Fractal Image Compression with varying Sub Image

Vedant Rastogi, Jaspreet Kaur
Associate Professor Department of Computer Sc. & Engineering IET Alwar,
M.Tech Student Department of Information & Technology IET Alwar

Abstract— *Fractal image compression partitions an image in to square range blocks that are coded via self-references to other parts of the image itself. In this project we propose an algorithm for Fractal Image Compression (FIC) that covers an image with square blocks without overlapping and introduces domain blocks. We define local contractive collection of affine transformation mapping domain block D to the range block R. For each range block, a corresponding domain block and symmetry block is developed so that the domain block looks mostly like part of the image. The algorithm is written in Matlab, and is tested on various images taken from real life. It is found that it produces satisfactory results.*

Keywords— *Fractal Image Compression, Fractal Iterated system.*

I. Introduction

With the advance of the information age the need for mass information storage and fast communication links grows. Storing images in less memory leads to a direct reduction in storage cost and faster data transmissions. These facts justify the efforts, of private companies and universities, on new image compression algorithms. Images are stored on computers as collections of bits, representing pixels or points forming the picture elements. Since the human eye can process large amounts of information, many pixels are required to store moderate quality images. Most data contains some amount of redundancy, which can sometimes be removed for storage and replaced for recovery, but this redundancy does not lead to high compression ratios. An image can be changed in many ways that are either not detectable by the human eye or do not contribute to the degradation of the image. The standard methods of image compression come in several varieties. The current most popular method relies on eliminating high frequency components of the signal by storing only the low frequency components (Discrete Cosine Transform Algorithm). This method is used on JPEG (still images), MPEG (motion video images), H.261 (Video Telephony on ISDN lines), and H.263 (Video Telephony on PSTN lines) compression algorithms.

A. Fractal Image compression

Imagine a special type of photocopying machine that reduces the image to be copied by half and reproduces it three times on the copy (see Figure 1). What happens when we feed the output of this machine back as input? Figure 1.2 shows several iterations of this process on several input images. We can observe that all the copies seem to converge to the same final image, the one in 1.2(c). Since the copying machine reduces the input image, any initial image placed on the copying machine will be reduced to

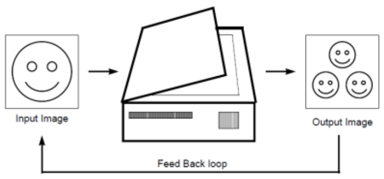

Figure 1 A copy machine that makes three Reduced copies of the input image.

a point as we repeatedly run the machine; in fact, it is only the position and the orientation of the copies that determines what the final image looks like.

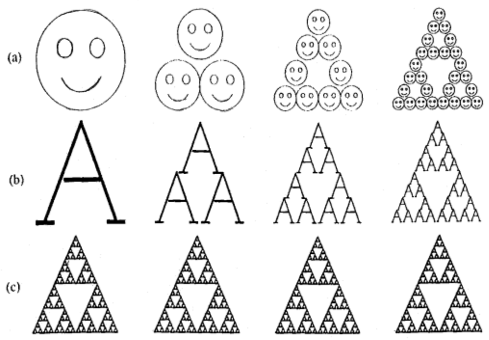

Figure 2 First three copies

The way the input image is transformed determines the final result when running the copy machine in a feedback loop. However we must constrain these transformations, with the limitation that the transformations must be contractive (see contractive box), that is, a given transformation applied to any two points in the input image must bring them closer in the copy. This technical condition is quite logical, since if points in the copy were spread out the final image would have to be of infinite size. Except for this condition the transformation can have any form. In practice, choosing transformations of the form:-

$$wi\begin{bmatrix} xi \\ yi \end{bmatrix} = \begin{bmatrix} aibi \\ cidi \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} ei \\ fi \end{bmatrix}$$

is sufficient to generate interesting transformations called affine transformations of the plane. Each can skew, stretch, rotate, scale and translate an input image. A common feature of these transformations that run in a loop back mode is that for agiven initial image each image is formed from a transformed (and reduced) copies ofitself, and hence it must have detail at every scale. That is, the images are fractals. This method of generating fractals is due to John Hutchinson [3], and more information about the various ways of generating such fractals can be found in books by Barnsley [4] and Peitgen, Saupe, and Jurgens [P1, P2].

B. Need for Compression

A picture may be worth a thousand words, but we require a lot of computer memory to store it. Images are stored on computers as collections of bits representing pixels. About 8 million bits are required to store even moderate quality images[6]**.**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 [7]. The figures in 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. The figures in 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.

TABLE I
Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required.

Multimedia Data	**Size/Duration**	**Bits/Pixel or Bits/Sample**	**Uncompressed Size (B for bytes)**	**Transmission Bandwidth (b for bits)**	**Transmission Time (using a 28.8K Modem)**
A page of text	11" x 8.5"	Varying resolution	4-8 KB	32-64 Kb/page	1.1 - 2.2 sec
Telephone quality speech	10 sec	8 bps	80 KB	64 Kb/sec	22.2 sec
Grayscale Image	512 x 512	8 bpp	262 KB	2.1 Mb/image	1 min 13 sec
Color Image	512 x 512	24 bpp	786 KB	6.29 Mb/image	3 min 39 sec
Medical Image	2048 x 1680	12 bpp	5.16 MB	41.3 Mb/image	23 min 54 sec
SHD Image	2048 x 2048	24 bpp	12.58 MB	100 Mb/image	58 in 15 sec

C. Principal behind Compression

A common characteristic of most images is that the neighbouring 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.

- **Spatial Redundancy** or correlation between neighboring pixel values.
- **Spectral Redundancy** or correlation between different color planes or spectral bands.
- **Temporal Redundancy** or correlation between adjacent frames in a sequence of images (in video applications) [7].

D. Image Compression

Image Compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages. Image compression is a mapping from a higher dimensional space to lower dimensional space. Image compression plays an important role in many multimedia applications, such as image storage and transmission. The basic goal of image compression is to represent an image with minimum number of bits of an acceptable image quality.

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 [11].

Image Compression is of two types:-

a. Lossless Image Compression

Lossless data compression is a class of data compression algorithms that allows the exact original data to be reconstructed from the compressed data. The term lossless is in contrast to lossy data compression, which only allows an approximation of the original data to be reconstructed, in exchange for better compression rates.

Lossless data compression is used in many applications. For example, it is used in the popular ZIP file format and in the UNIX tool gzip. It is also often used as a component within lossy data compression technologies.

Lossless compression is used when it is important that the original and the decompressed data be identical, or when no assumption can be made on whether certain deviation is uncritical. Typical examples are executable programs and source code. Some image file formats, like PNG or GIF, use only lossless compression, while others like TIFF and MNG may use either lossless or lossy methods.

b. Lossy Image Compression

A lossy compression method is one where compressing data and then decompressing it retrieves data that is different from the original, but is close enough to be useful in some way. Lossy compression is most commonly used to compress multimedia data (audio, video, still images), especially in applications such as streaming media and internet telephony.

E. Existing Research work

In 1997, Yung-ching, Bin-kai&Jia-sung proposed an algorithm for a low bit-rate region based fractal compression, they improved the performance of quad tree segmentation by adaptive threshold. In 2000, Md. Elsherife, Mohsen proposed and implemented an algorithm for matching criteria in fractal image compression.

Later in 2000, Hannes Matthias and Saupe, implemented an algorithm for Region-based fractal image compression. In it a fractal coder partitions an image into blocks that are coded via self-references to other parts of the image itself. A fractal coder that derives highly image-adaptive partitions and corresponding fractal codes in a time-efficient manner using a region-merging approach. The proposed merging strategy leads to improved rate-distortion performance compared to previously reported pure fractal coders and it is faster than other state-of-the-art fractal coding methods.

In 2002, KamelBelloulata and Janusz proposed a new approach to fractal image coding that permits region-based functionalities; images are coded region by region according to a previously-computed segmentation map. They use rectangular range and domain blocks, but divide boundary blocks into segments belonging to different regions. Since this prevents the use of standard dissimilarity measure.

Thereafter in 2007, Fan Ce, proposed a new fast fractal encoding algorithm for the process of searching and matching process in fractal image compression. The number of domain blocks searched to find the best match for each range block and corresponding encoding time are much reduced by elimination domain blocks not searching using the current minimum distortion and variance difference between the range block and domain block. Fractal Encoding can now be done in reduced time.

II. THEORETICAL MODELLING

The fractal system is based on the iterated function systems

A. Iterated Function Systems

In iterated function systems or IFSs are a method of constructing fractals; the resulting constructions are always self-similar. IFS fractals, as they are normally called, can be of any number of dimensions, but are commonly computed and drawn in 2D. The fractal is made up of the union of several copies of itself, each copy being transformed by a function. The canonical example is the Sierpinski gasket. The functions are normally contractive which means they bring points closer together and make shapes smaller. Hence the shape of an IFS fractal is made up of several possibly-overlapping smaller copies of itself, each of which is also made up of copies of itself, ad infinitum. This is the source of its self-similar fractal nature.

B. Fractal Transformation Theory

Fractal transform theory is the theory of local IFS. Although local IFS does complicate the theory of fractal image compression, in practice it simplifies the process.

A global transformation on a space X is a transformation, which is defined on all points in X; whereas, a local transformation is one whose domain is a subset of the space X and the transformation need not act on all points in X. Rather than allowing an IFS to act upon only on the whole domain, it is convenient to allow an IFS to act upon domains that are subsets of the space. This type of IFS is called a local IFS. The idea of fractal image compression, as briefly mentioned above, is to find subspaces (or sub-images) of the original image space, which can be regenerated using an IFS. Where possible, if one IFS can be used in place of several IFS's which reproduce similar sub-images, it is more efficient in terms of storage space to use that one IFS. It is more likely that an image will require more than one IFS to reproduce a compressed image, which closely resembles the original.se a page size corresponding to A4 which is 210mm (8.27") wide and 297mm (11.69") long. The margins must be set as follows:-

C. Algorithm

- Input a binary image, call it M.
- Cover M with square range blocks. The total set of range blocks must cover M, without overlapping.
- Introduce the domain blocks D; they must intersect with M. The sides of the domain blocks are twice the sides of the range blocks.
- Define a collection of local contractive affine transformations mapping domain block D to the range block R_i.
- For each range block, choose a corresponding domain block and symmetry so that the

domain block looks most like the part of the image in the range block.

- Write out the compressed data in the form of a local IFS code.
- Apply a lossless data compression algorithm to obtain a compressed IFS code.

In practice these steps can be carried out on a digital image. The compression is attained by storing the coefficients of the transformations, rather than storing the image pixel by pixel.

Advantages of fractal image compression:-

One advantage of fractal image compression over traditional transform coding is that fractal compression takes advantage of local scale invariance. Several natural image features such as, straight edges and constant regions are unchanged by rescaling.

- ➢ As the image increases in size the size of the compressed file stays constant.
- ➢ This method does not require a code-book unlike the JPEG method.
- ➢ Compression ratio is high.
- ➢ Decoding stage of the algorithm is independent of the to-be-reconstructed image.
- ➢ In the decoding stage the image is reconstructed quickly through a number of recursive operations.
- ➢ The reconstructed image has an adequate quality.

Disadvantages of fractal image compression:-

- ➢ Fractal image compression basically exploits the self-similarity present in the image. Hence images with random content are not likely to be compressed well as only few similarities of different sizes are likely to exist.
- ➢ The encoding complexity of fractal image compression is high when compared to decoding complexity. The encoding complexity is due to the search for similarity between the blocks in the image.

III. Implementation

The following is an explanation of 'RIFSbat', a simple fractal image compression program and 'fdec' the corresponding decompression program. These two programs can be run on Mat lab and only compress grayscale square images that are in .png format, although further changes can be implemented later to account for non-square images and other formats. This batch program runs through the program 10 times, allowing 10 different tolerances. Ten different mat files are saved representing 10 different compressed images. The difference in the compressed files is not the number of bits needed to store the files (this is the same as long as the range size is the same), but the time needed to produce the compressed files based on the error tolerance. The tolerances are determined by what the variable min0 is set equal to the minimum error, denoted by the variable minerr is defined by the norm of the difference between the range blocks and the transformed domain blocks. 'RIFSbat' searches for the transformation with least error from domain blocks to range blocks. During the first loop, the tolerance is min0 = 10. So, the program searches for a transformation until it finds a transformation with minerr< min0.

As min0 increases, more error is allowed. With each run, the tolerance of allowable error increases by 10.

First, the user must enter the name of the pgm image file in the first line of the program: M = getpng ('imagename.png'). In the examples, we use 'sisters.png'. Then, the user specifies the desired range block size by setting rsize equal to the length of the side of the desired range block. Presently, rsize is set equal to 4, which allows range blocks of size 4×4. We next create the domain blocks, which are twice the size of the range blocks, in this case 8×8. In determining which mapping will need to be made from the domain blocks to the range blocks, we will need to compare the domain blocks to the range blocks. To accurately compare these blocks, they must be the same size. So, we do some averaging over the domain blocks which allows us to shrink the domain blocks to half of its size in order to match the size of the range blocks. Originally, each domain block is 8×8. The averaging only takes place over each distinct block of 2×2 pixels within the domain block. Then the average grayscale value in each 2×2 block of pixels is represented in one pixel in the scaled domain blocks. Next the program cycles through each domain block and tests each symmetry that is stored, along with the four possible gray scales for the best transformation that will map to a given range block. Depending on the block size chosen for the range blocks, one may need to vary the number of iterations applied to the seed image in order to arrive at the attractor image. As more iterations of the IFS are applied to the image, the clearer the attractor will become. After the nth iteration, the image produced corresponds to the A_n compact set as discussed in the local IFS theory

IV. Results

The proposed algorithm is applied on test pictures and those taken from real life. And it is found that the compression works very well on real life images also.

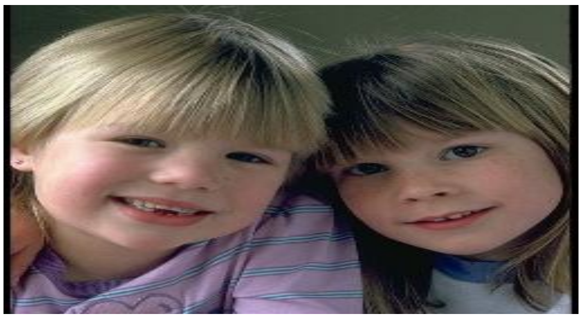

Figure 3 Uncompressed girls.png image

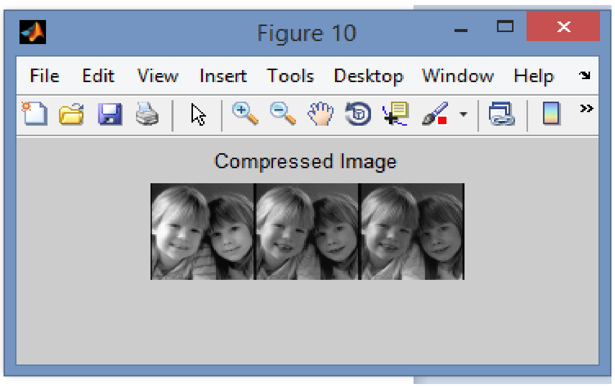

Figure 4 Compressed Image

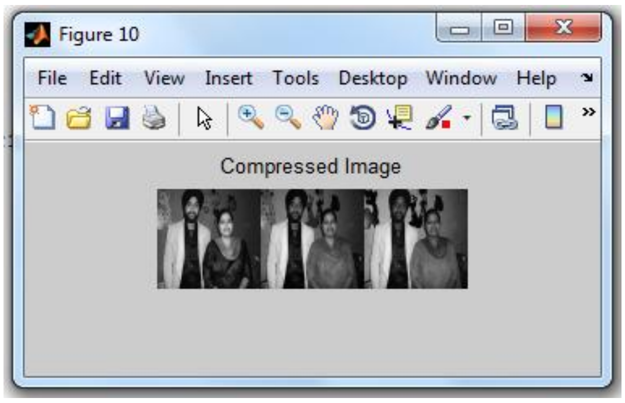

Figure 5 Compression of a real life image

A test image as shown in Figure 3, provided by Matlab is compressed first. The compression results are shown in Figure 4, 5, 6 and Figures 7 and shows the compression of real life images

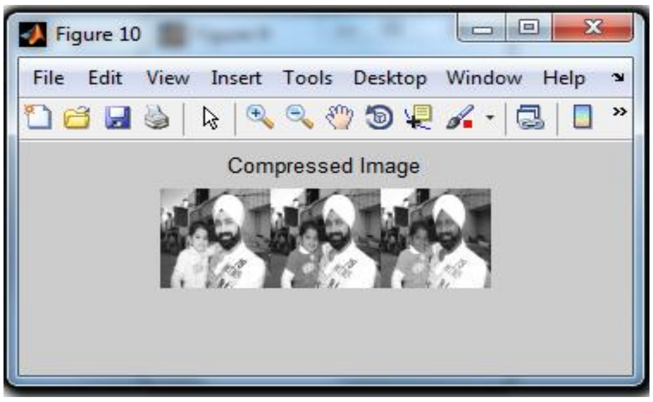

Figure 6 Compression of another real life image

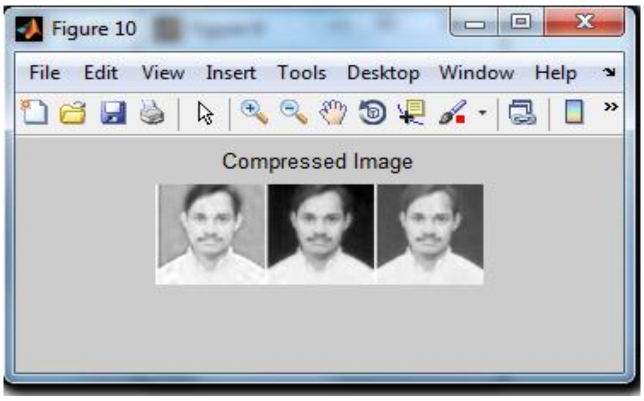

Figure 7 Compression of a third real life image

The implementation of this simple method of fractal compression will produce great compression ratios. Considering that each pixel requires 8 bits to store the values of 0 to 255, to store an 256×256 image pixel by pixel would require 65536 bytes (around 65KB). Using 'RIFSbat' and 'fdec' with any chosen error, to store an image of this size with a range block size of 4×4 pixels only requires 11776 bytes. The compression ratio is better than 5:1. Of course, increasing the range block size to 8×8 pixels improves the compression to only 2688 bytes, with a compression ratio of approximately 24:1. The larger range block sizes allow higher compression ratios. The time needed to produce the attractor image is based on how much error is allowable in the transformations. The larger the error, the quicker the compression. The use of the image will determine the required amount of compression and image quality.

V. Conclusions

With each iteration the quality of the compressed image is enhanced. This can be noted with the gradual increase in the peak signal values, which in turn results in a higher PSNR with each iteration. Enhanced Varying Sub-Image Shape algorithm achieves a doubling of compression ratio over traditional Block Based Fractal Image compression technique. The most time-consuming part of the system is the exhaustive search used to find the seed block of the transformation. For the block-based system, the exhaustive search is carried out for every block in image. But, for the varying sub-image shape system, it is carried out only for the first block in each region. It shows that the time taken to extend each region is relatively small, which results in a faster encoding time than block-based system.

REFERENCES

[1] R.D. Boss, E.W. Jacobs, "Fractals-Based Image Compression," NOSC Technical Report 1315, Sept. 1898. Naval Ocean Systems Center, SanDiego CA 92152-5000.

[2] Jacquin A., Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding, Doctoral Thesis, Georgia Institute of Technology, 1989.

[3] John E. Hutchinson, Fractals and Self Similarity. Indiana University Mathematics Journal, Vol. 35, No.5. 1981

[4] Barnsley M., Fractals Everywhere. Academic Press. San Diego, 1989.

[5] Y. Fisher, Fractal Image Compression, Siggraph 92 course Notes.

[6] Y. Fisher, Editor: Fractal Image Compression, Theory and application to digital images, Springer Verlag,1994.

[7] Image Compression - from DCT to Wavelets a Review, Subhasissaha.

[8] C.Gonzalez; E. Woods Richard,Digital Image Processing, 2nd edition, 2002.

[9] M. F. Barnsley and L. P. Hurd, Fractal Image Compression. Wellesley, MA: A. K. Peters, 1993.

[10] Arnaud E. Jacquin, Fractal Image Coding: A Review, Proceeding of the IEEE,Vol.81,No.10,October1993.