Data Compression Techniques: A Comparative Study

Authors:

1Pooja Jain*, 2Zeeshan Khan, 3Anurag Jain

Address For correspondence:

1, 2, 3 Dept. of Computer Science, Radharaman, Institute of Technology and Science, Bhopal, India.

Abstract - Data compression is one of many technologies that enables today’s information revolution. Image compression is an application of data compression. Image compression is now essential for applications such as transmission and storage in data bases. In this paper we review and discuss about the image compression, need of compression, its principles, and classes of compression and various algorithm of image compression. This paper attempts to give a recipe for selecting one of the popular image compression algorithms based on Wavelet, JPEG/DCT, VQ, and Fractal approaches. We review and discuss the advantages and disadvantages of data compression techniques and algorithms for compressing grayscale images; give an experimental comparison on 256x256 commonly used image of Lenna and one 400x400 fingerprint image.

Keywords - Data compression, Image compression, JPEG, DCT, VQ, Wavelet, Fractal.

I. INTRODUCTION

Data compression is one of many technologies that enables today’s information revolution. Lossless data compression is used to compact files or data into a smaller form. It is often used to package up software before it is sent over the Internet or downloaded from a web site to reduce the amount of time and bandwidth required to transmit the data. Lossless data compression has the constraint that when data is un compressed, it must be identical to the original data that was compressed. Graphics, audio, and video compression such as JPEG, MP3, and MPEG on the other hand use lossy compression schemes which throw away some of the original data to compress the files even further. 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. Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology.

II. Principles Behind Compression

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction. Redundancy reduction aims at removing duplication from the signal source (image/video). Irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System (HVS). In general, three types of redundancy can be identified:

A. Coding Redundancy

A code is a system of symbols (letters, numbers, bits, and the like) used to represent a body of information or set of events. Each piece of information or events is assigned a sequence of code symbols, called a code word. The number of symbols in each code word is its length. The 8-bit codes that are used to represent the intensities in the most 2-D intensity arrays contain more bits than are needed to represent the intensities.

B. Spatial Redundancy and Temporal Redundancy

Because the pixels of most 2-D intensity arrays are correlated spatially, information is unnecessarily replicated in the representations of the correlated pixels. In video sequence, temporally correlated pixels also duplicate information.

C. Irrelevant Information

Most 2-D intensity arrays contain information that is ignored by the human visual system and extraneous to the intended use of the image. It is redundant in the sense that it is not used. Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible.

* Corresponding Author Email-Id

jain.pooja1025@gmail.com
Multimedia Data & Size/Duration & Bits/ Pixel or Bits/Sample & Uncompressed Size (B for bytes) & Transmission Bandwidth (b for bits) & Transmission Time

<table>
<thead>
<tr>
<th>Multimedia Data</th>
<th>Size/Duration</th>
<th>Bits/ Pixel or Bits/Sample</th>
<th>Uncompressed Size (B for bytes)</th>
<th>Transmission Bandwidth (b for bits)</th>
<th>Transmission Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A page of text</td>
<td>11&quot;x8.5&quot;</td>
<td>Varying resolution</td>
<td>4.8KB</td>
<td>32.64 Kbps/page</td>
<td>1.1 - 2.2 sec</td>
</tr>
<tr>
<td>Telephone quality speech</td>
<td>10 sec</td>
<td>8 bps</td>
<td>80KB</td>
<td>64 Kbps/sec</td>
<td>22.2 sec</td>
</tr>
<tr>
<td>Grayscale Image</td>
<td>512x512</td>
<td>8 bpp</td>
<td>262KB</td>
<td>2.1 Mbps/Image</td>
<td>1 min 13 sec</td>
</tr>
<tr>
<td>Color Image</td>
<td>512x512</td>
<td>24 bpp</td>
<td>786KB</td>
<td>6.29 Mbps/Image</td>
<td>3 min 39 sec</td>
</tr>
<tr>
<td>Medical Image</td>
<td>2048x1680</td>
<td>12 bpp</td>
<td>5.16MB</td>
<td>41.3 Mbps/Image</td>
<td>23 min 54 sec</td>
</tr>
<tr>
<td>SHD Image</td>
<td>2048x2048</td>
<td>24 bpp</td>
<td>12.58 MB</td>
<td>100 Mbps/Image</td>
<td>58 min 15 sec</td>
</tr>
</tbody>
</table>

### III. NEED OF COMPRESSION

The Table 1 show the qualitative transition from simple text to full-motion video data and the disk space transmission bandwidth, and transmission time needed to store and transmit such uncompressed data.

| Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required. The prefix kilo- denotes a factor of 1000 rather than 1024. |

The examples given in the Table I clearly illustrate the need for sufficient storage space, large transmission bandwidth, and long transmission time for image, audio, and video data.

At the present state of technology, the only solution is to compress multimedia data before its storage and transmission, and decompress it at the receiver for play back. For example, with a compression ratio of 32:1, the space, bandwidth, and transmission time requirements can be reduced by a factor of 32, with acceptable quality.

### IV. Different classes of compression techniques

Two ways of classifying compression techniques are mentioned here.

**A. Lossless vs. Lossy compression**

In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. However lossless compression can only achieve a modest amount of compression. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards redundant information. However, lossy schemes are capable of achieving much higher compression. Under normal viewing conditions, no visible loss is perceived (visually lossless).

### B. Predictive vs. Transform coding

In predictive coding, information already sent or available is used to predict future values, and the difference is coded. Since this is done in the image or spatial domain, it is relatively simple to implement and is readily adapted to local image characteristics. Differential Pulse Code Modulation (DPCM) is one particular example of predictive coding. Transform coding, on the other hand, first transforms the image from its spatial domain representation to a different type of representation using some well-known transform and then codes the transformed values (coefficients). This method provides greater data compression compared to predictive methods, although at the expense of greater computation.

### V. CONVENTIONAL UNIVERSAL DATA COMPRESSION

Most universal compression techniques collect statistical information to identify the highest frequency patterns in the data. In this section some of the previous attempts to address the universal compression problem by utilising statistical approaches are reviewed.

**A. Adaptive Compression Algorithm**

Typically data compression algorithms have been implemented through a two-pass approach. In the first pass, the algorithm collects information regarding the data to be compressed, such as the frequencies of characters or substrings. In the second pass, the actual encoding takes place. A fixed encoding scheme is required to ensure that the decoder is able to retrieve the original message. This approach has been improved through so called adaptive compression, where the algorithm only needs one pass to compress the data [25]. The main idea of adaptive compression algorithms is that the encoding scheme changes as the data being compressed. Thus, the encoding of the nth symbol is based on the characteristics of the data until position n - 1 [25]. A key advantage of adaptive compression is that it does not require the entire message to be loaded into the memory before the compression process can start.

Adaptive Huffman coding [26] is a statistical lossless compression where the code is represented using a binary tree structure. This algorithm gives the most frequent characters that appear in the file to be compressed shorter codes than those characters which are less frequent. Top nodes in the tree store the most frequent characters. To produce the compressed version of a file, the algorithm simply traverses the tree from the root node to the target character. At each step the algorithm will encode 1 if it has moved to the left branch and 0 otherwise. The Huffman tree creates a unique variable length code for each character. The novelty of the Adaptive Huffman compression is that nodes swap locations within the tree during the compression. This allows the algorithm to adapt its compression as the frequency of the symbols changes throughout the file.

Adaptive Arithmetic Coding (AC) is a statistical compression
that learns the distribution of the source during the compression process [23]. AC has historic significance as it was the best alternative to Huffman coding after a gap of 25 years [23]. Adaptive AC encodes the source message to variable-length code in such a way that frequently used characters get fewer bits than infrequently used ones. Unlike other compression techniques which replace blocks of the data with smaller code words, AC encodes the entire message into a single real number. Hence, the AC is based on the mathematical fact that the cumulative probability of a particular sequence of characters has a unique and small subinterval within [0; 1). The algorithm simply calculates the cumulative probability of each symbol within the message at a time. The count for each encountered symbol increases after it has been encoded. Then, the cumulative count table is updated accordingly. Each symbol leads the algorithm to a new smaller subinterval based on its probability. The compression process starts with the initial interval [0; 1), and keeps iterating until it reads the entire source. The final output is a single real number selected from the smallest identified subinterval. In the decompression process, the algorithm receives the encoded real number as input and starts from the initial interval. The algorithm divides the main interval into smaller subintervals according to which section the input falls in. The new subinterval becomes the new main interval and output the corresponding symbol. This process keeps iterating until the entire message is decoded. Another adaptive compression is Prediction by Partial Matching (PPM) which was proposed by Cleary and Witten in 1984 [21]. This algorithm is classified as statistical compression. The main ideas of this algorithm are context modelling and prediction. PPM tries to predict the probability of a particular character being in a specific location from the previous n symbols previously occurred. To this end, a table of statistical information is created, which stores strings of size o with the probabilities of their following characters. The number of previous symbols o is called the order of the PPM model. If the symbol to be encoded has not been previously encountered in the context then an escape symbol is encoded and the o is reduced. In the next iteration the algorithm uses a table of size o = 1. This process keeps iterating until the algorithm encounters the target symbol or it concludes that the symbol has never been seen before. In this case (i.e., the symbol was never seen before), the algorithm updates the statistical tables with a new entry (i.e., add the new symbol with its probability and its o previous symbols). The probability of the newly added symbol is l = M, where M is the size of the source alphabet [21]. The actual encoding of the symbols is done with Arithmetic Coding compression. A variant of the algorithm called PPMD [27] where it increments the probability of the escape symbol every time it is used.

Lempel and Ziv developed a universal compression system in 1977 known as LZ77 [28]. Later, in 1978, an improved version was presented which is referred to as LZ78 [21]. These algorithms are classified as dictionary-based compression. The main idea is to replace frequent substrings in the data with references that match the data that has already been passed to the encoder. The encoder creates a dictionary for the common substring in the file and uses it as reference for the data. Both encoder and decoder must share the same dictionary to perform the compression/decompression process. In LZ77 a sliding window of predefined length is used to maintain the dictionary. Data are scanned via a look-ahead buffer and compared with the previous data within the sliding window. Once a match is found the dictionary is updated. LZ78 eliminated the need for a fixed size window and builds the dictionary out of all previously seen symbols. LZ78 works very well where the frequent symbols are distributed in isolated locations in the file. Unlike LZ77, strings in LZ78 can be longer which gives flexibility and higher performance. The adaptability of the LZ algorithms comes from the fact that the algorithms update the dictionary contents during the compression process to tune the algorithm’s behaviour as the frequency of the symbols change throughout a file.

In some sense adaptive compression algorithms represent a form of intelligent systems that adapt their behavior based on the given encoding situation. The primary advantage of adaptive compression is that it uses a single pass to compress data. This accelerates the compression process and makes it appealing for use in the cases where speed is favoured over performance. The use of a single pass has the disadvantage of sacrificing the ability to see ahead in the file to determine future fluctuations which may contain valuable information for the encoding scheme. In adaptive compression once the algorithm makes the decision to encode the data in a particular way it will not be able to change it, even if the algorithm later learns a better way of encoding this information. Another problem with adaptive compression algorithms is that they are sensitive only to changes in relation to the particular redundancy type which they are specialised to detect, hence, they are not able to exploit different forms of redundancy [24]. Also, because they learn the alphabetic distribution during the compression time, they cannot handle drastic changes in the files streams as, for example, is the case with changes from text to picture. Consequently, adaptive schemes are not recommended for heterogeneous files where multiple types of data are stored in a single file and their constituent parts have different degrees of compressibility.

Cormack and Horspool introduced Dynamic Markov Compression (DMC) in 1987 [25]. DMC is another statistical compression algorithm using on context modelling and prediction. Similarly to PPM, it uses Arithmetic Coding to encode predicted symbols. The difference, however, is that the model operates at the bit level rather than on bytes. Although this model has similar performance to PPM it is not widely used.

http://www.ijars.in
This is another approach to develop universal compression algorithms and to achieve higher compression. It has been reported that this approach show superior performance to standard compression algorithms. A successful composition of compression algorithms can be achieved in two ways. Firstly, by running a number of compression algorithms successively. Secondly, by combining a number of simple compression algorithms and heuristically selecting them where they are expected to perform well.

A well-known composite compression system is Bzip2. Bzip2 is a free, open-source, composite and lossless compression algorithm developed in 1996 [29]. It has been widely used in many commercial compression applications, such as WinZip [30]. In this algorithm, the data are compressed through several compression and transformation techniques in a particular order. The reverse order is used in the decompression process. Bzip2 compresses files using the Burrows-Wheeler transform and the Huffman coding. This technique has been found to be better than LZ77 and LZ78 [29]. Bzip2 is known for being slow in the compression process and much faster in decompression.

Katz in [31] proposed an algorithm that combines LZ77 with Huffman coding as an improvement for the PKZIP archiving tool. This algorithm is referred to as Deflate. In the original implementation of LZ77, the algorithm tries to match strings within a sliding window with a look-ahead buffer. Matched strings are used to build a dictionary. Each repeated string is replaced each with a triplet (pointer, length and next symbol). The next symbol element is needed in case there is no exact match in the dictionary (e.g., William and Will). In Deflate a variant of LZ77 is used, which eliminates the third element and encodes a pair (pointer, length). Unmatched characters are written in the compressed stream [22]. In the original implementation of the Deflate algorithm, compressed data consisted of a sequence of blocks corresponding to successive blocks of input data. These blocks can be of different lengths based on the various prefix codes used and the memory available to the encoder. The algorithm has three options for each input block: i) Apply no compression, which is used if the data is already compressed, ii) Compress with fixed Huffman code and iii) Compress with dynamic Huffman code. Each uncompressed block is individually compressed using the previously described modification of LZ77 and then Huffman coding. Thus, each block is composed of two parts: i) a Huffman tree that describes the data and ii) the compressed data itself. Deflate has been widely implemented in many commercial compression applications, such as gzip, the HTTP protocol, the PPP compression protocol, PNG (Portable Network Graphics) and Adobe’s PDF (Portable Document File) [22].

Another composite compression system is the Lempel-Ziv Markov-chain Algorithm (LZMA) [22]. LZMA uses a similar approach to Deflate. The difference is that it uses range encoding instead of Huffman coding (the range encoder is an integer-based version of Arithmetic Coding) [22]. This enhances the compression performance, but at the expense of increasing the encoder’s complexity. In LZMA, the input stream is divided into blocks, each block describing either a single byte, or an LZ77 sequence with its length and distance.

The main disadvantage of standard composite compression algorithms is that they are unreliable when dealing with heterogeneous files, i.e., files that are composed of multiple data types such as archive files (e.g., ZIP and TAR). This is because most standard composite compression schemes follow deterministic procedures that involve applying several compression models in a particular order. These procedures are selected based on the designer experience or experimental evidence which demonstrated their superior performance under certain conditions. Whilst these methods have proven to be successful in achieving high compression ratios, the use of deterministic steps to perform compression entails the disadvantage of making the encoding decision fixed and unable to deal with unpredictable types of data.

Another approach to developing composite compression system is allowing the system to heuristically select and apply compression algorithms where they are expected to perform best. This idea has been explored by Hsu in [24]. Hsu’s system segmented the data into blocks of a fixed length (5 KB) and then compressed each block individually with the best compression algorithm. Four compression algorithms were used in Hsu’s system, namely, Arithmetic Coding, Runlength encoding, LZW and JPEG for image compression. The system works in two phases. In the first phase, the blocks are scanned to determine the compressibility and the contents of each block. The compressibility of the blocks is calculated by measuring three different quantitative metrics: alphabetic distribution, average run length and string repetition ratio. The system considers the blocks already compressed if these measures were under a predefined threshold.

Consequently, already compressed blocks are skipped. The files contents are determined using a modified Unix file command. This command is able to classify ten different types of data. The modified file command works by examining the first, middle and last (if it exists) 512 bytes and thereafter comparing their patterns with collections of known patterns from the Unix operating system. In the second phase, the actual compression takes place, where the system passes the blocks to the appropriate compression model based on the gathered information from the first phase. Experimentation with 20 heterogeneous test files revealed that the proposed system was able to outperform other commercial compression systems with 16% saving on average. The main disadvantage of Hus’s system is that the size of the blocks is fixed to 5KB, which limits the algorithm’s ability to identify the true boundaries of heterogeneous fragments within the data. Moreover, the modified Unix command used detects ten file types only and is
not reliable enough to guarantee that the blocks are passed to the optimal compression algorithm. This is because the system assumes that one particular compression algorithm is suitable for all files of a particular type. This assumption, however, is flawed as the compressibility of the data depends on the forms of regularities within the data and whether the compression algorithm is designed to capture them. These regularities are not necessarily to be correlated with a particular file type. Thus, at least in principle, two text files may be better compressed with two different compression algorithms depending on their contents.

VI. What does a typical image coder look like?
A typical lossy image compression system which consists of three closely connected components namely (a) Source Encoder (b) Quantizer, and (c) Entropy Encoder. Compression is accomplished by applying a linear transform to decorrelate the image data, quantizing the resulting transform coefficients, and entropy coding the quantized values.

A. Source Encoder (or Linear Transformer)
Over the years, a variety of linear transforms have been developed which include Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) [1], Discrete Wavelet Transform (DWT)[13] and many more, each with its own advantages and disadvantages.

B. Quantizer
A quantizer simply reduces the number of bits needed to store the transformed coefficients by reducing the precision of those values. Since this is a many-to-one mapping, it is a lossy process and is the main source of compression in an encoder. Quantization can be performed on each individual coefficient, which is known as Scalar Quantization (SQ). Quantization can also be performed on a group of coefficients together, and this is known as Vector Quantization (VQ). Both uniform and non-uniform quantizers can be used depending on the problem at hand.

C. Entropy Encoder
An entropy encoder further compresses the quantized values lossless to give better overall compression. It uses a model to accurately determine the probabilities for each quantized value and produces an appropriate code based on these probabilities so that the resultant output code stream will be smaller than the input stream. The most commonly used entropy encoders are the Huffman encoder and the arithmetic encoder, although for applications requiring fast execution, simple run-length encoding (RLE) has proven very effective.

A. JPEG: DCT-Based Image Coding Standard
The JPEG/DCT still image compression has become a standard recently. JPEG is designed for compressing full-color or gray-scale images of natural, real-world scenes. To exploit this method, an image is first partitioned into non-overlapped 8×8 blocks. A discrete Cosine transform (DCT) [10, 14] is applied to each block to convert the gray levels of pixels in the spatial domain into coefficients in the frequency domain. The coefficients are normalized by different scales according to the quantization table provided by the JPEG standard. Conducted by some psycho visual evidence. The quantized coefficients are rearranged in a zigzag scan order to be further compressed by an efficient lossless coding strategy such as run length coding, arithmetic coding, or Huffman coding. The decoding is simply the inverse process of encoding. So, the JPEG compression takes about the same time for both encoding and decoding. The encoding/decoding algorithms provided by an independent JPEG group [14] are available for testing real-world images. The information loss occurs only in the process of coefficient quantization. The JPEG standard defines a standard 8×8 quantization table [14] for all images which may not be appropriate. To achieve a better decoding quality of various images with the same compression by using the DCT approach, an adaptive quantization table may be used instead of using the standard quantization table.

B. Image Compression by Wavelet Transform
1. What is a Wavelet Transform?
Wavelets are functions defined over a finite interval and having an average value of zero. The basic idea of the wavelet transform is to represent any arbitrary function (t) as a superposition of a set of such wavelets or basis functions. These basis functions or baby wavelets are obtained from a single prototype wavelet called the mother wavelet, by dilations or contractions (scaling) and translations (shifts). The Discrete Wavelet Transform of a finite length signal x(n) having N components, for example, is expressed by an N x N matrix. For a simple and excellent introduction to wavelets, see [3].

2. Why Wavelet-based Compression?
Despite all the advantages of JPEG compression schemes based on DCT namely simplicity, satisfactory performance, and availability of special purpose hardware for implementation; these are not without their shortcomings. Since the input image needs to be “blocked,” correlation across the block boundaries is not eliminated. This results in noticeable and annoying “blocking artifacts” particularly at low bit rates as shown in Fig.1. Lapped Orthogonal Transforms (LOT) [5] attempt to solve this problem by using smoothly overlapping blocks. Although blocking effects are reduced in LOT compressed images, increased computational complexity of such algorithms do not justify wide replacement of DCT by LOT.
Fig. 1: (a) Original Lena Image, and (b) Reconstructed Lena with DC component only, to show blocking artifacts.

Over the past several years, the wavelet transform has gained widespread acceptance in signal processing in general and in image compression research in particular. In many applications wavelet-based schemes (also referred as sub band coding) outperform other coding schemes like the one based on DCT. Since there is no need to block the input image and its basis functions have variable length, wavelet coding schemes at higher compression avoid blocking artifacts. Wavelet-based coding [2] is more robust under transmission and decoding errors, and also facilitates progressive transmission of images. In addition, they are better matched to the HVS characteristics. Because of their inherent multi-resolution nature [6], wavelet coding schemes are especially suitable for applications where scalability and tolerable degradation are important.

C. VQ Compression

A vector quantizer is composed of two operations. The first is the encoder, and the second is the decoder. The encoder takes an input vector and outputs the index of the codeword that offers the lowest distortion. In this case the lowest distortion is found by evaluating the Euclidean distance between the input vector and each codeword in the codebook. Once the closest codeword is found, the index of that codeword is sent through a channel (the channel could be computer storage, communications channel, and so on). When the encoder receives the index of the codeword, it replaces the index with the associated codeword. The fundamental idea of VQ[4] for image compression is to establish a codebook consisting of code vectors such that each code vector can represent a group of image blocks of size m x m, (m=4 is always used). An image or a set of images is first partitioned into m x m non overlapping blocks which are represented as m2-tuple vectors, called training vectors. The size of training vectors can be very large. For example, a 512 x 512 image contributes 16,384 training vectors. The goal of codebook design is to establish a few representative vectors, called code vectors of size 256 or 512, from a set of training vectors. The encoding procedure is to look for a closest code vector in the codebook for each non overlapped 4 x 4 block of an image to be encoded. The most important work is to design a versatile codebook. Nasrabadi and King [11] give a good review of VQ. Chen’s comparison [16] indicates that a codebook developed based on LBG [12] algorithm generally has higher PSNR values over some other schemes despite its slow off-line training. In this paper, we adopt LBG algorithm for training a codebook of size 256x256 to meet a desired 0.5 bpp compression ratio.

D. Fractal Compression

Fractal image coding was introduced in the late 1980s and early 1990s [20]. It is used for encoding/decoding images in Encarta/Encyclopedia [15]. Fractal coding is based on the Collage theorem and the fixed point theorem [15, 19] for a local iterated function system consisting of a set of contraction affine transformations [15]. A fractal compression algorithm first partitions an image into non overlapping 8x8 blocks, called range blocks and forms a domain pool containing all of possibly overlapped 16x16 blocks, associated with 8 isometries from reflections and rotations, called domain blocks. For each range block, it exhaustively searches, in a domain pool, for a best matched domain block with the minimum square error after a contractive affine transform is applied to the domain=block.

A fractal compressed code for a range block consists of quantized contractively coefficients in the affine transform, an offset which is the mean of pixel gray levels in the range block, the position of the best matched domain block and its type of isometry. The decoding is to find the fixed point, the decoded image, by starting with any initial image. The procedure applies a compressed local affine transform on the domain block corresponding to the position of a range block until all of the decoded range blocks are obtained. The procedure is repeated iteratively until it converges (usually in no more than 8 iterations).

Two serious problems that occur in fractal encoding are the computational demands and the existence problem of best range-domain matches [19]. The most attractive property is the resolution-independent decoding property. One can enlarge an image by decoding an encoded image of smaller size so that the compression ratio may increase exponentially [15,18]. An algorithm based on [20] using range and domain block matches of fixed sizes is written and is used for a comparison in this paper [17].

Advantages and Disadvantages Of Various Compression Algorithm

There are some advantages and disadvantages of various algorithms which are shown in table 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet</td>
<td>High Compression Ratio State-Of-The-Art</td>
<td>Coefficient quantization Bit allocation</td>
</tr>
<tr>
<td>JPEG</td>
<td>Current Standard</td>
<td>Coefficient (dct) quantization Bit allocation</td>
</tr>
</tbody>
</table>
Image compression algorithms based on Wavelet Transform [9], JPEG/DCT [7], Vector Quantization [16], and Fractal [15] methods were tested for 256×256 real image of Lenna and 400×400 fingerprint image. The results of performance are shown in Table 3, 4 and 5.

In Table 3, 4 and 5 the performance of different algorithms is shown in which there is PSNR value and CPU Time (Encoding and Decoding) is shown. And we summarize the comparison of Compression ratio of different algorithm in Table 6 given below.

**TABLE 4: Performance of Coding Algorithms on a 400x400 Fingerprint Image of 0.5bpp**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PSNR Values</th>
<th>Encoding Time</th>
<th>Decoding Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet</td>
<td>36.71</td>
<td>0.8 sec</td>
<td>0.7 sec</td>
</tr>
<tr>
<td>JPEG</td>
<td>34.27</td>
<td>0.2 sec</td>
<td>0.2 sec</td>
</tr>
<tr>
<td>VQ</td>
<td>28.26</td>
<td>0.6 sec</td>
<td>0.7 sec</td>
</tr>
<tr>
<td>Fractal</td>
<td>27.21</td>
<td>6.3 hrs</td>
<td>3.5 sec</td>
</tr>
</tbody>
</table>

**TABLE 5: Performance of Coding Algorithms on a 400x400 Fingerprint Image of 0.25bpp**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PSNR Values</th>
<th>Encoding Time</th>
<th>Decoding Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelet</td>
<td>32.47</td>
<td>0.7 sec</td>
<td>0.5 sec</td>
</tr>
<tr>
<td>JPEG</td>
<td>29.64</td>
<td>0.2 sec</td>
<td>0.2 sec</td>
</tr>
<tr>
<td>VQ</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Fractal</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The decoded images of Lenna based on the four approaches (a) Wavelet Transform, (b) JPEG, (c) Vector Quantization, (d) Fractal are shown in Fig. 3.
approaches (a) Wavelet Transform, (b) JPEG, (c) Vector Quantization, (d) Fractal are shown in Fig. 4.

Fig.4: Decoded fingerprints by (a) Wavelet, (b) JPEG, (c) VQ, (d) Fractal algorithms.

VIII. Conclusion

The data compression algorithms require a great deal of processing power to analyze and then encode data into a smaller form. Creating a general purpose compressor that can take advantage of parallel computing should greatly reduce the amount of time it requires to compress files, especially large ones. We have reviewed and summarized the characteristics of image compression, need of compression, principles behind compression, different classes of compression techniques and various image compression algorithms based on Wavelet, JPEG/ DCT, VQ, and Fractal approaches. Experimental comparisons on 256x256 commonly used image of Lenna and one 400x400 fingerprint image suggest a recipe described as follows. Any of the four approaches is satisfactory when the 0.5 bits per pixel (bpp) is requested. However, for a very low bit rate, for example 0.25 bpp or lower, the embedded zero tree wavelet (EZW) approach is superior to other approaches. For practical applications, we conclude that (1) Wavelet based compression algorithms are strongly recommended. (2) DCT based approach might use an adaptive quantization table, (3) VQ approach is not appropriate for a low bit rate compression although it is simple, (4) Fractal approach should utilize its resolution-free decoding property for a low bit rate compression.

REFERENCES