ISSN: 0976-9102(ONLINE)
DOI: 10.21917/ijivp.2014.0113

ITERATION FREE FRACTAL COMPRESSION USING GENETIC ALGORITHM FOR STILL COLOUR IMAGES

A.R. Nadira Banu Kamal[1] and P. Priyanga[2]
Department of Computer Science, Thassim Beevi Abdul Khader College for Women, India
E-mail: [1]nadirakamal@gmail.com, [2]p.priyanga19@gmail.com

Abstract
The storage requirements for images can be excessive, if true color and a high-perceived image quality are desired. An RGB image may be viewed as a stack of three gray-scale images that when fed into the red, green and blue inputs of a color monitor, produce a color image on the screen. The abnormal size of many images leads to long, costly, transmission times. Hence, an iteration free fractal algorithm is proposed in this research paper to design an efficient search of the domain pools for colour image compression using Genetic Algorithm (GA). The proposed methodology reduces the coding process time and intensive computation tasks. Parameters such as image quality, compression ratio and coding time are analyzed. It is observed that the proposed method achieves excellent performance in image quality with reduction in storage space.

Keywords:
Fractal Image Compression, Genetic Algorithm, Synthetic Code Book, Iteration Free, Colour Images

1. INTRODUCTION

With the recent rapid growth of multimedia applications and digital transmission, image compression techniques have become a very important subject. The two-fold dimensions of image compression are the efficiency in transmission and storage capacity of digital data. Fractal Image Compression is described as a self-vector Quantization, where the image blocks are encoded applying a simple transformation to one of the blocks previously encoded. Transformations frequently used are combinations of scaling, reflection and rotation of another block. 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 Graphics Interchange Format (GIF). 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. Other techniques for image compression include the use of fractals and wavelets. These methods have not gained widespread acceptance for use on the Internet. However, both methods are being explored because they offer higher compression ratios than the JPEG or GIF methods.

The fractal image compression problem puts forward three major requirements: speeding up the compression algorithm, improving image quality and increasing compression ratio [3, 6]. An iteration-free fractal image coding using the technique Genetic Algorithm is proposed for lossy compression in this research work to improve decoded colour image quality, compression ratio and to reduce the coding time. Usage of synthetic codebook for encoding using Fractal does not require iteration at decoding and the coding error is determined immediately at the encoder [7]. Hence there is a reduction in decoding time. Very few literatures are available on iteration-free fractal coding. Genetic Algorithm tries to emulate biological evolutionary processes to solve optimization problems. Instead of searching one point at a time, they use multiple search points. Thus, they claim significant advantage of large reduction in search space and time. Optimal fractal coding is an NP-hard combinatorial optimization problem. So this technique is applied in this research work for fractal image compression.

An iteration-free fractal coding scheme with vector quantization was proposed in [1]. An improvement for reduction in time is suggested in the proposed method using GA, as GA can be applied to virtually any problem that has a large search space. It attempts to find near-optimal solutions without going through an exhaustive search mechanism [2, 8]. Hence it is proposed to use GA for finding the better match of the domain block to the range block without going through exhaustive search in the iteration-free fractal coding instead of the existing method using VQ.

In this research paper section 2 and 3 describes the architecture and the algorithm of the proposed method. Section 4 explains how it is implemented followed by results and discussions in section 5. Conclusion and its applications are given in section 6.

2. ARCHITECTURE OF THE PROPOSED TECHNIQUE

In the proposed method a synthetic codebook is created as the domain pool using the mean image, whose pixel values are the block means of all the range blocks. This code book is used as the domain pool for genetic algorithm technique. The architecture of the proposed method is described in Fig.1.

The sender sends the colour image for compression. In the preprocessing stage, the input M × N image under coding is divided into non-overlapping square blocks of B × B pixels called the range blocks. Then the mean and variance of each range blocks are determined. For each range block the red, green and blue component's mean and variance are computed and then concatenated. After the mean of all the range blocks are obtained, a mean image of size M/B × N/B with each pixel corresponding to the block mean is generated.

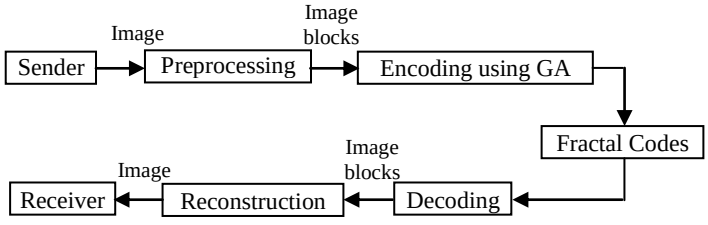


Fig.1. Architecture of the Proposed Iteration-Free Fractal Image Coding Method

The mean image must be larger than the size of the range block i.e. M/B × N/B > B × B. The maximum size of B is limited to 8 in order to produce a good quality of the decoded image. The higher the resolution of the input image (M × N) more blocks can be generated for the domain pool which helps to find a good mapping between the domain and range blocks. The initial domain pool with blocks of the same size as the range is generated using the mean image. In the encoder if the variance of the range block is smaller than the threshold value *E*, the range block is coded by the mean, or else the range block will be coded by the contractive affine transformation [4, 9].

$$V\{R\} = \frac{1}{B^2} \sum_{0 \le i,j < B} \left(r_{i,j} - \mu_R\right)^2 \quad (1)$$

The aim of the proposed scheme is to find the domain block for each image range block and the transformation parameters that minimize the distortion between the image block and the transformed domain block in a minimized time. This process of finding the best domain block makes use of the techniques GA.

In the decoder, shown in Fig.2 the mean information of each range block is extracted from the fractal codes. Using this information the mean image is constructed. This mean image is partitioned into blocks of the same size as the input image. This forms the domain pool for GA search methods. The decompressed image is constructed block by block by applying the transformation parameters to the selected domain block from the domain pool as per the code.

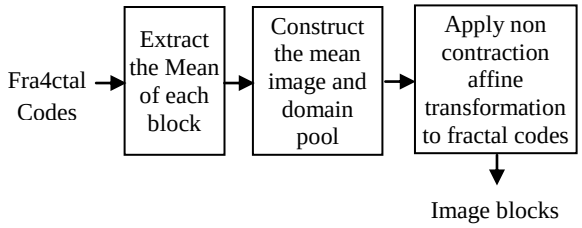


Fig.2. Proposed Decoder

2.1 ENCODER FOR THE PROPOSED ITERATION-FREE FRACTAL IMAGE CODING USING GA

The random numbers required for GA is globally generated using Knuth algorithm for random numbers. The sizes of the chromosomes and population, the number of generations and the probability for crossover and mutation are finalized only by trial that results in good compression ratio and PSNR. Then GA method is applied to search the best domain block with the required transformation that matches the range block.

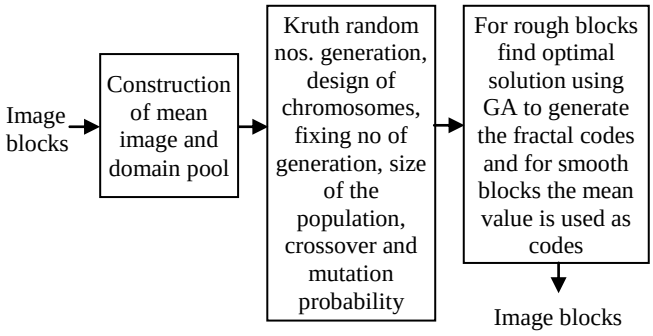


Fig.3. Proposed Encoder Using GA

The architecture of the encoder using GA method is described in Fig.3. The number of possible domain blocks to be searched is $(M/B^2) \times (N/B^2)$, the number of isometry transformations to be searched for each domain block is eight and the contrast scaling parameter is four for each RGB color components. Thus, the space to be searched consists of N1 elements for each RGB color components. $N1 = 8 \times 4 \times (M/B^2) \times (N/B^2)$. Let the space to be searched for each RGB color components be represented by P where,

$$p = \{1, 2, ..., 8\} \times \{1, 2, 3, 4\} \times \{1, 2, ..., (M/B^2) \times (N/B^2)\} \quad (2)$$

Binary strings are introduced to represent the elements of *p* [5, 10]. The set of 2^n binary strings, each of length n for each RGB color components, are constructed in such a way that the set exhausts the whole parametric space. The value for n depends on the values of M, N and B. The fitness value of a string between the given range block and the obtained range block is taken to be the MSE given in Eq.(3). Let S be the population size and T be the maximum number of iterations for the GA. Note that the total number of strings searched up to T iterations is S × T. Hence, N1/S × T provides the search space reduction ratio for each rough type range block.

$$MSE\left(R, \hat{R}\right) = \frac{1}{B^2} \sum_{0,i,j \le B} \left(r_{i,j} - \hat{r}_{i,j}\right)^2 \quad (3)$$

3. ALGORITHMS FOR ENCODING AND DECODING OF THE PROPOSED ITERATION-FREE FRACTAL IMAGE CODING

3.1 PROPOSED ENCODER

The encoding procedure can be summarized in the following steps.

Step 1: The mean μ_R and variance V_R of each range block R (*i*, *j*) is determined. A mean image of size M/B × N/B with each pixel corresponding to the block mean is generated. The mean image must be larger than the size of the range block i.e. M/B × N/B > B × B.

Step 2: The mean image is divided into blocks of the same size as the range block (B × B pixels) to form the domain pool.

Step 3: If the variance of the range block is smaller than the threshold value *E*, then the range block (smooth block) is coded by the mean, otherwise, the range block (rough block) will be coded by the fractal code f (*i*, α, μ_R, P_D) using GA technique. Here *i* represent the isometry transformations, α contrast scaling, μ_R the mean value of the range block and P_D the domain block number in the domain pool.

$$\hat{R}(i,j) \begin{cases} = \mu_R & \text{if } V_R \le E \\ = f(i, \alpha, \mu_R, P_D) & \text{if } V_R > E \end{cases} \quad (4)$$

Thus the input from the sender is the image and the output is the fractal codes.

3.2 ALGORITHM OF THE ENCODER FOR THE PROPOSED ITERATION-FREE FRACTAL IMAGE CODING USING GA

The algorithm of the proposed iteration-free fractal image coding using GA is given as follows:

Step 1: Generate Knuth random numbers globally

Step 2: Partition the given image into range blocks X of size B × B and find the mean and variance of each X.

Step 3: Plot the mean image using the mean of X as the pixel value and partition this mean image into blocks of size B x B to form the domain pool. Decide on the length of the chromosomes, crossover probability, mutation probability, population size and number of iterations. For each range block X:

Step 4: If variance(X) < E assign 0 to label and μ_x to code. Else assign 1 to label. Generate the initial population. Calculate the RMS value of X and each population. Store the values. Compute the next iteration using the roulette wheel selection procedure and apply crossover and mutation probability. Calculate the RMS value of X and each population for each iteration. Use elitism to restore the previous best solution. Return the Fractal code of the minimum RMS values after the last iteration.

3.3 ALGORITHM OF THE DECODER

The decoding process is as follows:

Step 1: Extract the mean information of each range block from the fractal codes and construct the mean image.

Step 2: The domain pool is obtained by partitioning the mean image using the same size as the range block for GA.

Step 3: For smooth blocks, the decompressed image blocks are obtained by the mean value and for rough blocks apply contractive affine transformation using the fractal codes.

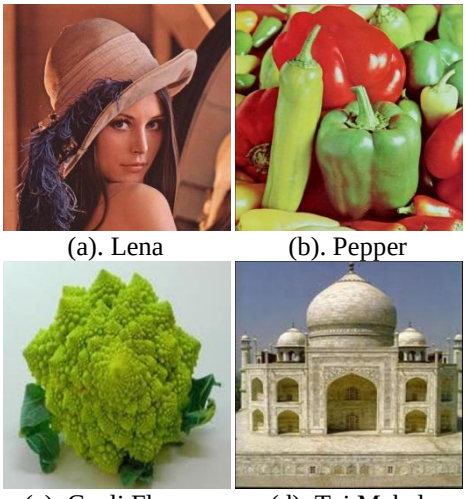
The outputs of the decoder are image blocks that are combined to form the decoded image at the receiver end. Thus the receiver gets the fractal codes as input and the decompressed image as output.

4. IMPLEMENTATION

These algorithms were implemented using the software Matlab 7.12 on the Intel (R) Core[TM]2 E7500 system with 2.93 GHz and 1.96 GB of RAM. For implementation of these algorithms, four 512 × 512 benchmark color images of Lena, Pepper, Cauliflower and Taj Mahal shown in Fig.4(a) to 4(d) with twenty four-bit RGB color resolution were used.

In the simulation, the images were partitioned into range blocks with the single size, either 8 × 8 or 4 × 4 or 2 × 2. The maximum block size is set to 8 × 8 because for a range block size greater than 8 × 8 the determination of the proper domain block was difficult and the quality of the image reconstructed was poor. The threshold value for the variance of range blocks was chosen by trial and error basis to be of size 20 for block size 8 × 8, 10 for 4 × 4 and 5 for 2 × 2 that results in good compression ratio and PSNR. The number of blocks in the mean image is the size of the domain pool.

The range block with a single size (8 × 8, 4 × 4 & 2 × 2) was considered for simulation. Here the total number of range blocks for the block size 4 × 4 for each RGB color component was n = 16384 and total number of domain blocks (m) to search for each RGB color component were (128/4) × (128/4) = 32 × 32. Thus, the cardinality (N1) of the search spaces for each RGB color component of this case was 8 × 4 × 1024.



(a). Lena (b). Pepper

(c). Cauli Flower (d). Taj Mahal

Fig.4. Original (512 × 512, 24 Bit/Pixel) Images

The string length n for each RGB color component was taken to be 15 (3 + 2 + 10). Out of these 215 binary strings, forty strings (S = 40) were selected randomly to construct an initial population. A high crossover probability, say p_c = 0.85, was taken for the crossover operation. For mutation operation, p_m (mutation probability) was 0.06. Roulette-wheel selection procedure was used. The probability of selection of a string in the population to the mating pool was inversely proportional to its fitness value because the present optimization problem is a minimization problem. The total number of generations (iterations) considered in the GA was T = 60.

(a). 64 × 64 (b). 128 × 128 (c). 256 × 256

Fig.5. Mean Image of Lena for block size 8 × 8, 4 × 4 and 2 × 2

The coding performance of the proposed method using the parameters like decoded image quality and encoding time was determined. For the image partitioned by 8 × 8, 4 × 4 and 2 × 2 range blocks, the 64 × 64 and 256 × 256 mean images for Lena was obtained and shown in Fig.5(a), 5(b) and 5(c) respectively.

5. RESULTS AND DISCUSSIONS

The range blocks were classified before coding. Range blocks were grouped into two sets according to the variability of the pixel values in these blocks. If the variability of a block was low, i.e., if the variance of the pixel values in the block as below a fixed value, called the threshold, the block is called smooth type range block. Otherwise, it is called a rough type range block. The purpose of choosing this block classification was for two reasons. One is to get higher compression ratio, and the other is to reduce the coding time. The threshold value that separates the range blocks into two types was chosen as stated earlier. After classification, GA-based coding was adopted for the rough type range blocks only. All the pixel values in a smooth type range block were replaced by the mean of its pixel values. This scheme is a time-saving one provided; the number of smooth type range blocks is significant. The storage requirements for the proposed method can be calculated from the number of smooth and rough blocks multiplied by the number of bits required to store the values. Table.1 gives the classification of blocks and bit rate using different types of encoding using the proposed technique on the color images chosen for simulation. From the results tabulated in Table.1, it is observed that for the images which have the number of smooth blocks significantly high has a high compression ratio. Table.2 gives PSNR and RMS using different types of encoding using the proposed technique on the color images chosen for simulation.

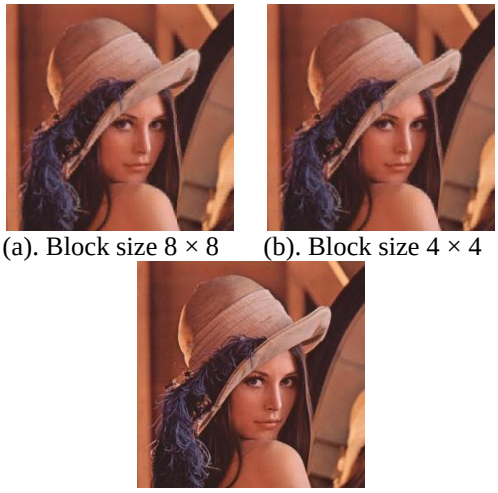
The Fig.6(a), 6(b) and 6(c) shows the decompressed Lena image using the proposed method for a single level partition of size 8 × 8, 4 × 4 and 2 × 2. The RMS of the decoded image partitioned by the 8 × 8 block size is higher than that partitioned by the 4 × 4 and 2 × 2 block size since a smaller block size leads to a smaller matching error for the affine transformation. However, the bit rate increases significantly because the number of the 2 × 2 range blocks is four times the number of the 4 × 4 range blocks and number of the 4 × 4 range blocks is four times the number of the 8 × 8 range blocks. The decompressed image of Pepper, Cauliflower and Taj Mahal for the single block partition of sizes 8 × 8, 4 × 4 and 2 × 2 using GA are shown in Fig.7, Fig.8 and Fig.9.

Table.1. Classification of Blocks Compression Ratio, Encoding Time and Bit Rate on the Chosen Images using the Proposed Technique

Image	Range	Bit Rate	Compression Ratio	No of Range Blocks		Encoding Time
				Smooth	Rough	
Lena	2 * 2	7.56	3.17	43927	21611	11033
	4 * 4	1.95	12.26	8394	7992	63871
	8 * 8	0.16	149.53	1559	2539	31250
Pepper	2 * 2	7.02	3.41	51409	14129	77985
	4 * 4	1.87	12.81	9851	6535	52157
	8 * 8	0.47	50.16	1638	2460	72323
Cauliflower	2 * 2	7.16	3.34	49433	16105	84643
	4 * 4	1.94	12.34	8640	7746	50683
	8 * 8	0.47	50.36	1682	2416	34054
Taj Mahal	2 * 2	7.51	3.19	44672	20866	11047
	4 * 4	2.05	11.65	6624	9762	64737
	8 * 8	0.50	47.58	1018	3080	43044

Table.2. Classification of Blocks RMS and PSNR value on the Chosen Images using the Proposed Technique

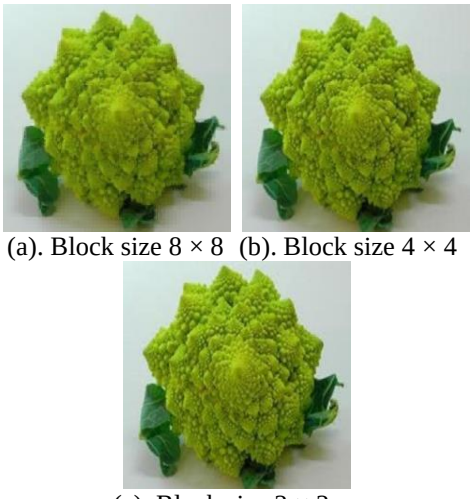
Image	Range	RMS	PSNR
Lena	2 * 2	2.12	41.60
	4 * 4	5.62	33.12
	8 * 8	9.29	28.76
Pepper	2 * 2	2.59	39.84
	4 * 4	5.37	33.52
	8 * 8	10.84	27.84
Cauliflower	2 * 2	3.34	37.63
	4 * 4	8.54	29.49
	8 * 8	15.39	24.23
Taj Mahal	2 * 2	1.84	42.83
	4 * 4	5.16	33.86
	8 * 8	11.86	26.64



(a). Block size 8 × 8 (b). Block size 4 × 4

(c). Block size 2 × 2

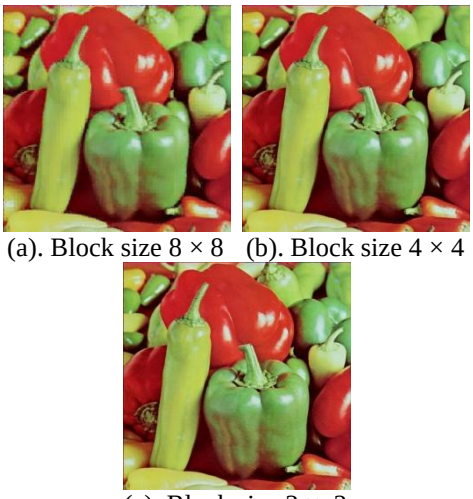
Fig.6. Lena



(a). Block size 8 × 8 (b). Block size 4 × 4

(c). Block size 2 × 2

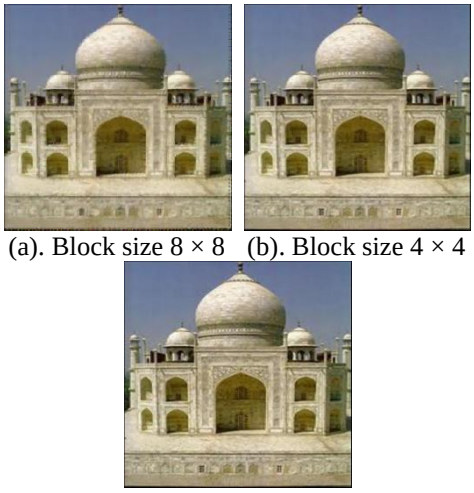
Fig.7. Cauliflower



(a). Block size 8 × 8 (b). Block size 4 × 4

(c). Block size 2 × 2

Fig.8. Pepper



(a). Block size 8 × 8 (b). Block size 4 × 4

(c). Block size 2 × 2

Fig.9. Taj Mahal

Table.3. PSNR of some methods for 512 × 512 images

Image	Range	PSNR			
		Proposed method	Rgb & Gray Scale Component on Mpq-Btc In Image Compression [11]	Color Image Compression with Modified Fractal Coding on Spiral Architecture [12]	Iteration free Fractal Compression using Simulated Annealing for Still Colour Images [13]
Lena	4 × 4	33.12	24.1209	29.05	31.68
Pepper	4 × 4	33.52	24.1531	-	32.20

The Table.3 gives the result of similar methods and the proposed methods for the bench mark image of Lena and Pepper (512 × 512, 24 bit color image). In the proposed method using GA the PSNR is highly effective when compared to the existing fractal methods [11- 13].

6. CONCLUSION

Color images are commonly used in most of the application now-a-days. An RGB color image is an M × N × 3 array of color pixels, where each color pixel is a triplet corresponding to the red, green and blue components of an RGB image at a specific spatial location. The fast encoding algorithm for iteration free fractal image coding for still colour image is implemented using GA. The proposed algorithm has the better performance in terms of image quality and coding time for RGB image. Only the encoding consumes more time but the decoding is very fast. Only the encoding consumes more time but the decoding is very fast. Applications where images can be stored in a compressed form, which require faster retrieval, like medical images and photographs for identification can use the proposed method. The execution time can be further reduced by implementing the proposed method in parallel for encoding.

ACKNOWLEDGMENT

This research work is supported by the UGC, New Delhi, India.

REFERENCES

[1] Hsuan T. Chang and Chung J. Kuo, "Iteration-Free Fractal Image Coding Based on Efficient Domain Pool Design", *IEEE Transaction on Image Processing*, Vol. 9, No. 3, pp. 329 - 339, 2000.

[2] Y. Chakrapani and K. Soundara Rajan, "Genetic Algorithm Applied To Fractal Image Compression", *Asian Research Publishing Network Journal of Engineering and Applied Sciences*, Vol. 4, No. 1, pp. 53 - 57, 2009.

[3] Erjun Zhao Dan Liu, "Fractal Image Compression Methods: A Review", *Proceedings of the Third International Conference on Information Technology and Applications*, Vol. 1, pp. 756 - 759, 2005.

[4] Raouf Hamzaoui and Dietmar Saupe, "Combining Fractal Image Compression and Vector Quantization", *IEEE Transaction on Image Processing*, Vol. 9, No. 2, pp. 197 - 208, 2000.

[5] Suman K. Mitra, C.A. Murthy and Malay K. Kundu, "Technique for Fractal Image Compression Using Genetic Algorithm", *IEEE Transaction on Image Processing*, Vol. 7, No. 4, pp. 586 - 593, 1998.

[6] N.A. Koli and M.S. Ali, "A survey on Fractal Image Compression Key Issues", *Information Technology Journal*, Vol. 7, No. 8, pp. 1085 - 1095, 2008,

[7] B. Wohlberg and G. de Jager, "A Review of the Fractal Image Coding Literature", *IEEE Transaction on Image Processing*, Vol. 8, No. 12, pp. 1716 - 1729, 1999.

[8] Yang Xuan and Liang Dequn, "An Improved Genetic Algorithm of solving IFS code of Fractal Image", *Proceedings of the Third International Conference on Signal Processing*, Vol. 2, pp. 1405-1408, 1996.

[9] Yung -Gi, Wu, Ming-Zhi, Huang and Yu-Ling Wen, "Fractal Image Compression with Variance and Mean", *Proceedings of International Conference on Multimedia and Expo*, Vol. 1, pp. I- 353 to I-356, 2003.

[10] A.R Nadira Banu Kamal, S. Thamarai Selvi and Henry Selvaraj, "Iteration Free Fractal Image Compression Using Genetic Algorithm", *International Journal of Computational Intelligence and Applications*, Vol. 7, No. 4, pp. 429 - 446, 2008.

[11] K. Somasundaram and P. Sumitra, "RGB & Gray scale Component on MPQ-BTC in Image Compression", *International Journal on Computer Science and Engineering*, Vol. 3, No. 4, pp. 1462 - 1467, 2011.

[12] Nileshsingh V. Thakur and O.G. Kakde, "Color Image Compression with Modified Fractal Coding on Spiral Architecture", *Journal of Multimedia*, Vol. 2, No. 4, pp. 55 - 66, 2007.

[13] A.R. Nadira Banu Kamal, "Iteration free Fractal Compression using Simulated Annealing for Still Colour Images", *International Journal of Advances in Artificial Intelligence and Neural Networks*, Vol. 2, No. 3, pp. 10 - 14, 2012.