Digital Compression Technique - A Novel Method of Implementation

V.J. Rehna
Research Scholar (Noorul Islam Univ., TN) & Asst. Prof., ECE Dept., HKBKCE, B'lore, K'taka

Kehkeshan Jalall S
Sr. Lecturer, ECE Dept., HKBKCE, Bangalore, Karnataka

T.C.Manjunath
Principal
HKBK College of Engg., Bangalore, Karnataka, India

A A Powly Thomas
Prof. & Head, EEE Dept., HKBK College of Engg., Bangalore, Karnataka, India

ABSTRACT
A method of compressing the binary images, i.e., in the form of 0's & 1's is presented in this research paper. The starting pixel value is taken and the lengths of the subsequent runs are taken into consideration while designing the compression code. The main advantage of this code is the saving of the memory space, which will lead to the faster transmission rate over the communication channels. The simulation results show the effectiveness of the developed method.

General Terms
Run length code, Compression.

Keywords
Memory space, Binary image, Gray scale image.

1. INTRODUCTION
Compressing an image is significantly different than compressing raw binary data. Of course, general-purpose compression programs can be used to compress images, but the result is less than optimal. This is because images have certain statistical properties, which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space. This also means that lossy compression techniques can be used in this area. Lossless compression involves with compressing data which, when decompressed, will be an exact replica of the original data. This is the case when binary data such as executables, documents etc. are compressed. They need to be exactly reproduced when decompressed [16].

On the other hand, images (and music too) need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable. Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the internet or downloaded from web pages. Image compression is the application of data compression on digital images. In effect, the objective is to reduce redundancy of the image data in order to be able to store or transmit data in an efficient form. Image compression can be lossy or lossless [17].

Lossless compression is sometimes preferred for artificial images such as technical drawings, icons or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossless compression methods may also be preferred for high value content, such as medical imaging or image scans made for archival purposes. Lossy methods are especially suitable for natural images such as photos in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate [18].

In this paper, we discuss about a method of run length encoding for compression of the binary images. The paper is organized as follows. A brief introduction about the image compression technique was presented in the previous paragraphs. In section 2, a brief introduction about the storing of images in computers is dealt with. The comparison of high-resolution images along with that of image compression is described in section 3. The next section, i.e., section 4 deals with the types of image compression techniques. Section 5 describes the run length-encoding scheme developed to compress the binary images along with some simulation examples. Section 6 gives the simulation results. The user developed code in C / C++ used for simulation is presented in section 7. Conclusions are presented in section 8 followed by the references.

2. STORING IMAGES IN COMPUTERS
Camera converts a 3D physical object into the image of the object. The output of the camera is 2D analog image which is represented by i (x, y). Computer cannot process this analog image. They have to be digitized. The analog image i (x, y) is converted into a digital image (DI) / gray scale image I (k, j) using the A to D conversion. The digital / gray scale image is threshold to obtain a BI - binary image B (k, j) which consists of foreground objects represented by 1's and background objects represented by 0's. Each 1 or 0 is a pixel or an element of a binary image L (k, j) and can be stored either in 1 bit or in 1 byte. This is how an image is stored in the memory of the computer [2].

Note that in the memory of the computer or in the CD's or in the DVD's or in the floppy disks or in the hard disks, the images are always stored in the form of 0's and 1's. Storage capacity of the images goes on increasing with the size, brightness, contrast, resolution, colors, whether it is 2D or a 3D image (hologram), etc. The different methods of storing the pixel values of the binary image are: 1 pixel can be stored in 1 bit or 1 pixel stored in 1 byte [2].

3. COMPARISON BETWEEN HR IMAGES AND COMPRESSED IMAGES
High resolution images consume more amount of memory space for storage, take long time to transmit over the communication channels, size of memory consumed for storing is more, & has got good clarity. Hence, images are often compressed / packed or coded. Images are often compressed in order to reduce the storage capacity. The

advantages of image compression being, the memory storage gets reduced by a factor of 8, takes less time to transmit over the communication channels. But the main disadvantage is the retrieval of images takes more time. For example, when we are downloading the images from the Internet, we can see that it takes more time to open because the image is getting decoded [2].

4. TYPES OF IC TECHNIQUES

Image compression aims to reduce the number of bits required to represent an image by removing the redundancies which makes it one of the most useful and commercially successful technologies in the field of digital image processing.

Three principle types of data redundancies that can be identified are:

A. *Coding redundancy:* Coding redundancy consists of variable length code words selected as to match the statistics of the original source. In the case of digital image processing, it is the image itself or the processed version of its pixel values. Examples of image coding schemes that explore coding redundancy are the Huffman coding and the Arithmetic coding technique.

B. *Spatial redundancy:* Spatial reduncancy is sometimes called interframe redundancy, Geometric redundancy or Interpixel redundancy. Here, because the pixels of most 2-D intensity arrays are correlated spatially that is each pixel is similar to or independent on neighbouring pixels, information is unnecessarily replicated in the representation of the correlated pixels. Examples of this type of redundancy include Constant area coding and many Predictive coding algorithms.

C. *Irrelevant information:* Most 2-D intensity arrays contain information that is ignored by the human visual system. Image and video compression techniques aim at eliminating or reducing any amount of data that is psycho visually redundant. Most of the image coding algorithms in use today exploit this type of redundancy, such as the discrete cosine transform based algorithm at the heart of the JPEG encoding standard.

There are various methods of data compression techniques such as the Hauffman's coding, Lempel-Zev method of coding, zip / unzip method, WinZip, WinRar, tar and the Run Length Encoding [RLE]. The different methods of storing the pixel values of the binary image are 1 pixel can be stored in 1 bit or in 1 byte. If 1 pixel is stored in 1 bit, then 8 pixels can be stored in 1 byte, so that the factor of compression is 8. There are different methods for lossless image compression such as

- Run-length encoding - used as default method in PCX and as one of possible in BMP, TGA, TIFF
- Entropy coding
- Adaptive dictionary algorithms such as LZW - used in GIF and TIFF
- Deflation - used in PNG, MNG and TIFF [2]

5. RUN LENGTH ENCODING (RLE)

In this section, we deal with the RLE, which is a method of compressing the binary images and uses an encoding scheme in which a RUN is represented by a sequence of pixels having the same value and the length of the run represents the total number of pixels in the sequence. Here, we store the starting value of the pixel and the lengths of the subsequent runs as the binary image is scanned from the left to the right and from the top to the bottom. Hence, we get a coded image or a compressed image. This run length encoding is a simplest dictionary based data compression technique. Image files frequently contain the same character repeated many times in a row. Images, particularly those having very few gray levels, often contain regions of adjacent pixels, all with the same gray levels. Each row of such images can have long runs of the same gray value. In, such cases, one can store a code specifying the value of the gray level, followed by the length of the run, rather than storing the same value many a times over. As is evident, the run length encoding achieves considerable compaction in images, which have a fairly constant background. The RLE also eliminates the inter-pixel redundancies [2].

		Column → n = 10									
		1	2	3	4	5	6	7	8	9	10
R	1	0	0	0	0	0	0	0	0	0	0
O	2	0	0	0	0	0	0	0	0	0	0
W	3	0	1	1	1	0	0	0	0	0	0
	4	0	0	0	1	1	1	0	0	0	0
	5	0	0	0	0	0	1	1	1	0	0
	6	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0	0	0
m =	8	0	0	0	0	0	0	0	0	0	0

Fig. 1: A (8 × 10) Binary Image

To consider a simulation example, let us have a binary image B (k , j) of size (8 × 10) of 8 rows and 10 columns as shown in Fig. 1, the foreground represented by 1's and the background represented by 0's. Scan the binary image from left to right and from top to bottom using raster-scanning technique. Store the first pixel value and the length of the subsequent runs as the binary image is scanned from left to right and from the top to bottom using the raster scanning method [2].

The sequence of storage of the pixel values is shown below in the form of an algorithm as follows.

1st pixel value is 0.
Store 0 in byte 1
The length of Run 1 is $L_{R1} = 0$

There are 21 pixels with value 0
Store 21 in byte 2
The length of Run 2 is $L_{R2} = 21$

There are 3 pixels with value 1
Store 3 in byte 3
The length of Run 3 is $L_{R3} = 3$

There are 9 pixels with value 0
Store 9 in byte 4
The length of Run 4 is $L_{R3} = 9$

There are 3 pixels with value 1
Store 3 in byte 5
The length of Run 5 is $L_{R5} = 3$

There are 9 pixels with value 0
Store 9 in byte 6
The length of Run 6 is $L_{R6} = 9$

There are 3 pixels with value 1
Store 3 in byte 7
The length of Run 7 is $L_{R7} = 3$

There are 32 pixels with value 0
Store 32 in byte 8
The length of Run 8 is $L_{R8} = 32$

Therefore, the sequence of run lengths is given by

$\Gamma = [0\ 21\ 3\ 9\ 3\ 9\ 3\ 32]^T$.

i.e., the memory storage = 8 bytes.

If 1 pixel / byte is used to store the given binary image, then memory storage = $(8 \times 10) = 80$ bytes.

If 1 pixel / bit is used to store the given binary image, then memory storage = $\frac{80}{8}$ = 10 bytes.

If Run Length Encoding (R L E) is used, then memory storage = 8 bytes. ∴, % saving in memory space if RLE is used = % $S = \frac{80-8}{8} \times 100 = 90$ %.

RLE method of compressing the binary images is not always successful. It depends on the number of 1's and 0's. For ex., when the image has got alternate 1's and 0's, then the RLE fails. It is as good as storing the pixel in 1 byte. For a chess board, the memory storage if RLE is used or 1 pixel / bit is used, then the memory consumed = 64 bytes as shown in Fig. 2. The shortest run length code is an image having all 0's (2 bytes) or an image having all 1's (2 bytes) as shown in Fig. 3. Another disadvantage with the run length code is from the run length code (first value), we cannot know which is the number of zeros or which is number of ones [2].

1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1

Fig. 2 : A Chess Board

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$\Gamma = \{0, 16\}^T$

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

$\Gamma = \{0, 16\}^T$

Fig. 3 : Shortest RL Code

Let us consider another simulation example as shown in the Fig. 4.

The run length code of the given binary image is

$\Gamma = \{ 0, 17, 2, 1, 1, 5, 4, 4, 3, 27 \}^T$.

The storage of Γ requires 10 bytes.

At one byte per number, the number of bytes of storage required to store the (8×8) image is 64.

∴, the percentage saving in memory [2]

$= \frac{100(64-10)}{64} = 84.4$ %.

		Col j →							
		1	2	3	4	5	6	7	8
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
Row	3	0	1	1	0	1	0	0	0
k	4	0	0	1	1	1	1	0	0
↓	5	0	0	1	1	1	0	0	0
	6	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0

Fig. 4 : A (8×8) binary image

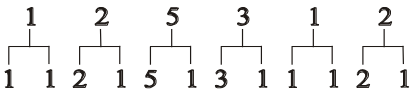

Fig. 5: A Image Data

Now, consider performing the RLE on the data shown in the Fig. 5. In RLE, the first value specifies the gray value while the second value specifies the run. Therefore, the RLE code is $\Gamma = [1\ 1\ 2\ 1\ 5\ 1\ 3\ 1\ 1\ 1\ 2\ 1]^T$. The RLE code here is double than that of the original sequence. Hence, RLE should only be used if we have the same character gray value repeated many a times in a row.

6. SIMULATION RESULTS

A graphical user interface program was developed in C / C++ language and many codes was compiled and run. On running the code, the following screens as shown below appeared, i.e., the inputs & the output screens, which are nothing but the simulation results [2].

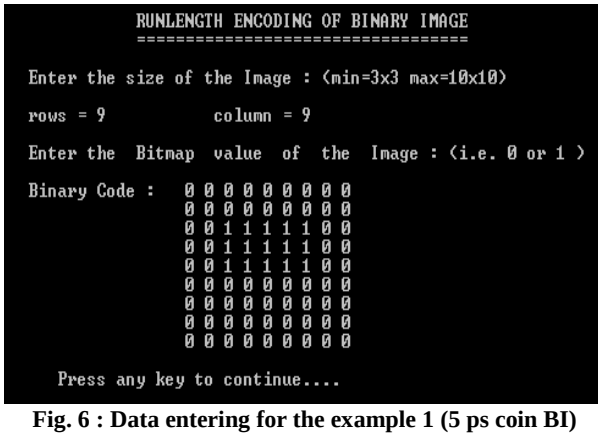

Fig. 6 : Data entering for the example 1 (5 ps coin BI)

```
        RUNLENGTH ENCODING OF BINARY IMAGE
        ==================================
Runlength Code:
Tau = <0, 20, 5, 4, 5, 4, 5, 38>
Storage of packed binary image   : 10 bytes
Storage of unpacked binary image  : 81 bytes
Storage of encoded image  : 8 bytes
Press any key to continue.....
```

Fig. 7 : Simulation result of the Fig. 6

```
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 1 0 1 0 0 0
0 1 0 0 0 1 0 0
0 0 1 0 1 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

  Input Bitmap

Run Length Code for the given bitmap :
[ 0 11 1 6 1 1 1 4 1 3 1 4 1 1 1 6 1 20  ]
```

Fig. 8 : Another simulation result of a binary image

7. THE TURBO C / C++ FLOWCHART FOR COMPILATION

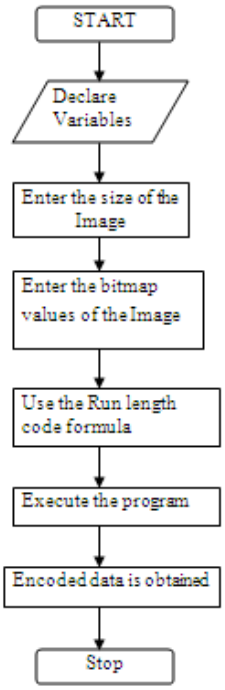

Fig. 9 : Flow chart of the developed program

To test the effectiveness of the written algorithm, it is applied on a real digital image. To start with, a (64 × 64) color image of the eye is considered as shown in the Fig. 10. Next, it is converted to its gray scale version as shown in the Fig. 11. Finally, the gray scale image is converted into a binary image & is as shown in the Fig. 12. This binary image acts as the input to the run length code algorithm. The flowchart shown above in the Fig. 9 is used to obtain the run length code, Γ of the binary image of the eye.

Fig. 10 : A Colored Image of the eye

Fig. 11 : A Gray Scale Image of the eye

Fig. 12 : A Binary Image of the eye

As the size of the obtained binary file was very big, it could not be incorporated in this paper. The binary image was given as input to the developed program & once the written C program was run, the run length code was displayed along with the results for the converted binary image.

Γ = {0, 30, 10, 21, 19, 33, ……., 52, 141} ^{T}…… storage of Γ requires 16 bytes as each run is stored in 1 byte.

If 1 pixel / byte is used to store the given binary image, then memory storage = (64× 64) = 4096 bytes = 4 KB.

If 1 pixel / bit is used to store the given binary image, then memory storage = $\frac{4096}{8}$ = 512 bytes.

If Run Length Encoding (RLE) is used, then memory storage for Γ = 16 bytes.

∴, % saving in memory space if RLE is used = % S = $\left(\frac{4096-32}{32}\right)$ × 100 = 127 %. This shows the effectiveness of the run length code algorithm.

8. CONCLUSIONS

A method of compressing the binary images was developed. A GUI in C / C++ was also developed for the same. It was demonstrated that run length encoding is one of the efficient method of compressing the binary images. Of course, it has got some drawbacks. Run-Length Encoding, or RLE thus, is a technique used to reduce the size of a repeating string of characters. This repeating string is called a *run*, typically RLE encodes a run of symbols into two bytes, a count and a symbol. RLE can thus compress any type of data regardless of

its information content, but the content of data to be compressed affects the compression ratio. RLE cannot achieve high compression ratios compared to other compression methods, but it is easy to implement and is quick to execute. But, some times, the RLE method will not be successful (chess board, an image having alternate zeros & ones) & often it depends on the type of the bit pattern of zeros & ones in the binary image. Thus, run-length encoding is supported by most bitmap file formats such as JPEG, TIFF, BMP and PCX. The algorithm is applied on a real digital image & its effectiveness is obtained, which can be seen from the simulation results.

9. REFERENCES

[1] Craig J, Introduction to Robotics : Mechanics, Dynamics & Control, Addison Wessely, USA, 1986.

[2] Robert, J. Schilling, Fundamentals of Robotics - Analysis and Control, PHI, New Delhi.

[3] Klafter, Thomas and Negin, Robotic Engineering, PHI, New Delhi.

[4] Fu, Gonzalez and Lee, Robotics: Control, Sensing, Vision and Intelligence, McGraw Hill.

[5] Groover, Weiss, Nagel and Odrey, Industrial Robotics, McGraw Hill.

[6] Ranky, P. G., C. Y. Ho, Robot Modeling, Control & Applications, IFS Publishers, Springer, UK.

[7] Crane, Joseph Duffy, Kinematic Analysis of Robotic Manipulators, Cambridge Press, UK.

[8] Manjunath, T.C., Fundamentals of Robotics, Fourth edn., Nandu Publishers, Mumbai, 2005.

[9] Manjunath, T.C., Fast Track to Robotics, Second edn., Nandu Publishers, Mumbai, 2005.

[10] Dhananjay K Teckedath, Image Processing, Third edn., Nandu Publishers, Mumbai, 2006.

[11] Gonzalvez and Woods, Digital Image Processing, Addison Wesseley Publishers.

[12] Anil K Jain, Digital Image Processing, Prentice Hall, Englewood Cliffs, New Jersey, USA.

[13] http://www.wikipedia.org

[14] Michael Dipperstein, Run Length Encoding (RLE) Discussion and Implementation.

[15] Amir Said and William A. Pearlman, "An Image Multiresolution Representation for Lossless and Lossy Image Compression, IEEE Trans. on Image Processing, vol. 5, pp. 1303-1310, Sept. 1996.

[16] Armando Manduca and Amir Said, "Wavelet compression of medical images with set partitioning in hierarchical trees," Proc. of the SPIE Symposium on Medical Imaging, Cambridge, MA, Mar. 1996.

[17] Skodras, A., Christopoulos, C., Ebrahimi, T. "The JPEG 2000 still image compression standard", IEEE Signal Processing Magazine, vol. 18, Issue 5, pp. 36 – 58, Sep. 2001.

[18] Majid Rabbani, Rajan Joshi, "An overview of the JPEG 2000 still image compression standard", Elsevier's Signal Processing: Image Communication, Vol. 17, Issue 1, pp. 3 – 48, Jan. 2002.