# Image Compression using Approximate Matching and Run Length

Samir Kumar Bandyopadhyay, Tuhin Utsab Paul, Avishek Raychoudhury
Department of Computer Science and Engineering,
University of Calcutta
Kolkata-700009, India

***Abstract*— Image compression is currently a prominent topic for both military and commercial researchers. Due to rapid growth of digital media and the subsequent need for reduced storage and to transmit the image in an effective manner Image compression is needed. Image compression attempts to reduce the number of bits required to digitally represent an image while maintaining its perceived visual quality. This study concentrates on the lossless compression of image using approximate matching technique and run length encoding. The performance of this method is compared with the available jpeg compression technique over a wide number of images, showing good agreements.**

***Keywords- lossless image compression; approximate matching; run length.***

## I. INTRODUCTION

Images may be worth a thousand words, but they generally occupy much more space in a hard disk, or bandwidth in a transmission system, than their proverbial counterpart. So, in the broad field of signal processing, a very high-activity area is the research for efficient signal representations. Efficiency, in this context, generally means to have a representation from which we can recover some approximation of the original signal, but which doesn't occupy a lot of space. Unfortunately, these are contradictory requirements; in order to have better pictures, we usually need more bits.

The signals which we want to store or transmit are normally physical things like sounds or images, which are really continuous functions of time or space. Of course, in order to use digital computers to work on them, we must digitize those signals. This is normally accomplished by sampling (measuring its instantaneous value from time to time) and finely quantizing the signal (assigning a discrete value to the measurement) [1]. This procedure will produce long series of numbers. For all purposes of this article, from here on we will proceed as if these sequences were the original signals which need to be stored or transmitted, and the ones we will eventually want to recover. After all, we can consider that from this digitized representation we can recover the true (physical) signal, as long as human eyes or ears are concerned. This is what happens, for example, when we play an audio CD. In our case, we will focus mainly on image representations, so the corresponding example would be the display of a picture in a computer monitor. However, the discussion in this paper, and especially the theory developed here, apply equally well to a more general class of signals.

There are many applications requiring image compression, such as multimedia, internet, satellite imaging, remote sensing, and preservation of art work, etc. Decades of research in this area has produced a number of image compression algorithms. Most of the effort expended over the past decades on image compression has been directed towards the application and analysis of different coding techniques to compress the image data. Here in this paper also, we have proposed a two step encoding technique that transform the image data to a stream of integer values. The number of values generated by this encoding technique is much less than the original image data. The main philosophy of this encoding technique is based on the intrinsic property of most images, that similar patterns are present in close locality of images.

The coding technique makes use of this philosophy and uses an approximate matching technique along with the concept of run length to encode the image data into a stream of integer data. Experimental results over a large number of images have shown good amount of compression of image size.

## II. RELATED WORKS

Image compression may be lossy or lossless. Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossy methods are especially suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces imperceptible differences may be called visually lossless.

Methods for lossless image compression are:

- Run-length encoding – used as default method in PCX and as one of possible in BMP, TGA, TIFF
- DPCM and Predictive Coding
- Entropy encoding
- Adaptive dictionary algorithms such as LZW – used in GIF and TIFF
- Deflation – used in PNG, MNG, and TIFF
- Chain codes

Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, 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 most useful on data that contains many such runs: for example, 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 greatly increase the file size.

DPCM or differential pulse-code modulation is a signal encoder that uses the baseline of PCM but adds some functionalities based on the prediction of the samples of the signal. The input can be an analog signal or a digital signal.

Entropy encoding is a lossless data compression scheme that is independent of the specific characteristics of the medium.

One of the main types of entropy coding creates and assigns a unique prefix-free code to each unique symbol that occurs in the input. These entropy encoders then compress data by replacing each fixed-length input symbol by the corresponding variable-length prefix-free output codeword. The length of each codeword is approximately proportional to the negative logarithm of the probability. Therefore, the most common symbols use the shortest codes.

Lempel–Ziv–Welch (LZW) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch. It was published by Welch in 1984 as an improved implementation of the LZ78 algorithm published by Lempel and Ziv in 1978. The algorithm is simple to implement, and has the potential for very high throughput in hardware implementations.

Deflate is a lossless data compression algorithm that uses a combination of the LZ77 algorithm and Huffman coding. It was originally defined by Phil Katz for version 2 of his PKZIP archiving tool, and was later specified in RFC 1951.

A chain code is a lossless compression algorithm for monochrome images. The basic principle of chain codes is to separately encode each connected component, or "blot", in the image. For each such region, a point on the boundary is selected and its coordinates are transmitted. The encoder then moves along the boundary of the image and, at each step, transmits a symbol representing the direction of this movement. This continues until the encoder returns to the starting position, at which point the blot has been completely described, and encoding continues with the next blot in the image.

## III. Our Work

The main philosophy behind selecting approximate matching technique along with run length encoding technique is based on the intrinsic property of most images, that they have similar patterns in a localized area of image, more specifically the adjacent pixels row differ in very less number of pixels. This property of image is exploited to design a very effective image compression technique. Testing on a wide variety of images has provided satisfactory results. The technique used in this compression methodology is described in this section.

We consider approximate matching algorithm and run length for our image compression. The approximate matching algorithm does a comparison between two strings of equal length and represent the second string with respect to the first only with the information of the literal position where the string mismatches.

Replace. This operation is expressed as (p; char) which means replacing the character at position p by character char.

Let C denote “copy", and R denote “replace” then the following are two ways to convert the string “11010001011101010" to “11010001111001010" (0,1 are stored in ASCII)via different edit operation sequences:

C C C C C C C C R C C R C C C C C
1 1 0 1 0 0 0 1 0 1 1 1 0 1 0 1 0
1 1 0 1 0 0 0 1 1 1 1 0 0 1 0 1 0

A list of edit operations that transform a string u to another string v is called an EditTranscription of the two strings [9]. This will be represented by an edit operation sequence (u; v) that orderly lists the edit operations. For example, the edit operation sequence of the edit transcription in the above example is (\11010001011101010",\11010001111001010") = ( 9; 1),(12,0);

Approximate matching method. In this case, the string \11010001011101010" can be encoded as f(17; 2)= ( 9; 1),(12,0), storing the ASCII characters require 136 bit or 17 byte where as storing 4 characters will require 4 byte. Thus a compression of approximate 76.4% is achieved. This technique is very useful in image compression because of the inherent property of an image because two consecutive rows of an image has almost same string of pixel values. Only a few pixel varies. Experimental results prove this hypothesis.

Apart from the concept of approximate matching method, the concept of run length is also used because using run length a row of image can be represented using much less literals than the original.

Run-length Encoding, or RLE 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 compress any type of data regardless of its information content, but the content of data to be compressed affects the compression ratio. Consider a character run of 15 'A' characters which normally would require 15 bytes to store :

AAAAAAAAAAAAAAA is stored as 15A

With RLE, this would only require two bytes to store, the count (15) is stored as the first byte and the symbol (A) as the second byte.

In this compression technique, we have used the approximate matching method in unison with run length. Starting from the left uppermost row of image, every three rows are considered at a time. Of these, the middle row is represented using run length, and the row above and below it are matched with the middle row using approximate matching method. This method is continued iteratively until the whole image is scanned and compressed.

The algorithms designed as per our technique are as follows:

*A. COMPRESS (Source Raw Image file)*

This is the main algorithm for compression. This algorithm will be used to compress the data part of the Source Image File.

Output: It will output the Compressed-Image file.

Input: This function will take Source Image file as input.

1. Read the Source Image file as input. Obtain it's size (say r*c).Store the data part(pixel values) of the image in an array A of the same size.
2. Quantize the color palate of the image, i.e array A with quantization factor 17.
3. If r is not divisible by 3, then duplicate the last row 1 or 2 times at the bottom of A such that the number of rows become divisible by 3. Reset r with the corresponding new size of A.
4. Take a blank array say 'Compress' of size n*2(n is a positive integer).Starting with the 1st row, choose consecutive 3 rows at a time and perform the following operations(say, we have chosen row number k-1,k and k+1) in each iterations:
   a. For each column in the array A, if any mismatch is found in the row k-1 and k, the corresponding column number and the value at that corresponding column with row number k-1 in array A, is stored in array Compress. For every mismatch, those two values are stored in a single row of array Compress.
   b. For row number k, the corresponding value (starting from the 1st column) and its runlength (Number of consecutive pixels with same pixel value) for kth row is stored in array Compress. Every set of value and it's runlength is stored in a single row of array Compress.
   c. For each column in the array A, if any mismatch is found in the row k and k+1, the corresponding column number and the value at that corresponding column with row number k+1 is stored in array Compress. For every mismatch, those two values are stored in a single row of array Compress.
5. Repeat Step 4 until all the rows are compressed. A marker should be used to distinguish between the encrypted versions of each row. Store also the value of 'r' and 'c' in Compress.
6. Array Compress now constitutes the compressed data part of the corresponding Source Image File.

*B. DECOMPRESS (Compressed-Image file)*

This is the main algorithm for decompression or decoding the image. This algorithm will be used to decompress the data part of the Source Image File i.e. the image from the 'Compress' array.

Output: It will output the Decompressed or Decoded Image file.

Input: This function will take the Compressed-Image file ('Compress' array) as input.

1. Read the Compress array. Obtain the size of the image (say r*c) from the array. Take a blank array say 'Rec' of the same size for reconstruction of the data part of the image.
2. Starting from the 1st row, consider the compressed values of consecutive 3 rows from Temp and perform the following operations(say, we have chosen row number k-1,k and k+1) in each iterations:
   a. Firstly, construct the kth row of Rec array with the corresponding positional value and runlength value in the Compress array, by putting the same positional value in runlength number of consecutive places in the same row.
   b. Then, construct the (k-1)th row. In the corresponding Compress array for this particular row for each column if an entry for column number 'v' is not present, then Rec[(k-1),v]=Rec[k,v]. Else, if Compress[i,1]=v then, Rec[(k-1),v]= Compress[i,2].
   c. Then, construct the (k+1)th row. In the corresponding Compress array for this particular row for each column if an entry for column number 'v' is not present, then Rec[(k+1),v]=Rec[k,v]. Else, if Compress [i,1]=v then, Rec[(k+1),v]= Compress [i,2].
3. Step 2 is repeated until the full Rec array is filled.
4. Rec array is stored as the Decompressed Image File.

## IV. RESULT AND DISCUSSION

*A. Complexity analysis of the stated algorithm*

Let the size of the image be r*c. Then, at the time of Compression, 3 rows are considered at a time and for each compression of rows 'c' number of columns are read. At this process 3 rows are compressed at a time taking 3*c number of comparisons. So, for compression of the whole image, total number of compression required is $\frac{r}{3}$ *3*c=r*c ,that is O(r*c).So, for an image of size n*n, the time complexity of the compression algorithm is O(n2).

In the receiver end, the Compress array is read and the Rec array is reconstructed which also takes number of comparisons=r*c ,that is O(r*c).So, for an image of size nxn, the time complexity of the de-compression algorithm is O(n2).

*B. Test Results*

Before Compression (For each image) :
Size : 300 x 280 = 84000 [Row x Col]
Size in bytes : 1,68000 byte = 168kb
After Compression (For each image) :

Figure 1. Nature

Size in bytes : 51348 byte = 102.69 kb
Compression percentage : 38.87 %

Figure 2. Library

Size in bytes : 55234 byte = 110.64 kb
Compression percentage : 34.14 %

Figure 3. Landscape

Size in bytes : 28790 byte = 57.58 kb
Compression percentage : 65.73 %

Figure 4. Crowd

Size in bytes : 13616 byte = 27.23 kb
Compression percentage : 83.79 %

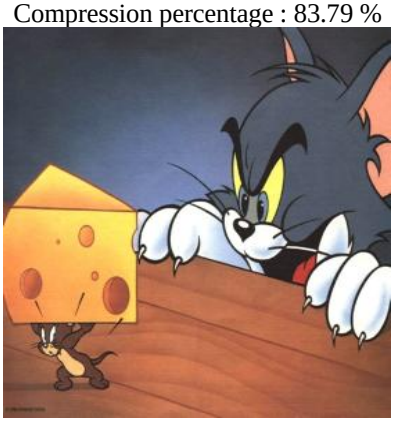

Figure 5. Tom_Jerry

Size in bytes : 35504 byte = 71.00 kb
Compression percentage : 57.73 %

Figure 6. Thumbnail

Size in bytes : 76680 byte = 153.36 kb
Compression percentage : 8.71 %

Figure 7. Model_face

Size in bytes : 63094 byte = 126.18 kb
Compression percentage : 24.89 %

### *C. Conclusion*

The algorithm proposed here is for lossless image compression ass it is evident from the algorithm, that the exact image data (pixel values) are extracted from the compressed data stream without any loss. This is possible because the compression algorithm does not ignores or discards any

original pixel value. Moreover the techniques such as approximate matching and run length encoding technique are intrinsically lossless.

This compression technique proves to be highly effective for images with large similar locality of pixel lay out. This technique will find extensive use in medical imaging sector because of its lossless characteristics and the medical images has large area of similar pixel layout pattern, like in X – ray images large area are black.

## REFERENCES

[1] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, Digital Signal Processing: System Analysis and Design. Cambridge University Press, 2002.

[2] J. M. Shapiro, "Embedded Image Coding Using Zero-Trees of Wavelet Coefficients", IEEE Transactions on Signal Processing, vol. 41, pp. 3445–3462, December 1993.

[3] A. Said and W. A. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp. 243– 250, June 1996.

[4] D. Taubman, "High Performance Scalable Image Compression with EBCOT", IEEE Transactions on Image Processing, vol. 9, no. 7, July 2000.

[5] Tse-Hua Lan and A. H. Tewfik, "Multigrid Embedding (MGE) Image Coding", Proceedings of the 1999 International Conference on Image Processing, Kobe.

[6] Civarella and Moffat. Lossless image compression using pixel reordering. Proceedings of twenty seventh Australian Computer Science conference,pp 125-132,2004.

[7] K.Veeraswamy, S.Srinivaskumar and B.N.Chatterji. Lossless image compression using topological pixel re-ordering. IET international conference, India, pp 218-222,2006.

[8] Memon and Shende. An analysis of some scanning techniques for lossless image coding. IEEE Transactions on Image Processing,9 (11),pp 1837-1848,2000.

[9] Memon and Wu X. Recent developments in context based predictive techniques for lossless image compression. The computer Journal,40,pp 127-136,1997.

[10] K.Sayood. Introduction of data compression. Acdemic press, 2nd edition,2000.

[11] D.Salomon. Data Compression. Springer,2nd edition, 2000.

## AUTHORS PROFILE

**Dr. Samir K. Bandyopadhyay**, B.E., M.Tech., Ph. D (Computer Science & Engineering), C.Engg., D.Engg., FIE, FIETE, currently, Professor of Computer Science & Engineering, University of Calcutta, visiting Faculty Dept. of Comp. Sc., Southern Illinois University, USA, MIT, California Institute of Technology, etc. His research interests include Bio-medical Engg, Image processing, Pattern Recognition, Graph Theory, Software Engg.,etc. He has 25 Years of experience at the Postgraduate and under-graduate Teaching & Research experience in the University of Calcutta. He has already got several Academic Distinctions in Degree level/Recognition/Awards from various prestigious Institutes and Organizations. He has published more than 300 Research papers in International & Indian Journals and 5 leading text books for Computer Science and Engineering. He has visited round the globe for academic and research purposes.

**Tuhin Utsab Paul** received his Bachelors degree in Computer science in 2008 and Masters degree in Computer and Information Science in 2010, both from the University of Calcutta. He is currently doing his M.Tech course in Computer science and engineering from the University of Calcutta. His research interest include Image processing, pattern recognisation, image steganography and image cryptography. He has published quite a few Research papers in International & Indian Journals and conferences. He is an IEEE member since 2009.

**Avishek Raychoudhury** received his Bachelors degree in Computer science in 2008 and Masters degree in Computer and Information Science in 2010, both from the University of Calcutta. He is currently doing his M.Tech course in Computer science and engineering from the University of Calcutta. His research interest include Image processing, image steganography, graph theory and its applications. He has published quite a few Research papers in International & Indian Journals and conferences.