Optimization of fractal image compression with genetic algorithms

A. BEN JMAA, M. BEN JEMAA and Y. BEN JEMAA

***Abstract*— Fractal image compression explores the self-similarity property of a natural image and utilizes the partitioned iterated function system (PIFS) to encode it. The image compression method is time consuming in the encoding process. The time is essentially spent on the search of the similar domain block. In this paper, we present a method that uses Genetic algorithms to speed up computation time in fractal image compression with acceptable image quality and high compression rate. These improvements are obtained by encoding all regions in the image with different size blocks.**

I. INTRODUCTION

Fractal image compression (FIC) was introduced by Bernesly [1] and Jacquin [2]. Since then, many researches have improved the original approach in various ways [3].

The image compression problem puts forward three major requirements : speeding up the compression algorithm, improving image quality after compression/decompression or increasing compression ratio.

The method based on the theory of Local Iterated Function Systems (LIFS) [3] has received a lot of attention in the last ten years. To encode an image according to the self-similarity property, each block must find the most similar domain block in a large domain pool.

In fractal image compression, the original image is partitioned into range blocks and for each range block, a suitable domain block D_j is searched, so it exists a transformation $T : Dom(I) \rightarrow Ranges(I)$; T must guaranty

$$\forall i, \exists j / T(D_j) \simeq R_i$$

A transformation is associated to each R_i, it codes the D_j coordinates and parameters of the transformation. The associated parameters for each R_i are : the isometric flip Rotation $\pi/2$,π ,$3\pi/2$, the horizontal flip, the vertical flip, the transposed of Dom(I), the rotation π of the transposed of Dom(I), the luminance and contrast.

We use a Quadtree method [4] [5]to partition an image that must be compressed.

This process is shown at Fig. 1. and can be summarized with the FIC algorithm.

A. BEN JMAA is with National School of Engineering of Sfax, BP W, 3038 Sfax ahmed.benjmaa@gmail.com

M. BEN JEMAA is with National School of Engineering of Sfax, BP W, 3038 Sfax maher.benjemaa@enis.rnu.tn

Y. BEN JEMAA is with National School of Engineering of Sfax, BP W, 3038 Sfax and Research unit on signals and systems, ENIT, BP37, 1002 Tunis yousra.benjemaa@enis.rnu.tn

Fig. 1. Regions decomposition

FIC Algorithm:
Decompose Image to 16x16 Regions size;
While Exist (Regions not coded)
    If Regions $size > 4$
        Try to Code all available Regions;
    Else
        Code all Regions;
    IEnd
(Region size)=(Region size)/2
Wend

The major problem of this method is time consuming compared with others methods of image compression.

We present in this paper, a new Genetic Algorithm for image compression, that speed up this method when finding a LIFS [6] whose attractor is close to a given image.

The next section is dedicated to the details of our algorithm: the representation of the fitness function, the Genetic operators and some others improvements to the simple Genetic Algorithms.

With experimental results, we prove that our Genetic compression method is a good choice.

II. GENETIC ALGORITHM FOR FRACTAL IMAGE COMPRESSION

There are many algorithms of optimization used for different domains. We have chosen Genetic Algorithm [7] [8] [9] to accelerate our fractal image compression algorithm. We will give details of Genetic characteristics in this section.

A. Chromosome attributes

According to the Regions parameters coding, a chromosome is constituted by N genes where N is the number of image Regions not yet coded.

The gene is composed of three parameters: (X_{Dom},Y_{Dom}) that represent the domain block coordinates and the isometric flip. These three parameters are integers.

- $X_{Dom} \in [0, L]$, L is the image length.
- $X_{Dom} \in [0, W]$, W is the image width.
- $Flip \in [0, 7]$, eight isometric flip.

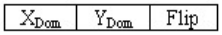

Fig. 2. Gene representation

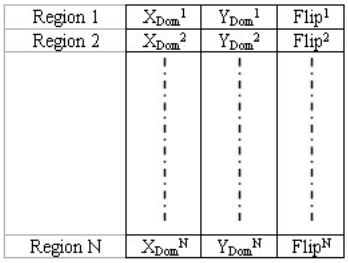

Fig. 3. Chromosome representation

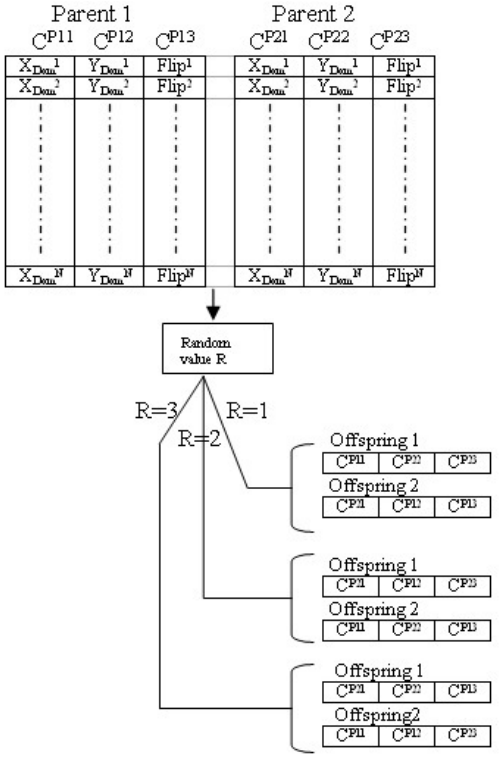

Fig. 4. Crossover operator scheme

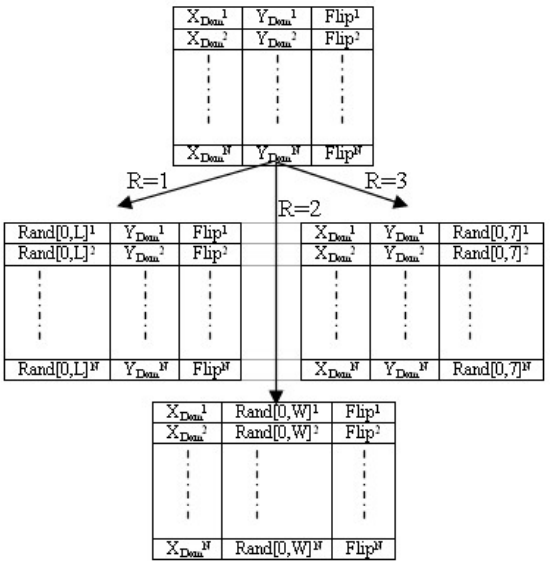

Fig. 5. Mutation operator scheme

B. Genetic operators

The crossover and mutation operators ensure the production of offspring. These genetic operators must be defined according to the chromosome specification.

With these basic components, a Genetic Algorithm works as follows: The first procedure is to generate the first population represented with string codification (Chromosome) that represents possible solution to the problem.

Each individual is evaluated, and according to its fitness, an associated probability to be selected for reproduction is assigned.

- The crossover operator combines two individuals (the parents) of the current generation whose chromosomes have not given selected solution to produce two offspring individuals.

According to our chromosome specification, a new scheme of the crossover operator is proposed :

The offspring coordinates and the isometric flip are selected randomly from the parents (Fig. 4.).

- Mutation operator modifies the chromosome genes randomly according to the mutation probability. Genes (X_{Dom}, X_{Dom}, Flip) are changed with random generated values respectively in [0,L], [0,W] and [0,7] intervals (Fig. 5.).

C. Fitness measure

The fitness function assigns to each individual in the population a numeric value that determines its quality as a potential solution. The fitness denotes the individual ability to survive and to produce offspring.

In our case, the fitness is the number of regions that can be coded with *RMS* error less than a fixed value. The *RMS* is the distance between the region and the domain block determined by its coordinates (X_{Dom}, X_{Dom}) and

transformed with corresponding luminance s and contrast value o.

The parameters *RMS*, s and o are given in the following (D_i is domain elements and R_j denotes the range elements):

$$RMS = \|s_k * D_i - R_j\|$$

Where is the two norm function

$$s = [0.45, 0.6, 0.8, 0.97]$$

and

$$o = (\sum_{i=1}^{n}\sum_{j=1}^{n} R_{ij})/n^2$$

with n is the region size.

D. Genetic compression algorithm

Genetic Algorithms have been used previously to find solutions to the minimization problems related to the fractal inverse problem [8].

In this section, we describe the Genetic Algorithm that we use to speed up compression algorithm.

This algorithm is used for all decomposition schemes. In spite of the range block size and position; the domain block is always the double size of the range one.

The Algorithm:

(Input I: NxN gray scale image [Image would be square]
Output W: Coded IFS);
$(RegionSize) = 16$;
$(FixedError) = X * (RegionSize)/4$;
Decompose the input image into (Region Size) blocks;
While Exist (Regions not coded)
- Scale the Domain Blocks;
- Generate a random population of chromosomes;
While Exist (Regions not coded) and
(Last generation not reached)
- Compute fitness for all regions;
- When optimal domain block found write obtained transformation parameters to the output W;
- Generate new population {Apply Crossover and Mutation operators};

Wend
$(RegionSize) = (RegionSize)/2$;
$(FixedError) = X * (RegionSize)/4$;
If Regions $size > 4$
Decompose the rest region not coded into (Range Size) blocks;
Else
$(FixedError) = (FixedError) + X$;
Code all rest Regions;
IEnd
Wend

III. EXPERIMENTAL RESULTS

The proposed algorithm has been evaluated on various images with different sizes. The following results are obtained for 256x256 Lenna image.

We present the obtained results for different configurations of error limit, number of iterations and population size.

All these experiment results are obtained with algorithm developed with Matlab. Fig. 6. and 7. show the decompressed images obtained using our algorithm with error equal to 10, iterations number equal to 10 and population size equal to 12.

Fig. 6. Decompressed Lenna image

Fig. 7. Decompressed House image

In the next three tables, we evaluate the computational time, quality and compression rate with different error, iteration number and population size.

TABLE I
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT ITERATION NUMBER

Configuration					
Pop	Err	Iter	Times(s)	PSNR(db)	Rate(%)
12	10	5	255	26,25	78,01
12	10	10	416	26,94	78,62
12	10	15	571	27,07	78,74
12	10	20	729	27,31	79,19
12	10	25	871	27,93	79,39
12	10	50	1580	28,23	79,83

When increasing the iteration number, the search space increased consequently, we obtain higher compression rate and better quality.

TABLE II
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT POPULATION SIZE

Configuration					
Pop	Err	Iter	Times(s)	PSNR(db)	Rate(%)
6	10	10	275	26,19	77,87
12	10	10	416	26,94	78,62
18	10	10	581	27,27	78,93
24	10	10	720	27,57	79,00

To obtain a best quality, we must increase the population size in order to increase the search space.

TABLE III
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT ERROR

a. Our algorithm

Configuration					
Pop	Err	Iter	Times(s)	PSNR(db)	Rate(%)
12	10	10	416	26,94	78,62
12	20	10	260	26,53	83,24
12	30	10	173	25,75	87,04
12	40	10	119	24,89	89,13
12	50	10	81	23,9	91,18
12	60	10	49	22,91	93,39
12	70	10	33	21,96	95,1
12	80	10	19	21,28	96,14
12	90	10	12	20,55	96,98

b. Standard algorithm

Configuration					
Pop	Err	Iter	Times(s)	PSNR(db)	Rate(%)
-	10	-	10453	30,47	79,65
-	20	-	5935	29,34	85,55
-	30	-	3514	27,55	88,97
-	40	-	2067	26,31	91,31
-	50	-	1088	24,83	93,58
-	60	-	559	23,44	95,29
-	70	-	263	22,44	96,77
-	80	-	120	21,57	97,55
-	90	-	54	20,94	98,12

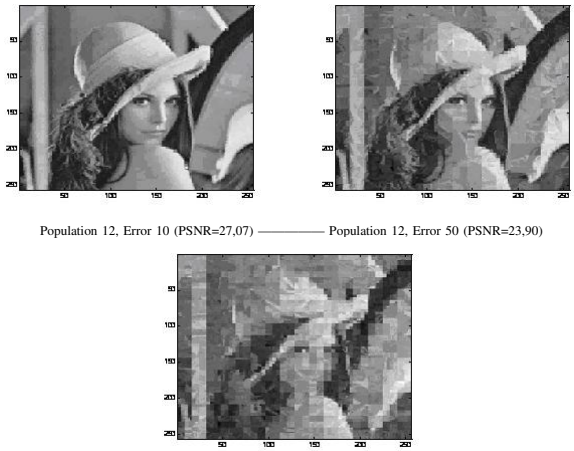

Fig. 8. Error influence on the quality

It's clear that Genetic Algorithm greatly reduce compression time without significant loss of image quality.

It is important to note that we can reach the same quality as the standard approach with less computational time, this is shown in (Fig. 9).

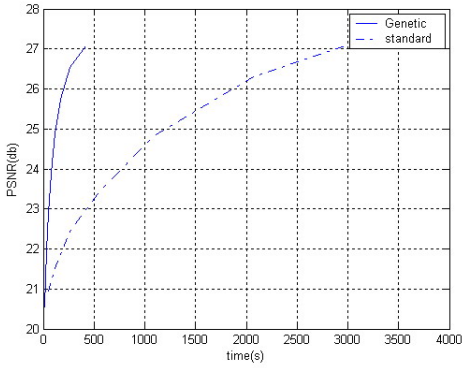

Fig. 9. Image quality versus execution time (Lenna image)

Finally, with our approach we can choose the best configuration (population size, iteration number, error) for our need of quality and time execution.

IV. CONCLUSION

In this paper, we have designed a new Genetic Algorithm to optimize the fractal image compression.

The Genetic Algorithm improves performances of the basic technique according to the results seen in previous section.

In particular, it provides higher compression speed and higher compression ratio while the quality of the decompression image is acceptable compared to the standard approach known as the best method that gives the best quality.

REFERENCES

[1] M.Barnsley and L. Hurt, "Fractal Image Compression", Peters, Wellesley, 1993.

[2] A. E. Jaquin, "Image coding based on a fractal theory of iterated contractive image transformations", IEEE Trans. on image Processing. 1, PP, 18-30, 1992.

[3] B.E. Wohlberg and G. de Jager, "A review of the fractal image coding literature", IEEE Trans. on image Processing. 8(12), PP, 1716-1729, 1999.

[4] Y. Fisher, " Fractal Image Compression with Quadtrees ", 1995.

[5] Ghim-Hwee Ong,Chorng-Meng Chew and Yi Cao, "A Simple Partitioning Approach to Fractal Image Compression", SAC 2001, Las Vegas, NV 2001 ACM.

[6] Seropian, Audrey and Grimaldi, Michel and Vincent, Nicole, " Diffrenciation entre Alphabets dans des Textes Manuscrits ". Communication, Confrence Internationale Francophone sur l'Ecrit et le Document (CIFED 04). 21 June 2004. Working Paper.

[7] D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley Publishing Company, 1989.

[8] R. Shonkwill, F.Mendivil and A.Deliu, "Genetic Algorithms for the 1-D Fractal Inverse Problem", Proceedings of the Fourth International Conference on Genetic Algorithms, San Diego, 1991.

[9] Ming-Sheng Wu, Wei-Chih Teng, Jyh-Horng Jeng and Jer-Guang Hsieh, "Spatial correlation genetic algorithm for fractal image compression", ScienceDirect 2005.

[10] E.R Vrscay, "Moment and Collage Methods of the Inverse Problem of Fractal Construction with Iterated Function Systems", Proceedings of 1st IFIP Conference on Fractals, Lisbon , PP 6-8, June 1990.