverse coding . Each node corresponding to the right branch is coded adding a "1" for each crossed node and a final zero (Level 0). All the ascending nodes corresponding to the left branch are coded adding a "0" for each crossed node or level. Therefore, the code of each node is unique.

The complete coding of the block is a sequence of the codewords generated by each of the pixels couples. Each codeword is made of a series of consecutive ones and zeros, which lets us reduce more the codeword. In each codeword "10" is substituted by "0". Afterwards, the binary tree is compressed. In the previous example, level three results inactive and we reduce the nodes in the upper level in one bit. Therefore, the block consisting in six different pixels needs 3 bits per pixel, that is, 48 bits. The coding method allows us to downsize to 31 bits (1.9 bits/pixel). Of course, the compression level will depend on the combination of the couples of pixels in each image. The reached results show that in the top limit (16 different pixels) the required number of bits to code the block is 42 bits (2.6 bits/pixel).

Table 2. Image Compression ratios (True Color)

METHOD	Satellite	Medical	Scanned	Lena	Baboon
Source size (Bytes)	400*300= 360.000	512*512= 786.432	400*500 =600.000	512*512= 786.432	100*100 =30.000
Substitution RLE-PackBits	268.024 Ratio: 26%	432.102 Ratio: 40%	472.727 Ratio: 21%	720.132 Ratio: 8%	28.388 Ratio: 1%
Statistical Huffman Coding	351232 Ratio: 4%	646.144 Ratio: 18%	555.008 Ratio: 8%	765.952 Ratio: 3%	29696 Ratio: 1%
Dictionary LZW, LZ77, LZ..	266.240 Ratio: 27%	432.128 Ratio: 46%	473.088 Ratio: 22%	719.872 Ratio: 9%	28.672 Ratio: 6%
JPEG-LS Standard	165.588 Ratio: 54%	239.400 Ratio: 70%	397.312 Ratio: 34%	446.464 Ratio: 43%	22.528 Ratio: 25%
INA Algorithm	156.242 Ratio: 67%	212.524 Ratio: 73%	366.368 Ratio: 39%	260.244 Ratio: 33%	19.936 Ratio: 34%

Experimental Results

We have analyzed and compared the proposed algorithm using the rules described above with the standard methods and the results we get are shown in table 2. We have applied these new methods to a number of static images and their compression ratios are greater than those of the standard methods.

References

1. Rabbani, Majid. Jones, Paul W. "Digital Image Compression Techniques". SPIE- International Society for Optical Engineering. 1991
2. H. Yohoo. "A Losssless Coding Algorithm for the Compression of Numerical Data". IEEE Proc. Data Compression Conference. 1993
3. P. G. Howard and J.S. Vitter. "Fast and Efficient Lossless Image Compression". New York: Van Nostrand Reinhold. 1993.
4. Pennebaker, William B. And Joan L. Mitchell. "JPEG Still Image Data Compression Standard". New York: Van Nostrand Reinhold. 1993
5. Held Gilbert and Thomas R. Marshall "Data and Image Compression: Tools and Techniques", 4th edition. Chichester. England, 1996.
6. Mark Nelson "The Data Compression Book ", 2nd edition. M&T EdiBooks, New York,1995.
7. David Salomon. "Data Compression: The Complete reference", 2nd edition, Springer, 2000.

Biography

Juan Ignacio Larrauri received his B.S. degree in Computer Science from the Universidad de Deusto at Bilbao in 1985. Since 1987 until 1999, he has worked in the Instituto de Formación y Estudios Sociales – IFES. Since 1999 he is professor in the Universidad de Deusto and actually he is developing the Doctoral Thesis about the image compression He is the responsible for the Industrial Perception Systems Laboratory in the Department of Automatics and Electronics of the Faculty of Engineering.