Fractal Image Compression by YIQ Color Space

Eman A. Al-Hilo
Physics Department/ Kufa University/Iraq
College of Education for Girls
Emanalhilo@yahoo.com

Rusul Zehwar
Physics Department/ Kufa University/Iraq
College of Education for Girls
rurut297@yahoo.com

***Abstract*- In fractal compression the image to be encoded is partitioned into blocks called (ranges). Each range is coded by reference to some other part of the image called (domain) and by some affine transformation parameters. The number of ranges plays an important role in the compression ratio, encoding time and reconstructed image quality. In this paper, the fractal compression technique proposed by Jacquin is investigated for 24 bits/pixel color image. The data of the color component (R,G,B) are transformed to (YIQ) color space, to take the advantage of the existing spectral correlation to gain more compression. Also the low spatial resolution of the human vision systems to the chromatic components (I,Q) was utilized to increase the compression ratio without making significant subjective distortion. The test results show that PSNR (31.05) dB with CR (8.73) and encoding time (57.55) sec for Lena image (256x256) pixel.**

***Key Words*–Compression image, Fractal image compression, Iterated function system**

I. INTRODUCTION

The YIQ color space has been used in NTSC (National Television System Committee) color TV system and has been employed in USA, Canada, Japan and Korea. The Y stands for luminance components, and I and Q represent in-phase and quadrature amplitude modulation components. Thus, Y is the only component employed by black-and-white TV. On the other hand, I and Q stand for chrominance information for representing color. the advantages is the ability of separation gray scale information from color data property enables to represent the same signal for both color and (black and white) sets, using luminance component (which represent the gray scale information), but the disadvantages, the color range is restricted in the color TV images because of the information compression required for the displayed image and due to the limitation of the YIQ standard the image displayed in computer cannot be recreate in TV screen. [1]

Many researches study the color image compression. Porat has been compare color compression techniques that take advantage of this inter color correlation [2]. Al-Hilo has studied speeding fractal color image compression by moment feature includes converting the RGB model system to YUV model and minimize of the rang of U and V because of most of the image data are concentrated in the range of Y [3]

In this paper RGB values transformed to YIQ color space using equation (1.1), while YIQ values converted back to RGB using equation (1.2). [4,5]

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ 0.596 & -0.247 & -0.322 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{.....(1)}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.105 & 1.702 \end{bmatrix} . \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \quad \text{.....(2)}$$

II. GENERATE THE REDUCED IMAGE AND THE RANGE BLOCKS

In the fractal image compression the original image of size (W x H) will be reduced to its quarter size to be an image of size (*W/2 x H/2*) by using fixed blocks partitioning method with jump step (j), and then the number of range block (n_r) and the number of domain blocks (n_d) of size (*kxk*) which determined by using the following equation: [6]

$$n_d = \left(\left\lfloor \frac{W-2k}{j} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{H-2k}{j} \right\rfloor + 1\right) \quad \text{.....(3)}$$

Taking into consideration that each range/domain matching instance involves 8 square symmetry versions, the total number of domain blocks (n_m) in the overall range/domain matching processes will be:

$$n_m = 8 n_r n_d \quad \text{..... (4)}$$

This process will be responsible for reducing the number of the image range blocks to about its quarter number using any partitioning techniques. A fixed size partitioning scheme is used to partition the domain. Thus the domain pool will be divided into “domain blocks” but possibly they can be overlapping blocks with a certain step size (in pixel) for the horizontal and vertical shift. Overlapping blocks increase the number of possible domain blocks listed in the domain pool, and thus better approximation may be obtained. As the step size is small, the domain pool will be large and this will ensure the good approximation and high quality of the reconstructed image. But at the same time this will lead to high encoding time because searching a pool consisting of a large number of domain blocks is time-consuming.

Choosing a big step size reduces the encoding time but the reconstructed image will have low quality. It is important to notice that the step size must be less than or equal to the block size.

978-1-4799-3010-4/14 $31.00 © 2014 IEEE
DOI 10.1109/CSCI.2014.45

III. MATCHING PROCESS

After the generation of the range and domain pools; one takes each range block listed in the range pool and map it with all the domain blocks listed in the domain pool. At each mapping instance one determines the mapping coefficients, i.e., scale (s) and offset (o), which are called the IFS coefficient. These parameters (s) and (o) are determined by applying the least sum X^2 of square errors between r_i' and r_i according to following equation: [7]

$$x^2 = \sum_{i=0}^{n-1}(r_i' - r_i)^2 \quad \text{.....(5)}$$

The minimum of x^2 occurs when:

$$\frac{\partial x^2}{\partial s} = 0 \text{ And } \frac{\partial x^2}{\partial o} = 0 \quad \text{.....(6)}$$

Substituting equation (4) in (5) and using equations (6) one gets:

$$s = \frac{n\sum_{i=0}^{n-1} r_i d_i - \sum_{i=0}^{n-1} r_i \sum_{i=0}^{n-1} d_i}{n\sum_{i=0}^{n-1} d_i^2 - \left(\sum_{i=0}^{n-1} d_i\right)^2} \quad \text{..... (7)}$$

$$o = \frac{\sum_{i=0}^{n-1} r_i \sum_{i=0}^{n-1} {d_i}^2 - \sum_{i=0}^{n-1} r_i d_i \sum_{i=0}^{n-1} d_i}{n\sum_{i=0}^{n-1} d_i^2 - \left(\sum_{i=0}^{n-1} d_i\right)^2} \quad \text{.....(8)}$$

$$x^2 = \frac{1}{n}\left[\sum_{i=0}^{n-1} {r_i}^2 + s\sum_{i=0}^{n-1} d_i^2 - 2\sum_{i=0}^{n-1} r_i d_i + 2o\sum_{i=0}^{n-1} d_i + o(no - 2\sum_{i=0}^{n-1} r_i)\right] \quad \text{.....(9)}$$

where, d_i is the i^{th} pixel value of the matched domain block.

r_i is the i^{th} pixel value of the range block.

n is the number of pixels in each block (i.e. the block size).

At each range-domain matching instance and before the determination of x^2, some conditions should be applied on the determined values of (s) and (o), these conditions are:

1. The value of (s) must be within a bounded interval ($s_{min} \leq s \leq s_{max}$). It was mentioned in the literature that the value of the minimum scale (s_{min}) must be not less than (-1), while the maximum scale value (s_{max}) must be not more than (1). So, the following conditional statement had been applied:

If s < MinScale then s=MinScale

Else if s > MaxScale then s=MaxScale

2. The value of (o) must be within a limited interval ($o_{min} \leq o \leq o_{max}$). The minimum offset ($o_{min}$) value must be not less than (-256), while the maximum offset value (o_{max}) must be not more than (511). The following conditional statement was been applied:

If o <MinOffset then o=MinOffset

Else if o > MaxOffset then o=MaxOffset

IV. ENCODING IN THE YIQ TECHNIQUE

The implementation encoding method could be summarized by the following steps:

1. Load BMP image and put it in (R,G,B) arrays (i.e., three 2D arrays).
2. Convert (R,G,B) arrays to (Y,I,Q) arrays.
3. Down sample I and Q to quarter of its original size.
4. For each component (i.e., the original Y, and the down sampled I,Q) do:
   a. Establish the range array
   b. Down sample the range image to produce the domain array.
   c. Partitioning:
      (1) The range array must be partitioned into non-overlapping fixed blocks, to generate the range blocks ($r_1,\ldots,r_n$).
      (2) The domain must be partitioned into overlapping blocks, using specific step size, to generate the domain blocks ($d_1,\ldots,d_n$). They should have the same size of range blocks.
   d. Searching:
      (1) Pick up a domain block from the domain pool.
      (2) Perform one of the isometric mappings.
      (3) Calculate the scale (*s*) and offset (o) coefficient using equations (7,8).
      (4) Apply the following condition to bound the value of (*s*) and offset (o) coefficient:

If $s < s_{min}$ *then* $s=s_{min}$

Else if $s > s_{max}$ *then* $s=s_{max}$

If $o < o_{min}$ *then* $o=o_{min}$

Else if $o > o_{max}$ *then* $o=o_{max}$

      (5) Quantize the value (*s*) and offset (o) using equations (13-18).
      (6) Compute the approximation error (χ^2) using equation (9).
      (7) After the computation of IFS code and the sum of error (χ^2) of the matching between the range and the tested domain block., the (χ^2) is compared with registered minimum error (χ^2_{min}); such that:
      If $\chi^2 < (\chi^2_{min})$ *then*

$s_{opt}=i_s$; $o_{opt}=i_o$, $\chi^2_{min}=\chi^2$

PosI=domain block index

Sym=symmetry index

End if

(8) If $\chi^2_{min} < \varepsilon$ then the search across the domain blocks is stopped, and the registered domain block is considered as the best matched block

(9) Repeat steps (4) to (10) for all symmetry states of the tested domain block.

(10) Repeat steps (3) to (11) for all the domain blocks listed in the domain pool.

(11) The output is the set of IFS parameters $(i.e., i_s, i_o, posI, Sym)$ which should be registered as a set of fractal coding parameters for the tested range block.

(12) Repeat steps (1) to (12) for all range blocks listed in the range pool

(13) Store all IFS mapping parameters as an array of record. The length of this array is equal to the number of range blocks in the range pool.

V. DECODING IN THE YIQ TECHNIQUE

The decoding process by YIQ model can be summarized in the following steps:-

1. Generating arbitrary domain pool. The domain pool could be initialized as a blank image or a piece of image extracted from any available image.

2. The values of the indices of (i_s) and (i_o) for each range block should be mapped to reconstruct the quantized values of the scale (s_q) and offset (o_q) coefficients. This mapping could be done by using equations (10, 12, 13 and 15). This step is called the dequantization step.

3. Choosing the number of possible iterations, and the threshold value of the mean square error (TMSE). At each iteration the following steps are performed:

a. For each range block one determines the coordinates (x_d, y_d), of the best matched domain, from the IFS parameters (*posI*), in order to extract the domain block (*d*) from the arbitrary domain image.

b. For each range block, its approximation r_i' is obtained by multiplying the corresponding best matched domain block (*d*) by the scale value (s_q) and adding to the result the offset value (o_q), according to equation:.

$$r_i' \approx sd_i + o \quad \ldots\ldots(16)$$

c. The generated block is transformed (rotated, reflected, or both) according to its corresponding IFS symmetry parameter value (*Sym*).

d. The generated r_i' block is placed in its position in the decoded image array (range image).

e. Checking whether there is another range block, if yes then steps (b,c,d) are repeated.

f. Down sampling the reconstructed image (range pool) in order to produce the domain pool by using the averaging (or integer) sampling.

g. Calculating the mean square error (*MSE*) between the reconstructed range and the previous reconstructed range image. If the MSE is greater than (*TMSE*) value then the iteration continued and the above steps (*a-f*) should repeated; this iteration is continued till reaching the attractor state (i.e.; the newly reconstructed range image is very similar to the previous reconstructed image), otherwise the iteration continue till reaching the predefined maximum number of iterations (m), in our program used (m=20).

4. The above (steps 3a-3g) should be implemented upon the three components (Y,I,Q) to produce the attractor of each component.

5. Converting the reconstructed (YIQ) color components to RGB components sing the inverse YIQ transform equations (1).

6. Calculating the fidelity criteria () for each RGB component, then determines the overall value of the fidelity criteria for the RGB reconstructed image.

VI. TESTS RESULTS

This work is carried out in Visual Basic 6.0 version on Laptop (*hp*): intel (R) Core (TM) i5-2430M CUP @ 2.40 GHz Processor, 64-bit Operating System and 6.00 GB RAM. To evaluate the performance of the established colour FIC system by YIQ model, the proposed system has been tested using Lena image (256x256 pixel, 24bits) as test image. These tests explore the effect of the following coding parameters on the compression performance parameters of the established system:

A. Maximum and Minimum Scale Tests

This set of tests was conducted to study the effect of MinScale, and MaxScale on the compression performance parameters of the reconstructed image. In these tests the value of compression parameters are set in table (1). The effects of this test shows in table (2). Figure (1,2,3) shows the effects of this test graphically:

Table (1) The set of Maximum and Minimum Scale tests

ScaleBits	offseBits	StepSize	BlockSize	DomSize	ε_o	TMSE
7	8	2	4×4	128x128	0.5	0.2

Table (2) Effects of MaxScale and MinScale on the compression performance parameters

MaxScale	MinScale	PSNR	CR	ET(sec)
1	-1	30.85	8.73	54.65
	-1.5	30.94	8.73	54.34
	-2	30.94	8.73	54.26
	-2.5	30.92	8.73	54.46
	-3	30.90	8.73	53.96
2	-1	31.02	8.73	54.99
	-1.5	31.05	8.73	54.17
	-2	31.03	8.73	54.36
	-2.5	31.02	8.73	53.99
	-3	31.01	8.73	54.97
3	-1	31.04	8.73	54.25
	-1.5	31.04	8.73	54.19
	-2	31.05	8.73	54.76
	-2.5	31.04	8.73	56.56
	-3	30.98	8.73	55.27

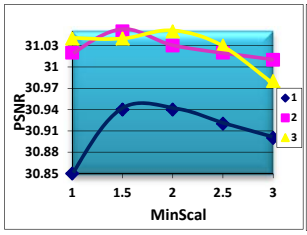


Fig (1) Effect of MinScale on PSNR

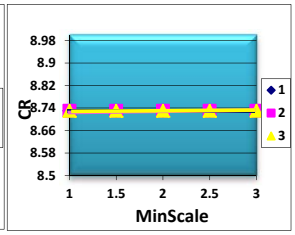


Fig (2) The effect of MinScale on CR

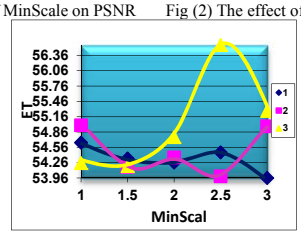


Fig (3.) The effect of MinScale on ET

B. *Step Size Tests*

In this test the effect of the shift StepSize parameter is studied. The values of compression parameters were set as in table (3). The effects of this test shows in figure (4). Figure (5,6,7) show the effects of this test graphically.

Table (3): The coding parameters that set in stepSize test

ScaleBits	offseBits	MinScal	MaxScal	DomSize	ε_0	TMSE
7	8	-1.5	2	128x128	0.5	0.2

Original image	StepSize (1)	StepSize (2)	StepSize (3)	StepSize (4)
				
PSNR	31.5	31.05	30.71	30.48
CR	8.73	8.73	8.73	8.73
ET	208.23	53.45	24.08	14.20

Fig (4) Effect of different StepSize values on the compression performance parameters

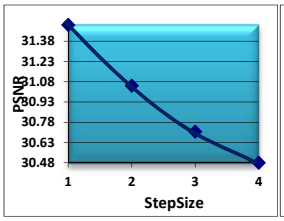


Fig (5) The effect of StepSize on PSNR

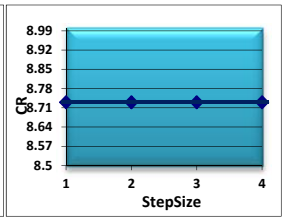


Fig (6) The effect of BlockSize on CR

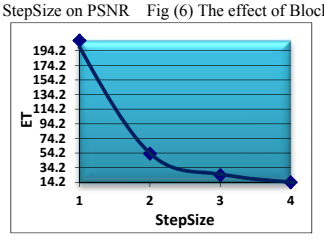


Fig (7) The effect of BlockSize on ET

C. *Block Size Tests*

This set of conducted tests aimed to investigate the effects of the BlockSize parameter on the compression performance parameters. In this test the value of compression parameters are set as in table (4). The effects of this test shows in figure (8). Figures (9,10,11) show the effects of this test graphically.

Table (4): The coding parameters that set in BlockSize test

ScaleBits	offseBits	MinScal	MaxScal	StepSize	DomSize	ε_0	TMSE
7	8	-1.5	2	2	128x128	0.5	0.2

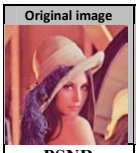
Original image	BlockSize (2×2)	BlockSize (4×4)	BlockSize (8×8)	BlockSize(16×16)
				
PSNR	34.32	31.05	27.15	23.68
CR	2.04	8.73	37.46	161.68
ET	30.68	53.57	31.27	24.17

Fig (8) Effects of BlockSize on the compression performance parameters

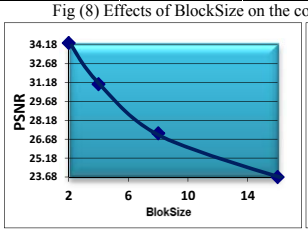


Fig (9) The effect of BlockSize on PSNR

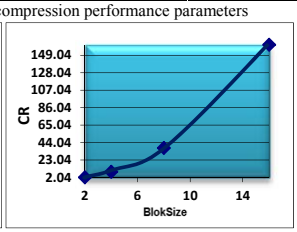


Fig (10) The effect of BlockSize on CR

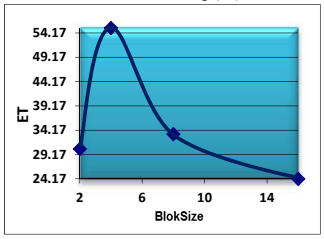


Fig (11) The effect of BlockSize on ET

D. *Domain Size Tests*

This set of tests was performed to define the effect of the DomSize on the compression parameters. The values of coding parameters were taken as in table (5). The effects of this test shows in figure (12). Figures (13,14,15) show the effects of this test graphically.

Table (5): The coding parameters that set in domain size test

ScaleBits	offseBits	StepSize	BlockSize	ε_0	TMSE
7	8	2	4×4	0.5	0.2

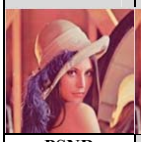
Original image	DomSize (128x128)	DomSize (64x64)	DomSize (32x32)	DomSize (16x16)
				
PSNR	31.05	30.49	29.49	27.80
CR	8.73	8.73	8.73	8.73
ET	53.75	13.24	3.48	1.14

Fig (12): The effects of DomSize on the compression performance parameters

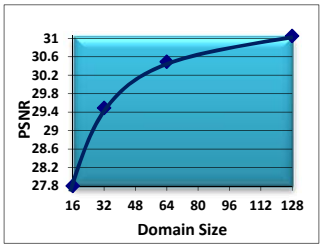


Fig (13) The effect of DomSize on PSNR

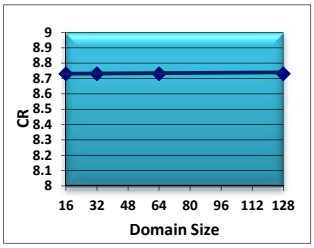


Fig (14) The effect of DomSize on CR

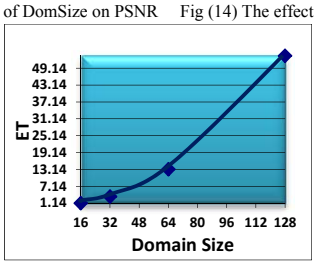


Fig (15) The effect of DomSize on ET

E. Permissible Error Value (ε_o) Tests

This set of tests was performed to study the effects of permissible error level (ε_o) on the compression performance parameters. The values of coding parameters were taken as in table (6). The effects of this test shows in table (7). Figures (16,17,18) show the effects of this test graphically.

Table (6): The coding parameters that set in stepsize test

ScaleBits	offseBits	BlockSize	StepSize	DomSize	TMSE
7	8	4×4	2	128x128	0.2

Table (7) Effects of permissible error value (ε) on the compression performance parameters

ε_o	PSNR	CR	ET(sec)
0.1	31.04771	8.73	54.72
0.2	31.04769	8.73	55.36
0.3	31.04751	8.73	54.49
0.4	31.04747	8.73	54.27
0.5	31.04799	8.73	54.27
0.6	31.04749	8.73	53.62
0.7	31.04502	8.73	55.33
0.8	31.04414	8.73	55.00
0.9	31.04101	8.73	51.67
1	31.03254	8.73	50.32
2	30.82562	8.73	33.21
3	30.36689	8.73	20
4	29.95941	8.73	14.22
5	29.53438	8.73	10.10

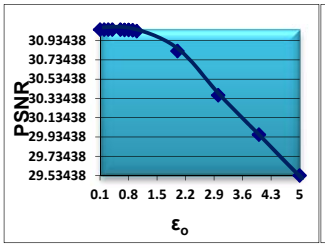


Fig (16) The effect of ε_o on PSNR

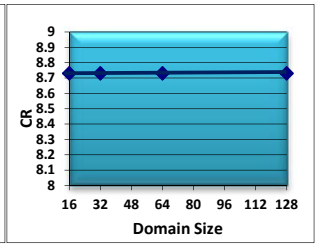


Fig (17) The effect of ε_o on CR

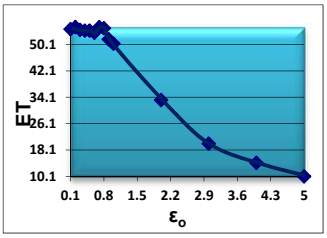


Fig (18) The effect of ε_o on ET

VII. CONCLUSIONS

The following remarks summarize the noticed behaviour in the above listed results:-

1. ***PSNR*** is directly proportional to DomSize, but inversely proportional to BlockSize, StepSize and permissible error value (ε).
2. ***ET*** is directly proportional to DomSize and it is inversely proportional to BlockSize, StepSize and permissible error value (ε), but it does not MinScale and MaxScale.
3. ***CR*** is directly proportional to BlockSize, but it does not depend on the value of MinScale, MaxScale, StepSize, DomSize, and permissible error value (ε).

REFERENCES

[1] Ibraheem, N., Hasan, M., Khan, R., Mishra, P., **" Understanding Color Models: A Review",** ARPN Journal of Science and Technology, VOL. 2, NO. 3, April 2012.

[2] Porat M., **"The Effect of Inter Color Correlation on Color Image Compression",** M.Sc., Electrical Engineering, January 2001.

[3] Loay E. George, Eman A. Al-Hilo, **"Speeding-up Fractal Color Image CompressionUsing Moments Features Based on Symmetry Predictor"** 8th International Conference on Information Technology: New Generations ITNG 2011, April 11-13, 2011, Las Vegas, Nevada, USA, (pages 508-513).

[4] Ford A., Roberts A., **"Color Space Conversions",** August11, 1998(b).

[5] González J. M. C., Rodríguez M. A. V., Pulido J. A. G., and Pérez J. M. S., **"Digital Signal Processing",** Available online 23 October 2009.

[6] Al-Hilo, E.A., George, L. " **Fractal Color Image Compression** ", Proceedings of the 4 International Conference on Information Technology and Multimedia at UNITEN, Malaysia , 2008

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