

Stochastic Image Compression Using Fractals

Aditya Kapoor*
Department of
Computer Science,
Institute of
Engineering and
Technology, Kanpur
adi_ull@yahoo.com

Kush Arora**
Department of
Computer Science,
Institute of
Engineering and
Technology, Kanpur
kusharora@lycos.com

Ajai Jain
Department of
Computer Science and
engineering, IIT
Kanpur, India
ajain@cse.iitk.ac.in

G.P. Kapoor
Department of
Mathematics, IIT
Kanpur, India
gp@iitk.ac.in

Abstract

Fractal objects like Sierpinski triangle and Fern have very high visual complexity and low storage-information content. For generating computer graphic images and compression of such objects, Iterated Function Systems (IFS) {[3] , [1]} are recently being used. The main problem in fractal encoding using IFS is large amount of time taken for the compression of the fractal object. Our endeavor in the present paper is to use a stochastic algorithm to improve upon the compression time as well as compression ratio obtained in [1], while maintaining the image quality. Our results show that we are able to reduce time taken for compression of Image by 55% - 80% and the size by 60% - 80% as compared to the non-stochastic algorithm.

1. Introduction

In the present paper, we discuss stochastic image coding based on the fractal theory of iterated contractive transformations. We mainly deal with solving the inverse problem of finding such transformations corresponding to a given image. In an effort to proceed in this direction, A.E Jacquin [1] first proposed a novel approach to image coding. His method is to construct contractive affine transformations for which the given image is a fixed point. The theories of Iterated Function System (IFS) and Recurrent Iterated Function System [4] form the basis for fractal image compression techniques [3]. The images produced by IFS are due to iterated applications of a deterministic image transformation to an initial image, an algorithm commonly used for the construction of deterministic fractal objects [6].

(* Currently student of University of Louisiana at Lafayette)

(** Currently student of University of Akron, Ohio)

The main problem with fractal compression is the large time taken for the compression. When we time stamped Lena and Baboon; and experimented by taking all eight isometries for the edge block as mentioned in Jacquin's paper (Using PIII, 700MHz processor) it took a large time to compress the images (e.g. for a 2x2 block of baboon time of compression was 54 minutes 07 sec).

This large amount of time is due to checking the eight isometries for each edge block. Reducing the number of isometries resulted in distortion of the image. To overcome this difficulty, we tried to use probabilistic measures and found that the use of probability is quite efficient. For the first certain number of blocks, the isometry with highest probability is chosen and the same isometry is implemented on rest of the edge blocks. This approach is found to be highly successful since, without distorting the image, we are able to reduce the compression time by up to 80% *(Table 6). Another improvement that our approach brings over the approach of Jacquin is that our algorithm makes further reduction in image size than that obtained by Jacquin's algorithm resulting in small storage requirement.

2. Theoretical background

2.1 Structure of transformation

Constructing a single transformation τ for the whole image with acceptable distortion is difficult. Therefore, the image is partitioned into small blocks and for each block a transformation is constructed. This set of transformations serves as the transformation for the whole image. Thus, we have the transformation τ in the following form:

$$\forall \mu \in M, \tau(\mu) = \sum_{0 \leq i < N} (\tau\mu)_{R_i} = \sum_{0 \leq i < N} \tau_i(\mu_{D_i})$$

where $\wp = \{R_i\}_{0 \leq i < N}$ denotes the nonoverlapping partition of the image in N range cells, usually squares. Here τ_i denotes transformation from domain cell D_i to range cell R_i and is the composition of two transformation \mathfrak{S} and \wp :

$\tau_i = \mathfrak{S} \circ \wp$, where \mathfrak{S} is the **massic** part and \wp is the **geometric** part described below.

2.2 Geometric part \wp

A domain cell of size 2B (2B x 2B) is mapped by geometric transformation on to a range cell of size B (B x B). Pixel value of the contracted image on the range block are average of 4 pixel values of domain block:

$$(\wp\mu)_{i,j} = (\mu_{2i,2j} + \mu_{2i+1,2j} + \mu_{2i+1,2j+1}) / 4$$

Where $i,j \in \{0, \dots, B-1\}$

2.3 Massic part \mathfrak{S}

These transformations affect pixel values of the transformed domain blocks. The transformations are:

- Absorption at constant gray level g:
 $(\theta\mu)_{i,j} = g, g \in \{0, \dots, 255\}$
- Luminance shift by
 $\Delta g : (\tau\mu)_{i,j} = \mu_{i,j} + \Delta g, \Delta g \in \{0, \dots, 255\}$
- Contrast scaling by $\alpha \in [0, 1] : (\sigma\mu)_{i,j} = \alpha\mu_{i,j}$
- Color reversal : $(\rho\mu)_{i,j} = 255 - \mu_{i,j}$
- Isometries

The isometries are (for block size = B) :

1. Identity
2. Orthogonal reflection about mid vertical axis, mid horizontal axis, diagonal (i=j) and another diagonal (i+j=B-1).
3. Rotation around center of block by (+90, 180, -90) degrees.

2.4 Distortion Measure

Suppose μ is the image block of size B and ν is its approximation. The mean squared distortion is defined by:

$$d_{L_2}(\mu, \nu) = \sum_{0 \leq i, j < B} (\mu_{i,j} - \nu_{i,j})^2$$

3. Encoding of digital images

During the course of our investigations several coding algorithms were developed. Although few of these

algorithms gave reasonable results for the ‘‘Lena’s’’ image, they were not very satisfactory when we applied them on the ‘‘Baboon’s’’ image. These algorithms were not able to clearly distinguish Shade, Midrange and Edge blocks. This led to the development of an algorithm that was able to clearly classify these blocks distinctly.

3.1 Class of domain blocks

For range block of size B, the maximal pool of domain blocks is a ‘‘huge’’ set of all possible blocks of size 2B. To trim this pool to manageable limit, we consider only those blocks as domain blocks, which fall under a sliding window of size 2B, which is shifted over the image horizontally and vertically by a fixed number of pixel values. The pool so obtained is further divided in the shade, midrange and edge blocks.

3.2 Transformation pool

For each range block μ , there is a transformation which depends on whether μ is a shade, midrange or edge block. If we have a domain block ν , then if μ is a

- **Shade Block:** We approximate it by uniformly gray block whose gray level is average of pixel values of μ . For these blocks we have to store a single value.
- **Midrange Block:** It is composition of contrast scaling and luminance shift:

$$\mathfrak{S}(\wp \nu) = \alpha(\wp \nu) + \Delta g$$

where α is contrast scaling factor that takes value in the set $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and Δg so that average gray levels of range block and scaled domain blocks are the same.

- **Edge Block :** It is composition of contrast scaling, luminance shift and an isometry

$$\mathfrak{S}(\wp \nu) = i(\alpha(\wp \nu)) + \Delta g$$

α is chosen such that μ and $\wp(\nu)$ have same dynamic range.

$$\alpha = \min\left[\frac{dr(\mu)}{dr(\wp \nu)}, \alpha_{\max}\right]$$

where **dynamic range** (dr) = (highest pixel value – lowest pixel value + 1) of the block under consideration. α so computed is quantized to nearest value in the set $\{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Then Δg is computed so that average gray levels of range block and scaled domain blocks are the same. Finally isometry with minimum distortion is selected.

3.3 Classification and search for edge, midrange and shade block

In the following, we propose an algorithm to classify various blocks, which is essential for reduction of encoding time and increase of compression ratio.

3.3.1 Edge block. For the classification of the edge block, let $f(x,y)$ be the grey level of image at (x,y) . The gradient of f at the point (x,y) is

$$\nabla f = (G_x, G_y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right).$$

Let,

$$F = \text{mag}(\nabla f) = [G_x^2 + G_y^2]^{1/2}$$

denote maximum rate of increase in $f(x,y)$ per unit distance in the direction of ∇f . We approximate F by $[|G_x| + |G_y|]$ as this is simple to implement. The quantities G_x and G_y are computed as:

$$G_x = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

$$G_y = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)$$

where $Z_1 \dots Z_9$ are grey levels of 3×3 part of the image

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

and G_x and G_y are derivatives of Z_5 .

Algorithm:

Whenever the $\text{mag}(\nabla f)$ crosses a threshold value (30: found through experimentations) a counter 'k' is incremented. If $k > \text{the block size}$, the block was designated as an edge block.

3.3.2 Midrange block. Many different approaches are tried to get satisfactory result. These are the blocks with fine texture. Such blocks are hard to detect in a pool of thousands of blocks. Initially we tried to classify those blocks as midrange blocks, which were neither edge blocks nor shade blocks. This approach did not give satisfactory results due to the presence of mixed block (blocks which have characteristics of both edge blocks and midrange blocks). Next, we tried the same approach that we applied for edge blocks keeping the threshold value less than that for edge blocks. This approach too did not give the desired results. Finally, we settled for an approach based on variance of the block under consideration. This approach gives quite satisfactory results and is described by the following algorithm:

Algorithm:

var = variance of block

$r = 1 - 1 / (1 + \text{var})$

If $(r < 0.7)$, block is classified as midrange

Note: The bound 0.7 on r was found by experimentation.

3.3.3 Shade block. For the classification of the shade block the dynamic range 'dr' of the block was calculated and for $(\text{dr} < 15)$ block was classified as shade block. Here we settled for $(\text{dr} < 15)$ after experimenting with images, which gave us large number of blocks as shade blocks. Shade block take minimum time for classification and require minimum storage space, as only the average grey level of the block is to be stored.

4. Output file

The output file has one line of code for each range block in the image. The number of integers in the output file decides the type of block. For the shade block only the average grey level of the block is stored. For mid range block the luminance shift, contrast-scaling factor and the co-ordinates of the upper left corner of the domain block are stored. The compressed file obtained after the encoding consists of several lines for edge block where each line consists of (i) Luminance shift (ii) contrast-scaling factor (iii) co-ordinates of the upper left corner of the domain block and (iv) the isometry number. Using our probabilistic approach, only isometry number for first certain number (say 100) of blocks needs to be written. So, if we have, say 4096 edge blocks, in a particular image, we are able to reduce the size of image further by $(4096 - 100)$ integer values. Therefore, if we assume that integer takes 2 bytes in general, then we are able to further reduce the file size by 7992 bytes. The file generated has the following format:

Line1{Image width, Image length, Block size}
 Line 2 {code of block 1}
Line n+1 {code of block n}

- Code of edge block => x, y, i, α, β
- Code of Midrange Block => x, y, α, β
- Code of shade block => avg. of the block

The position of blocks in the original image is as follows:

1	2	3	4
5	6	7	8
.	.	.	.
n-3	n-2	n-1	n

5. Decoding of image from a fractal code

In the decoding algorithm, the output file is read line by line. The process is repeated till all the lines of the output file are processed. This whole process completes one iteration of the decoding algorithm. We found after experimentations that in our images under consideration, after completing 8 – 22 such iterations, the sequence of image generated at each iteration finally converges to the required stable image.

5.1 Iteration-wise results of Lena and Baboon

Information about the coding system, the design specification of code and system performance for the two images are given below (table 3. and table 4.), following are the SNR (Signal to Noise Ratio) values for the Lena and the Baboon images:

No.of Itr.	1	2	3	4	5	6	7	8
SNR	36.1	35.7	35.7	37.0	41.4	41.9	42.4	42.1

Table 1. SNR values for lena

No.of Itr.	1	4	10	14	16	20	24	30
SNR	35.0	36.3	37.9	38.1	38.2	38.3	38.5	38.6

Table 2. SNR values for baboon

5.1.1 Image: Lena

Image type	Resolution	size (in bytes)	Gray levels	Range Blocks	compression ratio
.Pgm	256 x 256	266312	256	4 x 4	1:8

Table 3. Image specification (Lena)



Figure 1. Original image



Figure 2. First eight iteration for the lena image

5.1.2 Image: Baboon

Image type	Resolution	size (in bytes)	Gray levels	Range Blocks	compression ratio
.Pgm	256 x 256	266312	256	4 x 4	1:6

Table 4. Image specification (Baboon)

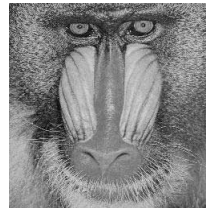


Figure 3. Original image

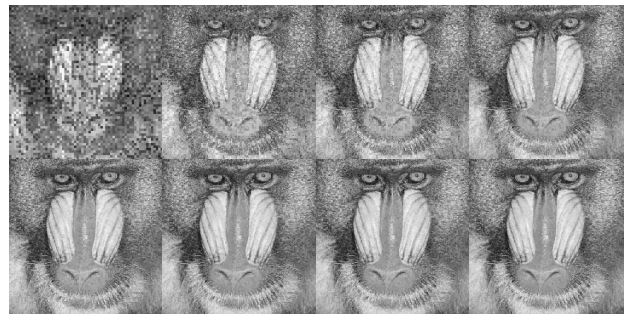


Figure 4. Few iteration for the baboon image (as the baboon image has more edge blocks, it requires larger number of iterations)

Note: The above decompressed images of Lena and baboon show extremely good reproduction of edge blocks and shade blocks, but some blockiness is visible. However, by taking the size of range block as 2 x 2, we find that the blockiness can be completely removed. In this case, the compression is around 50% of the original image and the time of compression increases as evident in table 6* below.

Comparison between the stochastic and non-stochastic algorithms in terms of Image quality






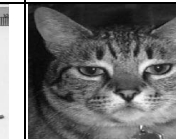











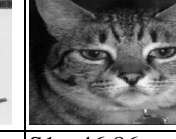
Images	Peppers	Columbia	Face	Madhu	Leaf	Cat
ORIGINAL IMAGE						
D1						
D2						
SNR VALUE	S1= 42.70; ITR = 10	S1= 43.35; ITR=10	S1= 43.73; ITR=10	S1= 43.95; ITR = 10	S1= 41.00; ITR=10	S1= 46.86; ITR = 12
	S2= 43.20; ITR =10	S2 = 43.58; ITR=10	S2= 46.94; ITR = 8	S2= 43.74; ITR =10	S2= 43.07; ITR = 10	S2= 46.64; ITR = 8

Table 5. Comparison between SNR values

***D1**,: Final decoded image without using probabilistic theory ***D2**: Final decoded image using probabilistic theory.
 ***S1**: SNR value of image without using probabilistic theory. ***S2**: SNR value of image using probabilistic theory.
 ***ITR**: No of Iterations used to decode the image.

Comparison between the stochastic and non-stochastic algorithms in terms of compression time and the compression ratio.

IMAGE 266132 bytes	Block Size	Image1	Image2	T1	T2	PR
Lena	4 x 4	39795 bytes	35280 Bytes	3 Min 32 s	1 Min 23 s	62.95%
	2 x 2	136984 bytes	123618 bytes	15 Min 43 s	4 Min 38 s	71.6%
Baboon	4 x 4	56433 bytes	48988 Bytes	8 Min 29 s	2 Min 42 s	70.8%
	2 x 2	209574 bytes	182703 bytes	54 Min 7 s	19 Min 23s	64.4%
Peppers	4 x 4	35374 bytes	31934 Bytes	2 Min 30 s	1 Min 00 s	56.52%
	2 x 2	121001 bytes	111065 Bytes	10 Min 07 s	3 Min 44 s	65.83%
Columbia	4 x 4	39055 Bytes	34639 Bytes	3 Min 26 s	1 Min 21 s	62.88%
	2 x 2	137436 bytes	124160 Bytes	17 Min 17 s	5 Min 04 s	70.65%
Face	4 x 4	32742 bytes	29766 Bytes	1 Min 55 s	0 Min 40 s	74.19%
	2 x 2	114592 bytes	104200 Bytes	8 Min 15 s	2 Min 31 s	71.65%
Madhu	4 x 4	41504 bytes	37001 Bytes	4 Min 16 s	1 Min 18 s	71.63%
	2 x 2	141655 bytes	128759 Bytes	19 Min 19 s	5 Min 39 s	71.91%
Leaf	4 x 4	33327 bytes	30000 Bytes	2 Min 15 s	0 Min 41 s	80.93%
	2 x 2	114911 bytes	105893 Bytes	7 Min 46 s	2 Min 20 s	70.50%
Cat	4 x 4	49767 bytes	43908 Bytes	5 Min 57 s	1 Min 56 s	72.00%
	2 x 2	169147 bytes	152458 Bytes	30 Min 43 s	8 Min 54 s	71.90%

Table 6. Comparison between compression time and the compression ratio.

- ***T1**: Time taken for compression of image without using probabilistic theory.
- ***T2**: Time taken for compression of image using probabilistic theory.
- ***Image1**: compression of image (in bytes) without using probabilistic theory.
- ***Image2**: compression of image (in bytes) using probabilistic theory.
- ***PR**: Percentage reduction in time using the probabilistic theory as compared to the non-probabilistic theory

6. Conclusion

We have presented an algorithm for the Stochastic Image Compression Using Fractals. This algorithm is suitable for all the digital gray scale images. Our algorithm is based on stochastic approach. We are able to reduce time taken for compression of Image by 55% - 80% as compared to the non-stochastic algorithm. Our algorithm also gives high compression ratio. The compression ratio ranges between 60% - 80% which too is greater than the compression ratio achieved using non-stochastic algorithm. The feature of our algorithm is that we are able to achieve almost same or better SNR values for most of the images as compared to non-stochastic algorithm while reducing the compression time and storage space significantly.

7. Acknowledgment

We thank Dr. (Mrs.) Renu Jain for her active interest and valuable suggestion during the preparation of the paper. We are also thankful to University of Akron, Ohio and University of Louisiana at Lafayette for their support.

8. References

- [1]. A. E. Jacquin "Image coding based on fractal theory of iterated contractive image transformations", IEEE Trans, On Image Proc, vol. 1, No. 1, January 1992.
- [2]. B. Ramamurthy and A. Gresho, "Classified vector quantization of images", IEEE Trans. Commun., vol. 34 Nov, 1986.
- [3]. Michael F. Barnsley "Fractals Everywhere" second edition.
- [4]. Michael F. Barnsley, J.H Elton and D.P Hardin, "Recurrent iterated function system, "constructive approximation. Berlin, Germany: Springer-Verlag, 1989, pp.3-31.
- [5]. Harold M. Hastings and George Sugihara," Fractals A User's Guide for the Natural Sciences".
- [6]. B. Mandelbrot, The Fractal Geometry of Nature, San Francisco, CA: Freeman, 1982.