Modified Fractal Image Compression Using Genetic Algorithms

P.V.Kasambe, Mohsin Patel

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 in the search of the similar domain block. Until now, a lot of work has been done to reduce the time spent by encoding process for fractal gray scale image compression, although no paper in the literature addresses a method to reduce the time spent for fractal color image compression. In this paper, we present a method that uses Genetic Algorithms to speed up computation time in fractal image compression with acceptable image quality for gray scale and color images.**

***Index Terms*-- Genetic Algorithms, PSNR, Fractal Image Compression.**

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.

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

$$\forall i, \exists j / T(Dj) \approx Ri$$

A transformation is associated to each Ri, it codes the Dj coordinates and parameters of the transformation. The associated parameters for each Ri are: the isometric flip Rotation $\pi/2, \pi, 3\pi/2$, 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 have selected a traditional way of image partitioning which is using square blocks of range and domain blocks for both gray scale and color image compression.

The coordinate systems used for color image are RGB, YIQ and YUV. To encode a color image the main idea is to divide the image into its three different layers or components (RGB, YIQ and YUV). It is then possible to compress each of these layers separately, in other words, handle each of the layers as an independent image [4].

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, which speed up this method.

The remainder of the paper is organized as follows: Basic principles and features of Genetic Algorithms are presented in Section 2.Genetic algorithms for fractal image compression is described in Section 3. Experimental Results are given in section 4. Finally, the paper is concluded in Section 5.

II. Basic principles and features of Genetic Algorithms

The GA's are adaptive search processes based on the notion of selection mechanism of natural genetic system [5]. GA's help to find the global near optimal solution without getting stuck at local optima as they deal with multiple points (spread all over the search space) simultaneously. To solve the optimization problem, the GA starts with the structural representation of a parameter set. The parameter set is coded as a string of finite length and the string is called chromosome. The chromosomes may be encoded as Binary or Continuous values.

1) Fitness Function: Usually, a function, called *fitness function* is defined on the set of chromosomes (strings). The problem here is to find the string (chromosome) that provides optimal fitness value among all strings (chromosomes). In this work, we are dealing with a minimization problem.

2) Initial Population: Out of all possible strings, initially a few strings (say S number of strings) are selected randomly and this set of strings is called initial population [6]. For simulation, we have taken S to be an even number.

Three basic genetic operators, i) selection, ii) crossover, and iii) mutation, are exploited in GA's. The genetic operators are applied on the initial population to give rise to a new population of same size (S). The operators are again applied

P. V. Kasambe is with Department of Electronics and Telecommunication Engineering at Sardar Patel Institute of Technology, Mumbai 400058 INDIA. (e-mail: p_vasantrao@rediffmail.com).

Mohsin Patel is with Department of Electronics Engineering at Sardar Patel Institute of Technology, Mumbai 400058 INDIA. (e-mail:mohsin36@gmail.com).

on this new population to give rise to another population. The process of creation of a new population from the existing one is called iteration. In this article the process is executed for a fixed number of iterations.

3) Selection: In this operation, a mating pool of strings of the current population is generated by using the fitness values of strings in the population. The probability of selection of a string in the population to the mating pool is inversely proportional to its fitness value. (Note that the present optimization problem is a minimization problem.) This scheme is known as proportional selection scheme. This scheme provides more representatives of the string having optimal fitness value among all the strings in the present population. The size or the number of strings in a mating pool is same as that of the initial population.

4) Crossover and Mutation: The crossover and mutation operators ensure the production of offspring. These genetic operators must be defined according to the chromosome specification. Mutation is a second way a GA explores a cost surface. If care is not taken, the GA can converge too quickly into one region of the search space.

With these basic components, a Genetic Algorithm works as follows: The first step 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.

These steps are repeated till the convergence criterion is not reached.

III. Genetic Algorithms for Fractal Image Compression

There are many algorithms of optimization used for different domains. We have chosen Genetic Algorithm [5, 6] to accelerate our fractal image compression algorithm. We will give details of Genetic characteristics in this section.

A. CHROMOSOME ATTRIBUTES

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 as shown in fig. 1.

- $X_{Dom} \in [0, L]$, L is the image length.
- $Y_{Dom} \in [0, W]$, W is the image width.
- $Flip \in [0, 8]$, Eight isometric flips.

X_{DOM}	Y_{DOM}	Flip

Fig. 1. Gene representation

According to the Regions parameters coding, a chromosome is constituted by N genes where N is the number of image Regions not yet coded. Its representation is as shown in fig. 2.

Region 1	X_{DOM}^1	X_{DOM}^1	$Flip^1$
Region 2	X_{DOM}^2	Y_{DOM}^N	$Flip^2$
!			
!			
!			
!			
!			
Region N	X_{DOM}^N	X_{DOM}^N	$Flip^N$

Fig. 2. Chromosome representation

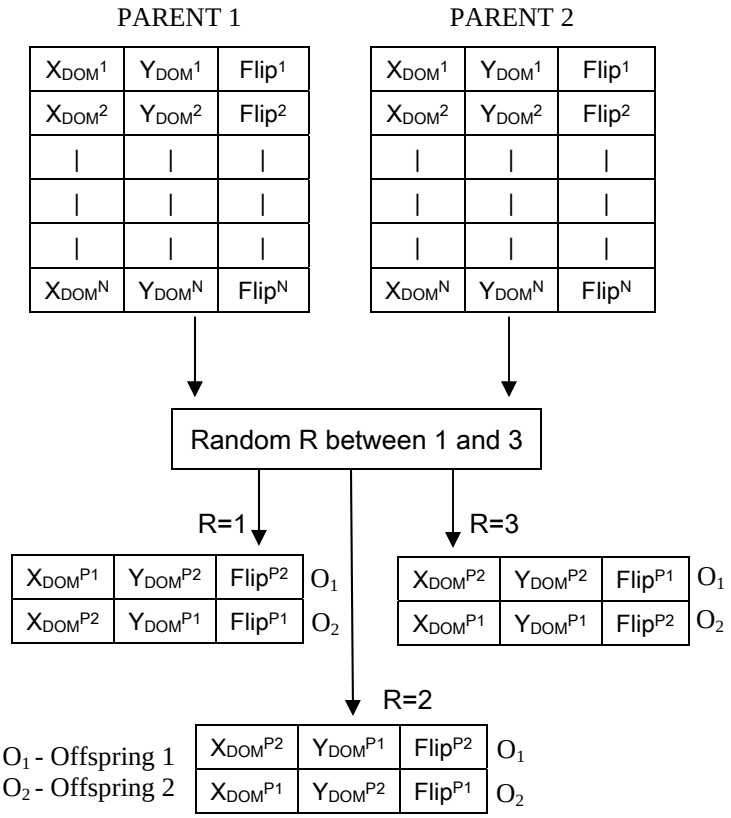


Fig. 3. Crossover operator scheme

B. GENETIC OPERATORS

According to our chromosome specification, a new scheme of the crossover operator is proposed [7] in which the offspring coordinates and the isometric flip are selected randomly from the parents as shown in fig. 3.

Mutation operator modifies the chromosome genes randomly according to the mutation probability. Genes (X_{Dom}, Y_{Dom}, Flip) are changed with random generated values respectively in [0,L], [0,W] and [0,7] intervals as shown in figure 4.

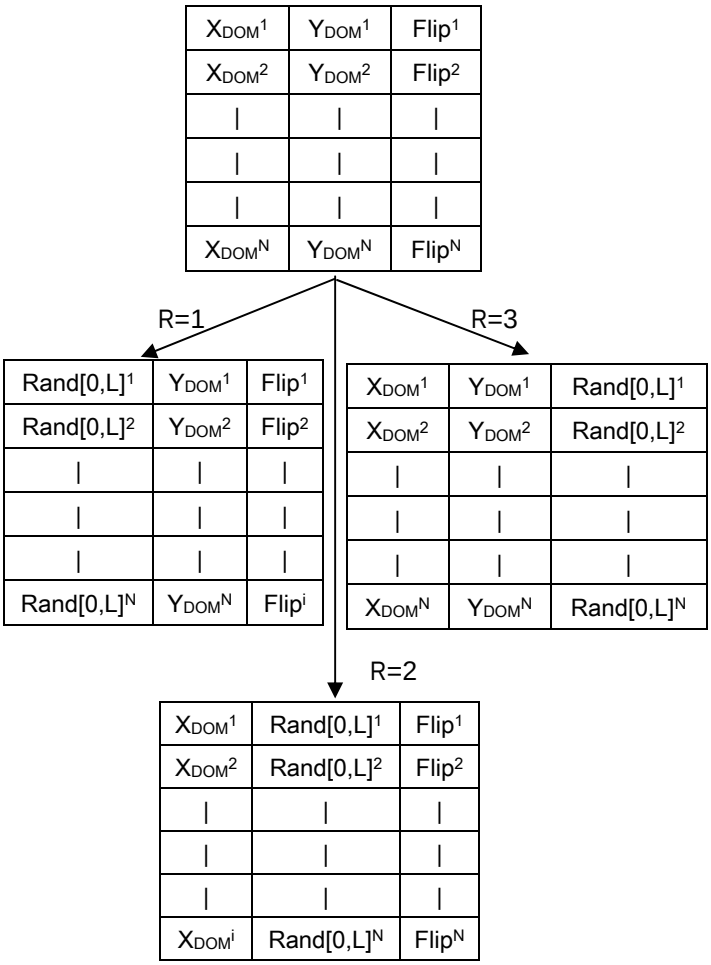


Fig. 4. Mutation operator scheme

C. FITNESS MEASURE

The fitness function assigns to each individual in the population a numeric value that determines its quality as a potential solution [7, 8]. 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} , Y_{Dom}) and transformed with corresponding luminance's' and contrast value 'o'. The parameters RMS, s and o are given in the following equations 1, 2 and 3 respectively (Di is a domain element and Rj denotes the range elements):

$$\text{RMS} = \left\| s_k * Di - Rj \right\| \quad \text{---------- (1)}$$

$$s = \frac{\sum_{i=1}^{n}\sum_{i=1}^{n}\left(Ri * Di\right)}{\sum_{i=1}^{n}\sum_{i=1}^{n} Di^2} \quad \text{----------(2)}$$

$$o = \left(\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} Rij}{n^2} \right) \quad \text{---------- (3)}$$

Where 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 [6].

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

The Algorithm:

(Input I: NxN image [Image would be square]
Output Coefficient;
Select Region Size;
Select Error Limit;
Initialize GA parameters;
Generate a random population of chromosomes;
Decompose the input image into (Region Size) blocks;
While Loop (Number of iterations reached OR Error exceeds Error limit)
  While Loop (Regions not coded)
    -Select Region Blocks sequentially
    While Loop (Last generation not reached)
      - Select and Scale the Domain Blocks;
      - Compute fitness for all regions;
      - When optimal domain block found write obtained transformation parameters to the Output Coefficient;
    Wend
  Wend
  -Generate new population {Apply Crossover and Mutation operators};
Wend

IV. 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 on a PIV-INTEL 2.4 GHz with 256MB of RAM size.

TABLE I
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT ITERATION NUMBER FOR GRAY SCALEIMAGE

Configuration				
Pop	Err	Iter	Times(s)	PSNR(dB)
12	8	2	15.52	30.03
12	8	8	59.81	31.34
12	8	14	105.66	31.83
12	8	20	149.61	32.15

TABLE II
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT ITERATION NUMBER FOR COLOR SCALE IMAGE

Configuration (Population Size = 12 and Error Limit =8)					
Coordinate system	Iter	Times(s)	PSNR(dB)		
			R	G	B
RGB	2	46.52	39.63	47.76	41.66
	8	188.98	40.77	42.63	42.40
	14	332.61	41.20	43.12	42.79
	20	451.75	41.50	43.34	42.84
YIQ	2	48.30	39.88	41.73	41.73
	8	195.53	41.02	42.73	41.82
	14	332.75	41.52	43.11	42.19
	20	472.33	41.76	43.30	42.36
YUV	2	49.11	39.82	41.64	41.22
	8	193.81	40.98	42.64	42.04
	14	334.78	41.54	43.07	42.42
	20	476.59	41.69	43.21	42.58

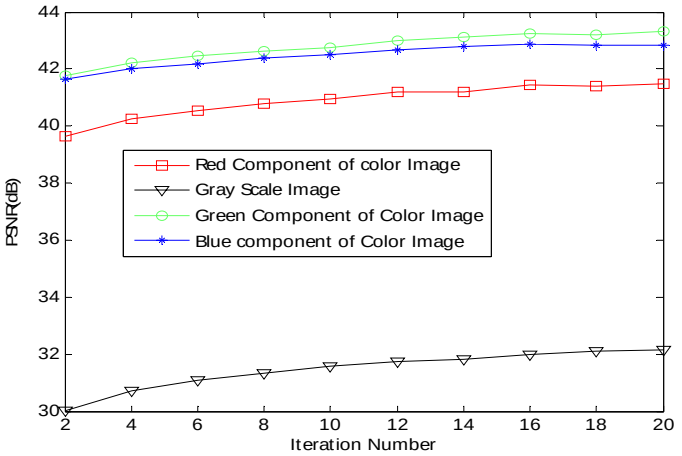


Fig. 4.:LENA Image quality variation according to Iteration Number for Population Size=12 and Error Limit=8 (To plot the variation RGB Co-ordinate system is used for Color Image).

It can be observed from fig. 4 and Table I and II, increasing the iteration number, the search space increases consequently, we obtain better image quality.

TABLE III
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT POPULATION SIZE FOR GRAY SCALEIMAGE

Configuration				
Pop	Err	Iter	Times(s)	PSNR(dB)
6	8	10	19.77	30.20
12	8	10	41.00	30.90
24	8	10	73.47	31.65
36	8	10	109.88	31.91

TABLE IV
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT POPULATION SIZE FOR COLOR SCALE IMAGE

Configuration (Iteration =10 and Error Limit =8)					
Coordinate system	Pop	Times(s)	PSNR(dB)		
			R	G	B
RGB	6	59.45	39.78	41.73	41.73
	12	110.36	40.38	42.25	42.15
	24	225.50	41.02	42.87	42.52
	36	344.52	42.23	43.22	42.81
YIQ	6	58.41	40.03	41.85	41.24
	12	114.86	40.66	42.40	41.65
	24	232.23	41.33	42.94	42.06
	36	355.80	41.59	43.17	42.27
YUV	6	57.89	39.96	41.10	41.27
	12	111.27	40.63	42.28	41.78
	24	223.89	41.22	42.80	42.16
	36	334.28	41.50	43.10	42.42

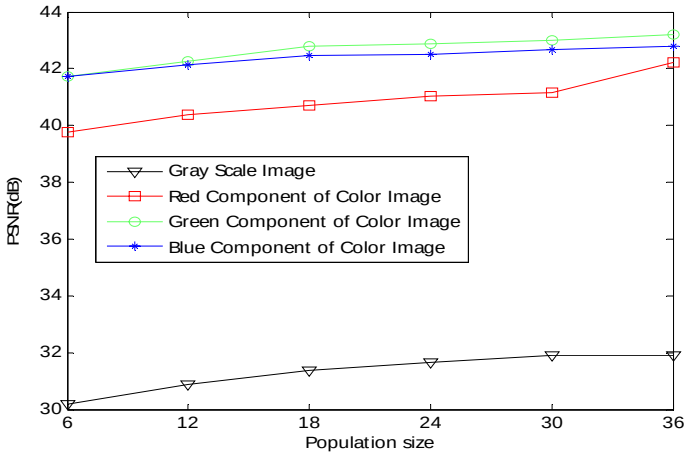


Fig. 5: LENA Image quality variation according to population size for Iteration Number=10 and Error Limit=8 (To plot the variation RGB Co-ordinate system is used for Color Image).

It can be observed from fig. 5 and Table III and IV that in order to increase the search space to obtain the best quality, we must increase the population size.

TABLE V
LENNA IMAGE RESULTS OBTAINED FOR DIFFERENT ERROR

Our Algorithm for Gray scale Image

Configuration				
Pop	Err	Iter	Times(s)	PSNR(dB)
12	6	10	49.82	31.27
12	7	10	22.00	30.48
12	8	10	14.48	30.15
12	10	10	7.40	29.16

Standard Algorithm for Gray scale Image

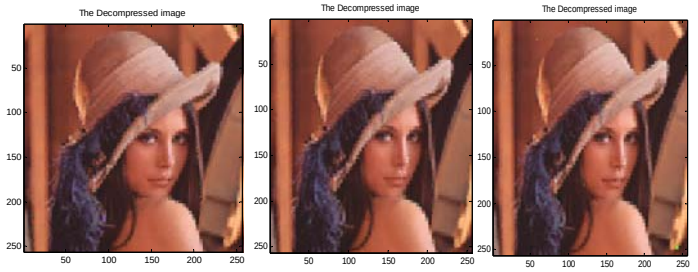
Configuration				
Pop	Err	Iter	Times(s)	PSNR(dB)
12	6	10	93.63	31.95
12	7	10	76.44	31.56
12	8	10	60.13	30.99
12	10	10	39.50	30.21

Our Algorithm for Color Image

Configuration (Iteration =10 and Population Size = 12)					
Coordinate system	Err	Times(s)	PSNR(dB)		
			R	G	B
RGB	6	95.55	40.79	41.73	41.61
	7	41.44	39.95	41.11	41.10
	8	24.67	39.92	41.00	40.90
	10	24.29	38.95	40.11	40.02
YIQ	6	35.78	39.37	41.49	40.06
	7	25.68	38.53	40.68	39.84
	8	25.14	38.76	40.77	40.06
	10	24.80	38.81	40.08	40.02
YUV	6	32.75	38.88	41.00	39.61
	7	25.38	38.42	40.37	39.80
	8	25.06	38.68	40.67	39.94
	10	24.31	38.36	40.38	40.18

Standard Algorithm for Color Image

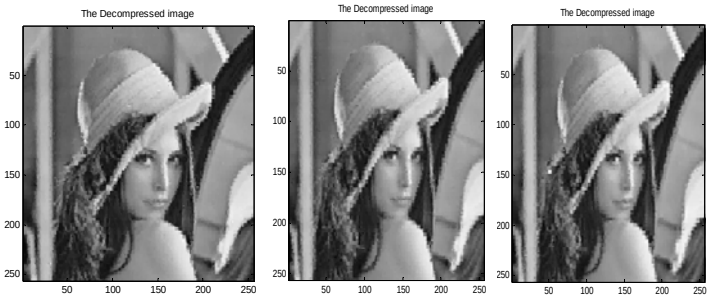
Configuration (Iteration =10 and Population Size = 12)					
Coordinate system	Err	Times(s)	PSNR(dB)		
			R	G	B
RGB	6	268.08	41.75	42.27	42.37
	7	171.97	40.90	41.61	41.89
	8	134.28	40.12	41.32	41.64
	10	87.31	39.40	41.04	41.39
YIQ	6	110.02	40.27	42.50	41.21
	7	83.08	39.57	41.79	41.32
	8	69.69	39.14	41.33	41.02
	10	61.88	38.86	41.06	40.92
YUV	6	108.45	40.30	42.44	41.80
	7	83.07	39.61	41.74	41.39
	8	67.63	39.17	41.28	41.11
	10	61.27	38.89	41.02	41.01



(a) PSNR (RGB) 40.916, 41.777, 41.792.

(b) PSNR (RGB) .814,41.687, 41.609

(c) PSNR (RGB) 39.7542, 41.667, 41.173



(a) PSNR = 31.335

(b) PSNR = 30.148

(c) PSNR) = 30.072

Fig. 6: LENA Decompressed Images showing the image quality variation according to RMS Error limits (a) Error Limit=6 (b) Error Limit=8 (c) Error Limit=10 for Population size=12, Number of Iterations=10.

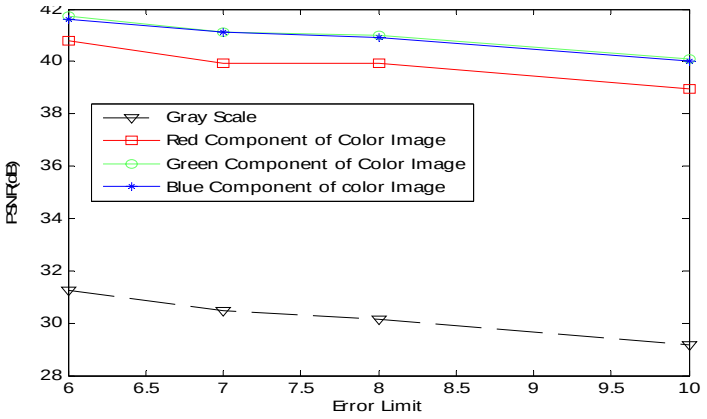
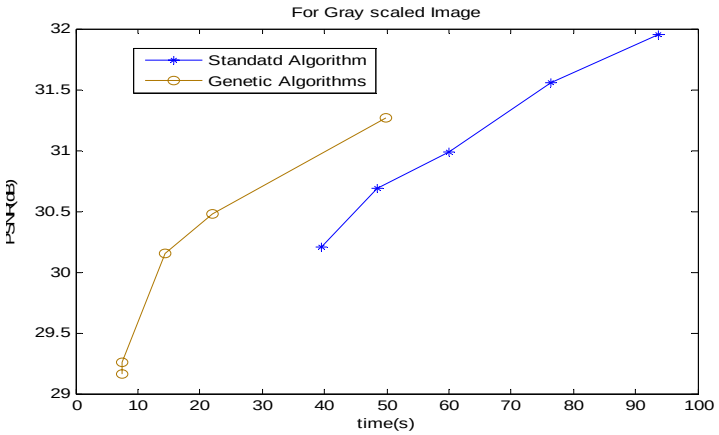
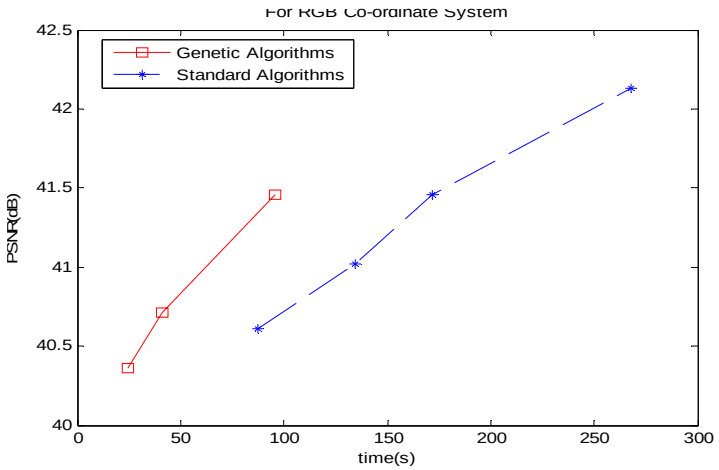


Fig. 7: LENA Image quality versus Error Limit for Population size=12,Number of Iterations=10

As can be seen from fig. 6 and fig. 7 and Table V, increasing the error limit reduces the image quality.



(a)



(b)

Fig. 8: LENA Image quality versus execution time.(a) Gray scale image (b) Color Image

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. 8 and Table V.

V. Conclusions

In this paper, we have designed a new Genetic Algorithm to optimize the fractal image compression for gray scale and color image.

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

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

VI. Acknowledgment

The authors gratefully acknowledge the facilities provided in DSP Laboratory of Electronics Engineering Department, Sardar Patel Institute of Technology, Mumbai.

VII. References

Periodicals:

[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] Fisher, Y (ed) 1995, *Fractal image compression-theory and application*, Springer-Verlag, New York.

[5] D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley, 1989.

[6] 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.

[7] Faraoun Kamel Mohamed and Boukelif Aoued "Speeding up Fractal Image Compression by Genetic Algorithms", Journal of Multidimensional Systems and Signal Processing, volume 16, April 2005, 217-238.

[8] Suman K. Mitra, C. A. Murthy, and Malay K. Kund, "Technique for Fractal Image Compression Using Genetic Algorithm", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 7, NO. 4, APRIL 1998.

VIII. Biographies

P. V. Kasambe (M'2007) was born in Yavatmal in India on Feb 28, 1975. He graduated from the College of Engineering Badnera, and completed his post graduation in the field of Electronics Engineering from S.P.C.E. (UA), Mumbai-India. His employment experience includes eight years as an educationalist. His special fields of interest include Image Processing, Process Control instrumentation and optimization algorithms. He is member of IEEE, ISTE, ISA and ISNT.

Mohsin Patel (M'2007) was born in Mumbai in India on 1986. He is graduate student of Sardar Patel College of Engineering, Mumbai. He won the 2nd prize in Extreme Machines at Technovanza 2005, a National Level Annual Technical festival, organized by VJTI COE, Mumbai, and 1st prize in Junkyard Wars during SPACE 2005 organized by Sardar Patel College of Engineering, Mumbai COE. His area of interest includes Image Processing and Digital Signal Processing.