International Journal of Engineering Research and Development
ISSN: 2278-067X, Volume 1, Issue 4 (June 2012), PP.44-48
www.ijerd.com

Fractal Image Compression Technique Using Fixed Level of Scaling

Mallikarjuna Swamy M L[1], Nagamani K[2]
[1]*M.Tech Student, Digital Communication Engg. R V College of Engg. Bangalore-59*
[2] *Assistant Professor, Dept. of Telecommunication Engg. R V College of Engg. Bangalore-59*

Abstract**—**Fractal compression is a technique for encoding images compactly. It is built on local self similarities within the images. Image blocks are seen as rescaled and intensity transformed approximate the copies of blocks found elsewhere in the image. In this paper the Fractal image compression technique using fixed level of scaling is proposed in which the time of encoding process is considerably reduced. This work explains the predefined fixed level of scaling instead of scanning the parameter(S) Space [0,1].This novel point is used to point the reasonable estimation of S and use them in a encoding process as predefined values. The algorithm is implemented in Matlab for a standard set of images.

Keywords— Domain and Range block, Fractal, Image compression, Iterative Function System (IFS)

I. INTRODUCTION

Fractal image compression (FIC) [1-5] is one of the recent methods of compression. It has generated much interest due to its promise of high compression ratios and to the advantage of having very fast decompression. Another advantage of FIC is its multi-resolution property. This method, which is based on the collage theorem [1], shows that it is possible to code fractals images by means of some contractive transformations defining an Iterated Function System (IFS). As natural signals do not often possess global self transformability, Jacquin [3] proposed to look for local or partial transformability what led to the first algorithm of compression by Local Iterated Function Systems (LIFS).

Fractal Image Compression (FIC) is expected to be a promising compression method for digital images in term of high compression ratios and fidelity. However, due to the unacceptable encoding time, use of this compression method is limited to areas where the encoding time is not critical. Therefore, acceleration of the encoding is indispensable to a variety of uses many attempts were done to speeding FIC using different methods [11, 12, 13]. One of these studies is applying Fixed level of Scaling in FIC to speeding image compression.

II. PRINCIPLE OF FRACTAL CODING

In the encoding phase of fractal image compression, the image of size N x N is first partitioned into non overlapping range blocks R_i , { R_1 , R_2 ,...R_p } of a predefined size BxB. Then, a search codebook (domain pool) is created from the image taking all the square blocks (domain blocks) D_j , { D_1 , D_2 ,...D_q } of size 2Bx2B, with integer step L in horizontal or vertical directions. To enlarge the variation, each domain is expanded with the eight basic square block orientations by rotating 90 degrees clockwise the original and the mirror domain block. The range-domain matching process initially consists of a shrinking operation in each domain block that averages its pixel intensities forming a block of size BxB.

For a given range R_i , the encoder must search the domain pool BigM for best affine transformation w_i , which minimizes the distance between the image R_i and the image $w_i(D_i)$, (*i.e.* $w_i(D_i) \approx R_i$). The distance is taken in the luminance dimension not the spatial dimensions. Such a distance can be defined in various ways, but to simplify the computations it is convenient to use the Root Mean Square RMS metric. For a range block with n pixels, each with intensity r_i and a decimated domain block with n pixels, each with intensity d*i* the objective is to minimize the quality

$$E(Ri, Di) = \sum_{i=1}^{n} (sDi + 0 - Ri)^2$$

This occurs when the partial derivatives with respect to s and o are zero. Solving the resulting equations will give the best coefficients s and o [5].The parameters that need to be placed in the encoded bit stream are S*i*, Oi, index of the best matching domain, and rotation index. The range index i can be predicted from the decoder if the range blocks are coded sequentially. The coefficient s_i represents a contrast factor, with | S*i* |<1.0, to make sure that the transformation is contractive in the luminance dimension, while the coefficient o*i* represents brightness offset.

III. THE EFFECT OF FIXED LEVEL OF SCALING, *S*

The important parameter that was investigated is the predefined four fixed level of scaling *s* instead of scanning the parameter(S) Space [0,1]. To do this, a large number of experiments with exhaustive search for *s* were performed S =[0.45 0.60 0.82 0.96]; is the best selected values of *s*.

IV. THE PROPOSED ALGORITHM

1. Input a binary image, call it M.
2. Cover M with square range blocks. The total set of range blocks must cover M, without overlapping.
3. Introduce the domain blocks D; they must intersect with M. The sides of the domain blocks are twice the sides of the range blocks.
4. Define a collection of local contractive affine transformations mapping domain block D to the range block R_i.
5. For each range block, choose a corresponding domain block, symmetry and one of the best scaling value out of four values, S =[0.45 0.60 0.82 0.96] so that the domain block looks most like the part of the image in the range block.
6. Save the compressed data in the form of a local IFS code T()=[posi sym S O].

V. ENCODER AND DECODER

A. Encoder

First enter the name of the image file, In the examples we use 'lena.pgm'. Then 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, called I1. I1 is a 4×4 block at this point. We subtract the average of the domain block from each entry in the domain block to account for possible darkening of the decompressed image. The resulting scaled domain block is D.

Now, we save 8 different transformations of each domain block in an eight dimensional monster matrix called *pool*I. The transformations include the original domain block, a 90°, 180° and a 270° rotation, a horizontal flip, and a vertical flip, as well as the transform of the domain block and a 180° rotation of the transformed domain block. We introduce a vector s, which contains different specific scaling to transform the grayscale of the domain block to make a better match to a range block.

At this point, 'Encoder' goes through all of the range blocks and offsets each of them by subtracting the average of the range block from each entry in the range block. Now we can equally compare the domain to the range blocks. We save the offset of the range blocks in *o*, which we will add back to the image later.

Next the program cycles through each domain block D_{ij} and tests each symmetry that is stored in *poolI*, along with the four possible gray scales for the best transformation that will map to a given range block. When the best map is found, the location of that domain block i0 and j0, the best symmetry m0 of the domain block, the best scaling s0, and the offset o is saved in the five dimensional matrixes. It is the entries of this matrix that determine the number of bytes needed to store the compressed image file. Once this information is saved in a file, it is possible to compress that file even more by applying a lossless coding algorithm. It is from the matrix, T, that the program 'decoder' can regenerate the image.

It is important to note that each transformation from the original 8×8 domain is a contraction mapping because the domain must be scaled by ½ in order to map the domain to the range. Also, the information stored in each $(k,l,:)$ entry of *T* represent the coefficients of the mappings w_i, $i = 1,2,3,\ldots N$ that make up the *N* local IFS mappings. The image regenerated after all the mappings in *T* are applied to some seed image, is the attractor of the local IFS.

B. Decoder

In order to regenerate the attractor of the contractive transformations found, we must use the program 'decoder' along with the saved information from 'fcomp'. First we load the correct data using the name that we saved it under in the batch file. Then we initialize a matrix to perform the mappings on. This matrix must be the same size as the original image. Although, in the program we initialize the seed image to all zeros, which is a uniformly gray image, choosing another image as the seed to the local IFS will arrive at the same result.

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 iteration of the IFS are applied to the image, the clearer the attractor will become. After the *n*th iteration, the image produced corresponds to the A_n compact set. First, the domain blocks of the seed image must be created and rescaled to the size of the range blocks. Then using the *T* matrix, the domain blocks are transformed and mapped to the range blocks. This process is repeated for each iteration. The attractor, *M*, is then output to be displayed on the screen. A result contains examples of an original image, and the consecutive images regenerated after iterating the local IFS created for that image. The quality of the attractors vary depending on the size of the range blocks used and the error allowed in finding an appropriate transformation form domain block to range block

VI. SIMULATION RESULTS

Our implementation of this simple method of fractal compression produced great compression ratios. Considering that each pixel requires 8 bits to store the values of 0 to 127, to store a 128x128 image pixel by pixel would require 16384 bytes (around 16KB). Using 'encoder' and 'decoder' with any chosen error, to store an image of this size with a range block size of 4×4 pixels only requires 5120 bytes. The compression ratio is better than 3.2:1. Of course, increasing the range block size to 8x8 pixels improves the compression to only 1280 bytes, with a compression ratio of approximately 12.8:1. 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.

Figure 1 is the original image in this example. Figure 2, 3, and 4 are the first through third iterations of the fractal compression transformations with *minerr* = 10. Referring to table 1 we can tell that this compressed file took about 18 minutes to complete compression for 4×4 pixel range block and 3.5 minutes to complete compression for 8×8 pixel range block. The attractor image that is regenerated is close to the original image, but the time needed to accomplish compression is not desirable. With a 4×4 pixel range block, a decent error is probably about 20or 30. Although to compress an image with this error takes about 123.1060 seconds, if the error is greater than that, the image quality becomes very low and blocky. For the 4×4 pixel range blocks, three different implementations based on a change in the allowable error of images are in figure 2, 3, 4 and 5 with the errors, *min0*, are 2, 10, and 80 respectively.

	Range block size 4x4			Range block size 8X8		
Emin	PSNR	CR	Te	PSNR	CR	Te
2	24.0705	3.2000	162.7200	20.3853	12.8	21.4200
4	24.0691	3.2000	149.9320	20.3853	12.8	21.6400
6	24.0652	3.2000	138.6690	20.3853	12.8	21.6700
8	24.0597	3.2000	129.4580	20.3853	12.8	22.2000
10	24.0528	3.2000	123.1060	20.3853	12.8	21.6300
12	24.0395	3.2000	111.7740	20.3867	12.8	20.7000
14	24.0302	3.2000	103.2560	20.3864	12.8	20.4300
16	24.0107	3.2000	95.5505	20.3859	12.8	20.1700
18	24.9993	3.2000	88.2430	20.3852	12.8	19.4700
20	23.9895	3.2000	81.9620	20.3834	12.8	19.4600

Table 1. The effect of Emin on compression performance parameters for different range size

By looking at Figure 6 we notice that the image quality is not as high as the previous case. The reason for this is because the range size in these images is 8×8 pixels. With *minerr* =10 to provide a comparison between image quality and the time used to produced the compressed file, which can be found in table 1.

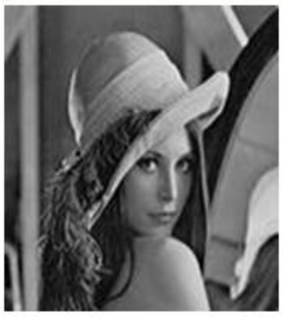
Figure 1 Original Lena image

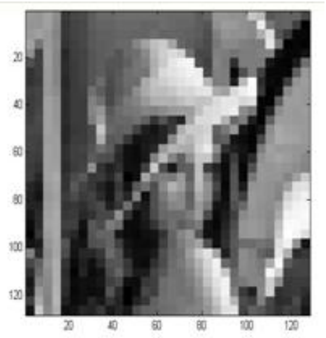

Figure 2 Lena 4x4 Range blocks

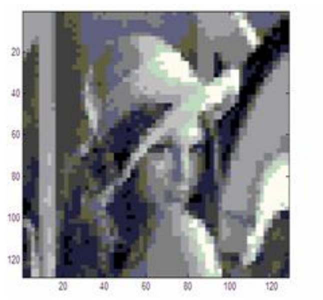
Figure 3 Lena 4x4 Range blocks
Min0=10 Iteration 2

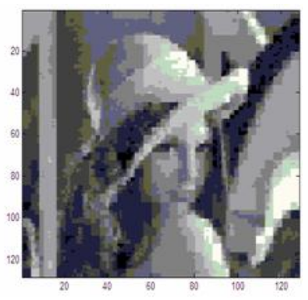
Figure 4 Lena 4x4 Range blocks
Min0=10 Iteration 3

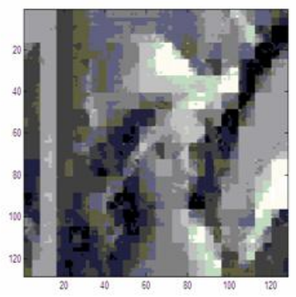
Figure. 5 Lena 4x4 Range blocks
Min0=10 Iteration 4

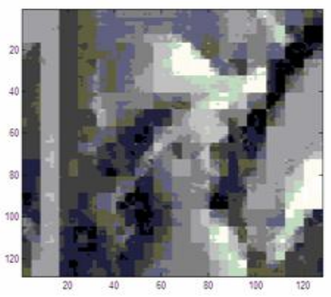
Figure. 6 Lena 8x8 Range blocks
Min0=10 Iteration 4

VII. CONCLUSION

In this work we presented a new method for fractal image compression to reduce encoding time. Centrally, our algorithm employed predefined values for fixed level of scaling factor rather than sweeping the entire parameter space during search. However, the searching of domain block is still carried out by global search; therefore the computation is still very large during encoding. In the future we intend to further develop this approach using moment features for fractal compression to minimize encoding time.

VIII. ACKNOWLEDGEMENTS

We thank The Staff and Department of Telecommunication Engineering, RV College of Engineering, Bangalore for providing us an opportunity and resource to carry out the work.

REFERENCES

[1]. Barnsley, Michael and Lyman P. Hurd; Fractal Image Compression, AK Peters, Ltd., 1993.

[2]. Fisher, Y., "Fractal Image Compression Theory and Application", University of California,Institute for Nonlinear Science, Springer-Verlay, New York, Inc, 1995

[3]. H. Miar Naimi, M. Salarian :"A Fast Fractal Image Compression Algorithm Using Predefined Values for Contrast Scaling" WCECS 2007, October 24-26, 2007, San Francisco, USA.

[4]. Viswanath Sankaranarayanan, "Fractal Image Compression Literature Survey", 19th October, 1998.

[5]. M. Hassaballa , M.M. Makky and Youssef B. Mahdy $^+$ "Fast Fractal Image Compression Method Based Entropy" Electronic Letters on Computer Vision and Image Analysis 5(1):30-40, 2005

[6]. H. Miar Naimi, M. Salarian, "A Fast Fractal Image Compression Algorithm Using Predefined Values for Contrast Scaling". Proceedings of the World Congress on Engineering and Computer Science 2007, San Francisco, USA

[7]. Richard J. Prokop and Anthony P. Reeves, "A Survey of Moment-Based Techniques For Unoccluded Object Representation and Recognition".

[8]. Tong, C.,"Fast Fractal Image Encoding Based on Adaptive Search", IEEE Transaction on Image Processing, Vol.10.No.9, 2001.

[9]. Frigaard, C., "Fast Fractal 2D/3D Image Compression ", Report, Institute of Electronic Systems, Alborg University, Laboratory of Image Analysis, 1995.

[10]. Ning, L., "Fractal Imaging", Academic Press, 1997.

[11]. Sangwine, S. J., Horne, R., "The Color Image processing Handbook", Champan & Hall, 1998

[12]. Dr. Loay E. George and Dr. Eman A. Al-Hilo:"Speeding-up Fractal Color Image Compression Using Moments Features Based on Symmetry Predictor," Eighth International Conference on Information Technology ieee2011

[13]. George, L., Al-Hilo, E., " Speeding-Up Color FIC using Isometric Process based on Moment Predictor", International Conference on Future Computer and Communication, ICFCC2009, April 3 - 5, IEEE, ISBN: 978-1-4244-3754 (PP 607-611)

[14]. Al-Hilo, E., George, L., "Color FIC by Zero-Mean Method", 4th International Conference of Information Technology and Multimedia (ICIMU' 2008), pp. 631 636

[15]. Al-Hilo, E., George, L., "Speeding- up Fractal Color Image Compression Using Moments Features", Digital Image Computing: Techniques and Applications, DICTA 2008, pp. 486-490, IEEE, ISBN: 978-0-7695-3456-5

[16]. Hartenstein, H., Ruhl, M., and Saup, D., "Region- Based Fractal Image Compression", IEEE Transaction on Image Processing, vol.9, No.7, July 2000.

[17]. Jihee Choi, Sheng Van, and Hong Jeong "An Automated Method Based on Second Order Moment for Defect Extraction in Photomask Images,"ICACT 2009.