

An Enhanced Approach in Run Length Encoding Scheme (EARLE)

A. Nagarajan,
Assistant Professor.

Dept of Master of Computer Application
PSNA College of Engineering & Technology
Dindigul.

Dr.K.Alagarsamy,
Associate Professor

Dept of MCA, Computer Center
Madurai Kamaraj University
Madurai.

Abstract:

Image compression is reducing 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 drives or memory space. It also consumes the time required for images to be sent over the Internet or downloaded from Web pages. Data compression schemes give the optimized solution to transfer the data and store the data in secondary storage. In this paper we are going to propose new idea about enhanced version of run length encoding algorithm. We enhanced the drawbacks of run length encoding scheme to provide an optimum compression.

Keywords:

Image Compression, Run Length Encoding (RLE), EARLE.

Image Compression:

Compression is used everywhere in the network applications. All the images you get on the web are compressed, typically in the JPEG or GIF formats, most modems use compression, HDTV will be compressed using MPEG-2, and several file systems automatically compress files when stored, and the rest of us do it by hand. These algorithms used in the real world make heavy use of a wide set of algorithmic tools, including sorting, hash tables, tries, and FFTs. Furthermore,

There are two different types of digital graphics system available one is vector and another one is bitmap. JPEG compression only works on bitmap images since vector graphics are not digital images, and cannot be made any smaller. Bitmaps, on the other hand, can be compressed and the process is

called lossy compression because when the image is compressed some of the information. Bitmap images are $m \times n$ matrices where every entry in the picture matrix corresponds to a small square in the picture. Bitmaps come in four main types the first is binary, where the image is an $m \times n$ matrix with every element in the matrix being either a 0 for black, or a 1 for white, these images are very poor quality, and are almost never used for storing digital images.

The second type is intensity images which store a number between 0 and 1, in every entry of the matrix. Intensity images, like binary images, have zero as complete black, and one as absolute white. However, unlike binary images, intensity images have many different shades of gray, corresponding to numbers between zero and one. The next type is indexed images, where there are 256 different colors, which are stored in the image file as the image index. Every element in image matrix has a number between 0-255 which corresponds to a certain color, in the index. The last type is true color, or RGB, where the image is composed of three layered $m \times n$ matrices, one for each red, green, and blue. Today most pictures are True color, but there are some indexed and intensity images in use. JPEG Compression is enough to compress all three types of images.

There are so many algorithms available for image compression like:

- RLE
- JPEG file format
- Wavelet
- JPEG 2000
- SPIHT, etc.

Not only these algorithms available. Still there are varieties of algorithmic models available. Among that Run length Encoding Scheme is simple one. But it has variety of drawbacks while coming to the

implementation of compression part. In this paper we tried our level best to enhance the drawbacks of Run length Encoding Scheme (RLE). Before get into the actual work we just discuss about the RLE procedure and methods then we will go for proposed works.

Run Length Algorithm:

RLE is a very simple form of data compression in sequences in which the same data value occurs in many consecutive data elements are stored as a single data value and count, rather than as the original run. This is useful on data that contains many such runs, for example, relatively simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could potentially file size is increase.

In our example, let take a screen containing plain black text on a solid white background. There will be too much long runs of white pixels in the blank space, and many short runs of black pixels within the text. Let us take a hypothetical single scan line, with B representing a black pixel and W representing white:

WWWWWWWWWWWWBWWWWWWWWWW
WWBBBWWWWWWWWWWWWWWWWWW
WWWWWWBWWWWWWWWWWWWWWWW

If we apply the run-length encoding data compression algorithm to the above hypothetical scan line, we get the following:

12W1B12W3B24W1B14W

Interpret this as twelve W's, one B, twelve W's, three B's, etc.

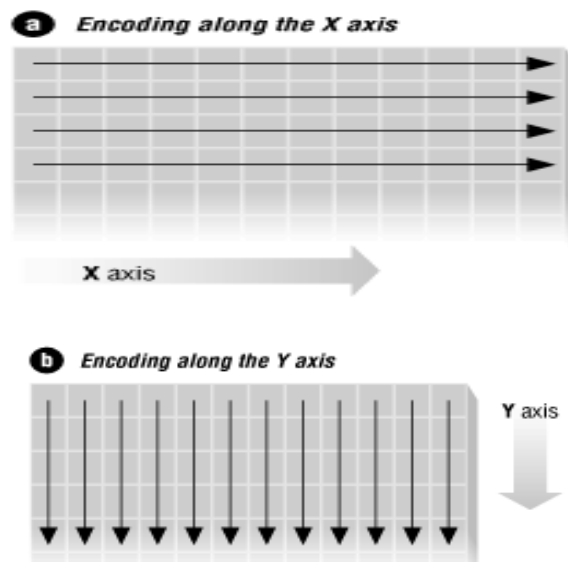
The run-length code depicts the original 67 characters in only 18. Of course, the original format used for the storage of images is generally binary rather than ASCII characters like this but the principle remains the same. We can compress binary data files with this method; file format specifications often dictate repeated bytes in files as padding space. Zero trees follow this data structure. A wavelet coefficient x is called to be insignificant with respect to a given threshold if $|x| < T$. The zero tree is based on the hypothesis that if a wavelet coefficient at a rough scale is insignificant with respect to a threshold, then all wavelet coefficients of the same orientation in the

same spatial location at the finer scale are likely to be insignificant with respect to the same threshold. More specifically, in a hierarchical sub band system, with the exception of the highest frequency sub bands, ever coefficient at a given scale can be related to a set of coefficients at the next finer scale of similar orientation. The coefficient at the rough scale is called the parent, and all coefficients corresponding to the same spatial location at the next finer scale of similar angles are called children.

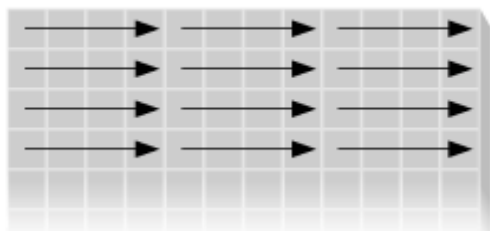
Proposed Method (EARLE):

Normally encoding algorithms are very complex to understand. There are numbers of algorithms available for encoding the schemes. Among that run length encoding algorithm is easy one to understand there are varieties of methods available to encode the long runs. The following diagram shows these methods.

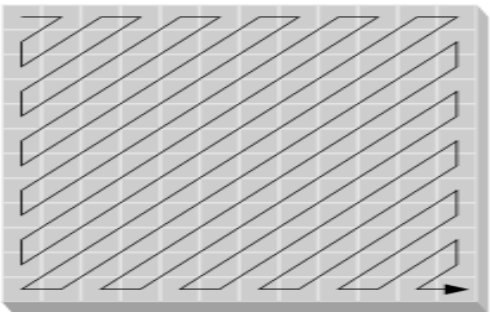
Figures:



c Encoding (4x4 pixel) tiles



d Zig-zag encoding



In our earlier example we gave text based. In this paper we are going to implement this technique into the image. Run length algorithm produce better result in text but not all type of images. Now we are trying to incorporate this technique into RGB color image. RGB color image always have three layers like Red, Green, Blue. In every layer have its own byte values. If the color of the image or color of the pixel values same with the neighbor pixel then run length will same. Now we can apply run length algorithm into the image.

Algorithm Procedure:

1. Input Image
2. Layer Separation
3. RLE encoding apply into byte level
4. Assign alpha index value in run length.
5. Link list data structure for organizing the RGB.
6. Store data structure top of image
7. Decoding.

In step one input image will take for the image compression. Even the image itself there are variety of forms available. Like plain image patterned image highly patterned image etc. if the image is plain then RLE works much efficient and good. In case the image is patterned every compression algorithm works slight dull.

Then input image will transformed into collection of pixels. In every pixel have three layers like red, green, blue. In every layer having color ratio between 0 to 255. Actually it will in binary form for our understanding we convert into integers then apply our RLE schemes.

Now We take a sample byte input of the images like

77 77 77 87 87 87 87 22 22 22 11 11 44 44 44 44 65 65

After encoding this series converted into like this

377, 487, 322, 211, 544, 265

If we store this value in direct manner inside the pixel it will take long bytes. To overcome this problem we go for data structure methods index table data dictionary and linked list. We just assign the alpha index to every runs. For example

M-377,N-487,O-322,P-211,Q-544,R-265

After assign the alpha index we have to form the linked list data structure for the pixel. In this there are four fields

1. Alpha Index
2. Data dictionary
3. Header
4. Link

Linked list stored in the top of picture so that we can retrieve the needed information or part of information as soon as and effective manner. We have compared EARLE with many compression algorithms. The result analysis show EARLE performed good compared with other algorithms. In appendix1 and appendix2 Sample Images and comparison ratio will explained refer the Appendix

column. Decompression is always the reverse process of encoding. First take the linked list index it will give the alpha index value from that we can get the run lengths. Using header field we can get the basic information of the pixel. In data dictionary we get more information about the pixels. We can also access the part of picture information

Conclusion:

We are having more no of algorithms for image compression but every algorithm has its own draw back and peculiarity depending on the image what we going to compress. From our result analysis we can say EARLE work better than other algorithmic models. It's our faith EARLE can act as one of the eminent algorithm for compression.

Future Work:

In future we can enhance this method with security aspect also. If we give any passwords and keys while compression we can get security aspect also. We can protect our image data from the hacker and third persons.

References:

[1] J. Bruce, T. Balch and M. Veloso, "Fast and Inexpensive Colour Image Segmentation for

Interactive Robots", IROS 2000, San Francisco, 2000.

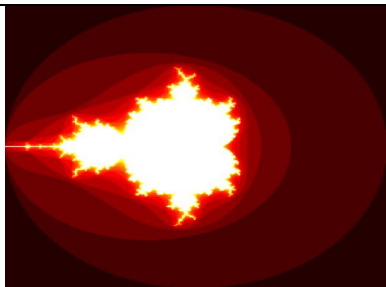
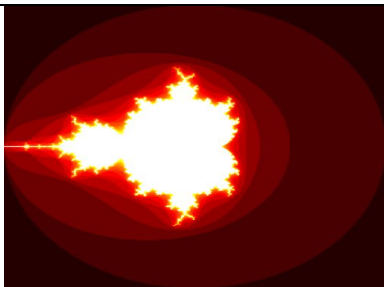
[2] Sami Khuri and Hsiu-Chin Hsu "Interactive Packages for Learning Image Compression Algorithms" lists, requires prior specific permission and/or a fee. ITiCSE 2000, Helsinki, Finland

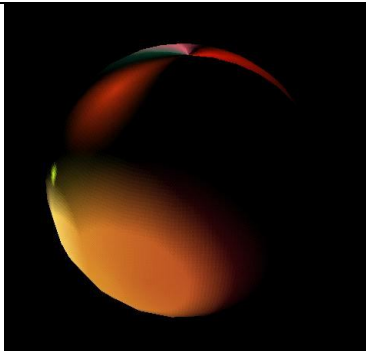
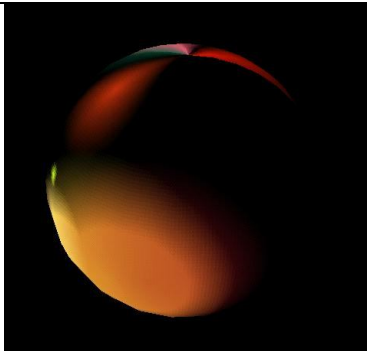




[3] S. Bhattacharjee, S., Das, D. Roy Choudhury and P. Pal Choudhuri, " A Pipelined Architecture Algorithm for Image Compression", Proc. Data Compression Conference, Saltlake City, USA, March 1997.

[4] J.G. Sen Gupta, D. Bailey, C. Messom, "A New Colour Space for Efficient and Robust Segmentation", Proceedings of IVCNZ 2004, pp 315-320, 2004. IEEE Instrumentation and Measurement Technology Conference, pp 1055-1060, ISBN 0-7803-7218-2, 2002,

[5] C. H. Messom, S. Demidenko, K. Subramaniam and G. Sen Gupta, "Size/Position Identification in Real-Time Image Processing using Run Length Encoding", IEEE Instrumentation and Measurement Technology Conference, pp 1055-1060, ISBN 0-7803-7218-2, 2002

Appendix: 1 : Sample Images:

Name	Original Image	Decoded Image from Compressed image
Design		

Globe		
leaves		
Scenery		

Appendix2: Compression ratio in %:

File Name	Size	Huffman	TIFF	GIF	EARLE
Design	1132 kb	55%	76%	79%	83%
Globe	1022kb	48%	72%	77%	86%
Leaves	987kb	51%	68%	80%	81%
Scenery	555 kb	59%	77%	82%	84%