LOSSY AND LOSSLESS IMAGE COMPRESSION TECHNIQUES FOR TECHNICAL DRAWINGS

Deepa Mittal[1], Prof. Manoj Mittal[2], Prof. Dr. P. K. Jamwal[3]
[1]Research Scholar, Computer Engineering Pacific Academy of Higher Education and Research University Udaipur, Udaipur, India
[2]Asst. Prof. Mechanical Engineering Department, Govt. Engg. College Jhalawar, Jhalawar, India
[3]Professor, Mechanical Engineering Dept. Rajasthan Technical University Kota, Kota, India

Abstract: Image compression is the process of reducing the amount of data required to represent digital image with no significant loss of information. We need compression for data storage & transmission in DVD's, remote sensing, video conferencing & fax. The objective of image compression is to reduce irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form. Image compression may be lossy or lossless. Lossless compression is preferred for archival purpose & often used for medical imaging, technical drawings, clipart or comics. Lossy compression methods especially used at bit rate i.e. suitable for natural images such as photographs. The lossy compression that produces imperceptible differences may be called visually lossless. This paper will focus on all image compression techniques.

Keywords: Technical drawings, remote sensing, redundancy, irrelevance, lossy, lossless.

I. INTRODUCTION

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. Data compression implies sending or storing a smaller number of bits. There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the JPEG format and the GIF format. The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple.

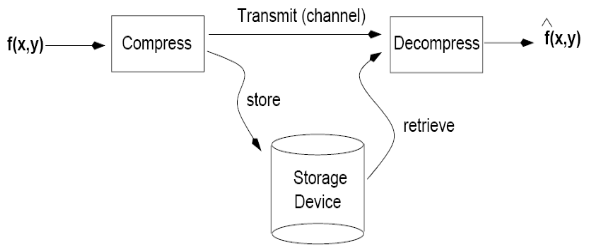

Fig 1.1 Image compression process

Although many methods are used for this purpose, in general these methods can be divided into two broad categories: lossless and lossy methods. A text file, drawings or program can be compressed without the introduction of errors, but only up to a certain extent. This is called lossless compression. Beyond this point, errors are introduced. In text, drawings and program files, it is crucial that compression be lossless because a single error can seriously damage the meaning of a text file, or cause a program not to run or damage in a mechanical product. In image compression, a small loss in quality is usually not noticeable. There is no "critical point" up to which compression works perfectly, but beyond which it becomes impossible. When there is some tolerance for loss, the compression factor can be greater than it can when there is no loss tolerance. For this reason, graphic images can be compressed more than text files or programs. Other techniques for image compression include the use of fractals and wavelets. These methods have not gained widespread acceptance for use on the Internet as of this writing. However, both methods offer promise because they offer higher compression ratios than the JPEG or GIF methods for some types of images. Another new method that may in time replace the GIF format is the PNG format.

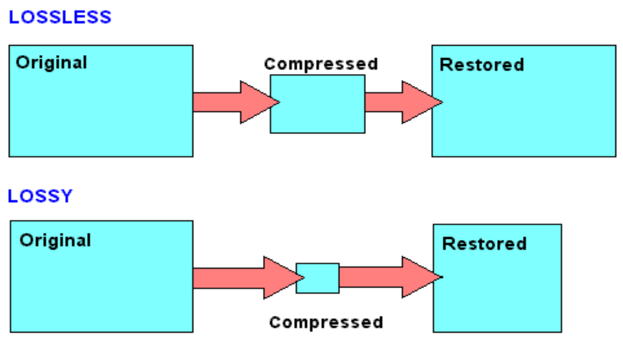

Fig 1. lossless & lossy compression process

II. VARIOUS TECHNIQUES OF IMAGE COMPRESSION

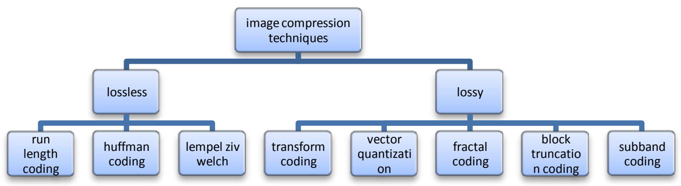

Fig. 2. Types of image compression techniques

A. Lossless compression:

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. 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. Lossless compression for legal and medical documents, computer programs, mechanical drawings exploit only code and inter-pixel redundancy. Following techniques are included in lossless compression:

B. Run Length Encoding:

Run: repetition of a symbol

Run-Length: number of repeated symbols. Run-length encoding (RLE) is a very simple form of data compression in which consecutive sequences of the same data value (runs) are stored or transmitted as a single data value and count, rather than as the original individual data elements. This is particularly useful for data that contains many such runs such as simple graphic images and faxed documents. If the original doesn't have many runs, it could increase rather than decrease the size of the data file or transmission.

For example, consider a screen containing plain black text on a solid white background. There will be many 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:

WWWWWWWWWWWWBWWWWWWWWWWWWBB
BWWWWWWWWWWWWWWWWWWWWWWWWB
WWWWWWWWWWWWWW

If we apply the run-length encoding (RLE) data compression algorithm to the above hypothetical scan line, we get the following: Interpret this as twelve W's, one B, twelve W's, three B's, etc. The run-length code represents the original 67 characters in only 18. Of course, the actual format used for the storage of images is generally binary rather than ASCII characters like this, but the principle remains the same. Even binary data files can be compressed with this method, file format specifications often dictate repeated bytes in files as padding space. However, newer compression methods such as DEFLATE often use LZ77-based algorithms, a generalization of run-length encoding that can take advantage of runs of strings of characters (such as WBWWBWWBWW). Applications: Run-length encoding performs lossless data compression and is well suited to palette-based iconic images. It does not work well at all on continuous-tone images such as photographs, although JPEG uses it quite effectively on the coefficients that remain after transforming and quantizing image blocks. Common formats for run-length encoded data include Truevision TGA, PackBits, PCX and ILBM. Run-length encoding is used in fax machines (combined with other techniques into Modified Huffman coding). It is relatively efficient because most faxed documents are mostly white space, with occasional interruptions of black. Data that have long sequential runs of bytes (such as lower-quality sound samples) can be RLE compressed after applying a predictive filter such as delta encoding.

C. Huffman Encoding:

A Huffman code is an optimal prefix code found using the algorithm developed by "David A".A Method for the Construction of Minimum-Redundancy Codes "Huffman's algorithm derives this table based on the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol. As in other entropy encoding methods, more common symbols are generally represented using fewer bits than less common symbols. Huffman's method can be efficiently implemented, finding a code in linear to the number of input weights if these weights are sorted.

The following example bases on a data source using a set of five different symbols. The symbol's frequencies are:

Symbol	Frequency
A	24
B	12
C	10
D	8
E	8

Total 186 bit (with 3 bit per code word)

The two rarest symbols 'E' and 'D' are connected first, followed by 'C' and 'D'. The new parent nodes have the frequency 16 and 22 respectively and are brought together in the next step. The resulting node and the remaining symbol 'A' are subordinated to the root node that is created in a final step.

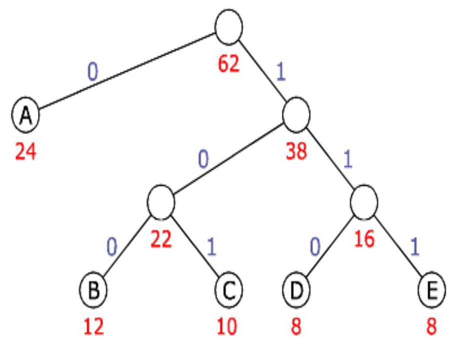

Fig. 3. Code Tree according to Huffman

Symbol	Frequency	Code	Code Length	Total Length
A	24	0	1	24
B	12	100	3	36
C	10	101	3	30
D	8	110	3	24
E	8	111	3	24

ges. 186 bit tot. 138 bit (3 bit code)

D. LZW(Lempel Ziv welch) coding :

LZW is the foremost technique for general purpose data compression due to its simplicity and versatility. It is the basis of many PC utilities that claim to "double the capacity of your hard drive". LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file. When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes. As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table. Decoding is achieved by taking each code from the compressed file, and translating it through the code table to find what character or characters it represents. LZW compression is the compression of a file into a smaller file using a table-based lookup algorithm invented by Abraham Lempel, Jacob Ziv, and Terry Welch. Two commonly-used file formats in which LZV compression is used are the GIF image format served from Web sites and the TIFF image format. LZW compression is also suitable for compressing text files. A particular LZW compression algorithm takes each input sequence of bits of a given length (for example, 12 bits) and creates an entry in a table (sometimes called a "dictionary" or "codebook") for that particular bit pattern, consisting of the pattern itself and a shorter code. As input is read, any pattern that has been read before results in the substitution of the shorter code, effectively compressing the total amount of input to something smaller. Unlike earlier approaches, known as LZ77 and LZ78, the LZW algorithm does include the look-up table of codes as part of the compressed file. The decoding program that uncompressed the file is able to build the table itself by using the algorithm as it processes the encoded input.

LZW Encoding Algorithm

```
1   Initialize table with single character strings
2   P = first input character
3   WHILE not end of input stream
4   C = next input character
5   IF P + C is in the string table
6   P = P + C
7   ELSE
8   output the code for P
9   add P + C to the string table
10  P = C
11  END WHILE
12  output code for P
```

LZW Decompression Algorithm

```
1   Initialize table with single character strings
2   OLD = first input code
3   output translation of OLD
4   WHILE not end of input stream
5   NEW = next input code
6   IF NEW is not in the string table
7   S = translation of OLD
8   S = S + C
9   ELSE
10  S = translation of NEW
11  output S
12  C = first character of S
13  OLD + C to the string table
14  OLD = NEW
    END WHILE
```

E. Lossy compression :

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. Lossy encoding is based on the concept of compromising the accuracy of the reconstructed image in exchange for increased compression. Lossy encoding techniques are capable of reproducing recognizable mono-chrome images from data that have been compressed by more than 100:1 and images that are virtually indistinguishable from the original at 10:1 to 50:1
Lossy compression techniques includes following schemes:

F. Transformation Coding :

Transform coding is a type of data compression for natural data like audio signals or photographic images. The transformation is typically lossy, resulting in a lower quality copy of the original input. In transform coding, knowledge of the application is used to choose information to discard, thereby lowering its bandwidth. The remaining information can then be compressed via a variety of methods. When the output is decoded, the result may not be identical to the original input, but is expected to be close enough for the purpose of the application.
A Simple Transform Encoding procedure maybe described by the following steps for a 2x2 block of monochrome pixels:
1. Take top left pixel as the base value for the block, pixel A.
2. Calculate three other transformed values by taking the difference between these (respective) pixels and pixel A, i.e. B-A, C-A, D-A.
3. Store the base pixel and the differences as the values of the transform.
The above transform scheme may be used to compress data by exploiting redundancy in the data. Any Redundancy in the data has been transformed to values, X_i. So we can compress the data by using fewer bits to represent the differences. i.e. if we use 8 bits per pixel then the 2x2 block uses 32 bits/ If we keep 8 bits for the base pixel, X0, and assign 4 bits for each difference then we only use 20 bits, which is better than an average 5 bits/pixel.

G. Vector quantization:

Vector quantization (VQ) is a classical quantization technique from signal processing which allows the modeling of probability density functions by the distribution of prototype vectors. It was originally used for data compression. It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in k-means and some other clustering algorithms. Vector quantization is used for lossy data compression, lossy data correction and density estimation. It is a fixed-to-fixed length algorithm. In the earlier days, the design of a vector quantizer (VQ) is considered to be a challenging problem due to the need for multi-dimensional integration. In 1980, Linde, Buzo, and Gray (LBG) proposed a VQ design algorithm based on a training sequence. The use of a training sequence bypasses the need for multi-dimensional integration. A VQ that is designed using this algorithm are referred to in the literature as an LBG-VQ. The density matching property of vector quantization is powerful, especially for identifying the density of large and high-dimensioned data. Since data points are represented by the index of their closest centroid, commonly occurring data have low error, and rare data high error. This is why VQ is suitable for lossy data compression. It can also be used for lossy data correction and density estimation.

An example of a 2-dimensional VQ is shown below:

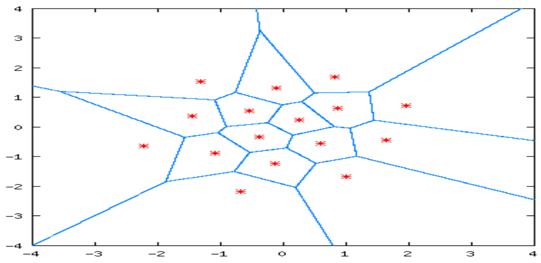

Fig. 4-dimensional vector quantization

Here, every pair of numbers falling in a particular region are approximated by a black star associated with that region. Note that there are 16 regions and 16 red stars -- each of which can be uniquely represented by 4 bits. Thus, this is a 2-dimensional, 4-bit VQ. Its rate is also 2 bits/dimension.

H. Fractal Coding:

Fractal compression is a lossy compression method for digital images, based on fractals. The method is best suited for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image. Fractal algorithms convert these parts into mathematical data called "fractal codes" which are used to recreate the encoded image. Fractal compression claims a better performance in that it produces an approximation that is closer to the original at higher compression ratios. Fractal image compression is a new technique for encoding images compactly. It builds on local self-similarities within images. Image blocks are seen as rescaled and intensity transformed approximate copies of blocks found elsewhere in the image. This yields a self-referential description of image data, which , when decoded shows a typical fractal structure.

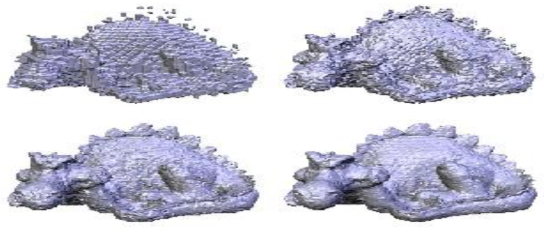

Fig : 5. Multi resolution modeling using fractal image compression technique

As with any new methodology it is interesting to study interpretations from different perspectives. Several such views of fractal image compression have been considered:

1. Iterated function systems (IFS). Such systems are operators in metric spaces and were introduced in a mathematical paper by Hutchinson in 1981, which showed that they have fractal subsets as attractors. This motivated Barnsley to search for an image compression system that models images as attractors of IFSs. Jacquin's solution of 1989 relies on a crucial modifi- cation of IFSs, namely that the mappings involved have domains that cover only part of the image. Thus, such IFSs were called local [BaHu93] or partitioned [Fish94a].
2. Self vector quantization. The basic fractal encoding is almost the same as a particular type of product code vector quantization (VQ), namely the so-called mean-removed shape-gain vector quantization (MRSG-VQ) [RaLe93]. In that approach an image block is approximated by the sum of a DC component and a scaled copy of an image block taken from the VQ codebook. Fractal encoding differs from MRSG-VQ because the codebook is not explicitly available at the decoder but rather given implicitly in a self-referential manner.
3. Self-quantized wavelet sub trees. Recently it has been noticed by Davis[Davi95a] and others that in some cases the fractal encoding is equal to a certain type of wavelet transform coding. The idea is to organize the (Haar) wavelet Coefficients in a tree and to approximate sub trees by scaled copies of other sub trees closer to the root of the wavelet tree.
4. Convolution transform coding. Also recently, it has been observed [Saup96b] that the operations carried out when searching a matching image region for a given one essentially are equivalent to a convolution operation. Only one of the convolution coefficients is selected for the fractal code. This establishes a close relation to common transform coding.

I. Block Truncation Coding:

Block Truncation Coding is a type of lossy image compression technique for grey scale images. It divides the original images into blocks and then uses a quantizer to reduce the number of grey levels in each block whilst maintaining the same mean and standard deviation. BTC has the advantage of being easy to implement. BTC achieves 2

bits per pixel (bpp) with low computational complexity.
A 256x256 pixel image is divided into blocks of typically 4x4 pixels. For each block the Mean and Standard Deviation are calculated, these values change from block to block. These two values define what values the reconstructed or new block will have, in other words the blocks of the BTC compressed image will all have the same mean and standard deviation of the original image. A two level quantization on the block is where we gain the compression, it is performed as follows:

$$y(i,j) = \begin{cases} 1, & x(i,j) > \bar{x} \\ 0, & x(i,j) \le \bar{x} \end{cases}$$

Where are pixel elements of the original block and are elements of the compressed block. In words this can be explained as: If a pixel value is greater than the mean it is assigned the value "1", otherwise "0". Values equal to the mean can have either a "1" or a "0" depending on the preference of the person or organization implementing the algorithm. This 16 bit block is stored or transmitted along with the values of Mean and Standard Deviation. Reconstruction is made with two values "a" and "b" which preserve the mean and the standard deviation. The values of "a" and "b" can be computed as follows:

$$a = \bar{x} - \sigma\sqrt{\frac{q}{m-q}}$$

Where the standard deviation, m is is the total number of pixels in the block and q is the number of pixels greater than the mean ($\bar{x}$)
To reconstruct the image, or create its approximation, elements assigned a 0 are replaced with the "a" value and elements assigned a 1 are replaced with the "b" value.

$$x(i,j) = \begin{cases} a, & y(i,j) = 0 \\ b, & y(i,j) = 1 \end{cases}$$

This demonstrates that the algorithm is asymmetric in that the encoder has much more work to do than the decoder. This is because the decoder is simply replacing 1's and 0's with the estimated value whereas the encoder is also required to calculate the mean, standard deviation and the two values to use.

J Sub band coding:
Sub-band coding (SBC) is any form of transform coding that breaks a signal into a number of different frequency bands and encodes each one independently or it decompose the input signal into different frequency bands. This decomposition is often the first step in data compression for audio and video signals.

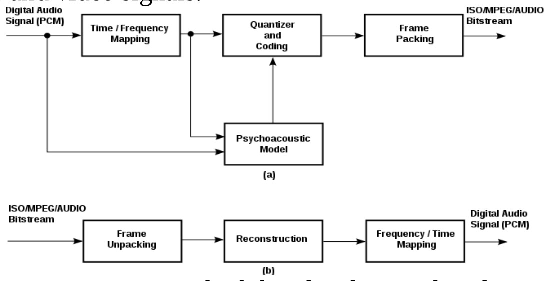

Fig. 6. process of sub band coding & decoding

After the input is decomposed to its constituents, we can use the coding technique best suited to each constituent to improve the compression performance. Each component may have different perceptual characteristics. Quantization errors that are objectionable in one component may be acceptable in a different component. Sub band algorithm consists three phases

- Analysis
- Quantization and coding
- Synthesis

1. Analysis:
Source output is passed through a bank of filters (analysis filters). Analysis filters cover the range of frequencies that make up source output. Pass band of the filters can be non-overlapping or overlapping. Output of filters are then sub sampled (also called decimation or down sampling). Justification for sub sampling: Nyquist rule (range of frequencies of output of the filter is less than input to the filter)

2. Quantization and Coding:
Selection of compression scheme and allocation of bits between sub bands is important and can have significant impact on the quality of the final reconstruction

3. Synthesis.
Encoded samples from each sub band are decoded.
Decoded values are then up sampled by inserting an appropriate number of 0s between samples.
Up sampled signals are passed through a bank of reconstruction filters.
Output of reconstruction filters are added to give final output.

III. CONCLUSION

In this paper we tried to provide all the image compression techniques. Basically there are two methods of image compression lossy and lossless. Each type has its advantages and disadvantages. We use both the types according to the use. With lossless compression, every single bit of data that was originally in the file remains after the file is uncompressed. All of the information is completely restored. Lossless compression is used for technical drawings, graphs in the mechanical field. Lossy compression reduces a file by permanently eliminating certain information, especially redundant information. When the file is uncompressed, only a part of the original information is still there (although the user may not notice it). Lossy compression is generally used for video and sound, where a certain amount of information loss will not be detected by most users.

REFERENCES

[1] Clement, T.P., The Extraction of Line-Structured Data from Engineering Drawings, PR(14), No. 1-6, 1981, pp. 43-52. WWW Version. BibRef 8100.
[2] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102. Huffman's

original article.

[3] David H. Kil and Fances Bongjoo Shin, " Reduced Dimension Image Compression And its Applications,"Image Processing, 1995, Proceedings, International Conference,Vol. 3 , pp 500-503, 23-26 Oct.,1995

[4] Ismail Avcibas, Nasir Memon, Bulent Sankur, Khalid Sayood, " A Progressive Lossless / Near Lossless Image Compression Algorithm,"IEEE Signal Processing Letters, vol. 9, No. 10, pp 312-314, October 2002.

[5] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Transactions on Information Theory, vol. 23, no. 3, pp. 337-343, May 1977.

[6] Ken Huffman. Profile: David A. Huffman, Scientific American, September 1991, pp. 54–58

[7] Ming Yang & Nikolaos Bourbakis, "An Overview of Lossless Digital Image Compression Techniques, "Circuits & Systems, 2005 48th Midwest Symposium ,vol. 2 IEEE ,pp 1099-1102,7 – 10 Aug, 2005 6.

[8] R. L. Joshi, H. Jafarkhani, J. H. Kasner, T. R. Fisher, N. Farvardin, M. W. Marcellin, and R. H. Bamberger, "Comparison of different methods of classification in subband coding of images," IEEE Transactions on Image Processing, vol. 6, pp. 1473–1486, Nov. 1997.

[9] T. Kohonen, "LVQ-.PAK Version 3.1 - The Learning Vector Quantization Program Package," LVQ Programming Team of the Helsinki University of Technology, (1995).