Fractal Color Image Compression

ZHAOHUI LI[1]
LIANG ZHAO[2]
NEI Y. SOMA[1]
[1]ITA-Instituto Tecnológico de Aeronáutica, CTA/ITA/IEC, CEP 12228-900, São José dos Campos, SP, Brasil
{zhaohui, nysoma}@comp.ita.cta.br
[2]ICMC-USP-Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Campos de São Carlos. CEP 13560-970, São Carlos, SP, Brasil
zhao@icmc.sc.usp.br

Abstract. This paper presents a new fractal color image compression method, called *Fractal Hierarchical Color Block Coding* (*FHCBC*), which transforms the three-color planes of a color image into a one-component image by extracting correlation among them. It hierarchically divides the three-color planes into homogeneous blocks, and for each such block, the variance of pixels' trichromatic coefficients is within a certain threshold value. Then, each block is represented by its mean value of the pixels' trichromatic coefficient ratios, and just a one-component image is composed and compressed by fractal coding. In the decoding process, the reconstruction is carried out by a fractal decoding algorithm and the associated mean values of trichromatic coefficient ratios. In comparison with the well known three-component *Separated Fractal Coding* (*SFC*), the suggested method generate performance gains in running time, high compression ratio and equal level of reconstruction quality.

1 Introduction

Data compression has become an important issue for information storage and transmission. This is especially true for databases consisting of a large number of detailed computer images [4, 5, 7]. Recently, a large quantity of methods has appeared in the literature for achieving high compression ratios for compressed image storage, and among them, the fractal approach become a feasible and promising compression technique. This assumption is backed up by noticing its inclusion into end user products such as Microsoft's Encarta or as a Netscape plug-in by Iterated Systems Inc. [2]. Fractal image compression exploits the natural affine redundancy present in typical images to achieve high compression ratios in a lossy compression format. The main idea of the method consists in finding a construction rule that produces a fractal image, approximating to the original one. Fractal image coding has its roots in the mathematical theory of *iterated function systems* (IFS) developed by Barnsley [1, 2], whilst the first fully automated algorithm was developed by Jacquin [12].

Up to now, image compression research is mainly performed on gray-level images, with relatively few results on color images, moreover, this is especially true for fractal coding. However, we encounter much more color images than gray-level images on day-to-day applications. Additionally, any gray-level image can be considered as a special case of a color image.

There are many color image representations [8]. For the convenience display colors on the monitor, a true color image is most commonly represented by 24 bits per pixel in a *RGB* color space with each component *R*, *G* and *B* assigned 8 bits [8]. Thus, the most straightforward method to encode a color image by gray-level fractal image coding algorithm is to split the *RGB* color image into 3 channels, red, green and blue, and compress them separately by treating each color component as a single gray-scale image [17], the so called three-component *Separated Fractal Coding* (*SFC*). This method, however, does not exploit the correlation among color components (inter-color-plane redundancy) resulting in a relatively low compression ratio and taking a long processing time.

In this paper, we introduce a new color image coding algorithm, called *Fractal Hierarchical Color Block Coding* (*FHCBC*). It is based on the *RGB* color model and by hierarchically partitioning the three-color planes into strongly correlated blocks. Then, only one color-plane needs to be coded, while the other two can be automatically reconstructed from the encoded color plane and correlation among them. In comparison with *SFC*, *FHCBC* can achieve the same level of reconstruction quality, a much higher compression ratio and, especially, one-third of the compression time.

This paper is organized as follows: Section 2 reviews the general idea and a classical algorithm of the fractal gray-level image coding. Section 3 introduces the *FHCBC* method, and in the next one, computer simulation results

0-7695-0878-2/00 $10.00 © 2000 IEEE

are presented. Finally, the last section presents some conclusions.

2 Fractal Image Compression

In this section, we describe, briefly, the fractal image compression technique. The mathematical foundations were laid in [1, 2]. Detailed description on computer implementations can be found in [6, 11] or [15].

It is well known that complex fractal images can be generated by iterating very few simple maps [1, 2, 18]. If we directly store these fractal images as a collection of pixels, clearly, a large amount of memory will be required; however, if just the maps are stored, only a fraction of the former is necessary. This is the main motivation for using the fractal image coding study. In this way, the corresponding maps can be considered as a compressed form of a fractal image, and the original image can be reconstructed by iterating these maps.

In order to guarantee the convergence of iterations, the set of maps should be *contractive* [1, 2]. Informally, a map is *contractive* if the distance of any two points in the image space tends to zero under iterating of the map.

Usually, we cannot expect to find a single map to represent a whole natural image, but it is always possible to find self-similarities amidst different portions of the same one. This suggests that we can divide the original image into a number of sub-image blocks and try to match the transformed sub-image blocks with other parts of the same image. Specifically, we partition the image twice to get the following two type of blocks: 1) a collection of non-overlapped smaller sub-image blocks, called *range blocks* $\{R_i\}_{i=1}^{N}$ and 2) a collection of possibly overlapping larger sub-image, called *domain blocks* $\{D_i\}_{i=1}^{M}$. For each range block R_i, the domain pool is searched to find a best matched domain block under certain transformation ω_i. This range block is, then, represented by the obtained transformation (map) ω_i. Finally, the whole image is represented by a collection of maps ω_1, ω_2, ..., ω_N, such that $W = \bigcup_{i=1}^{n} \omega_i$. The original image is an attractor of W, i.e., it can be reconstructed by iterating W on any initial image configuration.

For its simplicity and suitability to the image processing handling, usually the affine transformation is utilized to limit the form of maps:

$$\omega_i \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix}$$

where subscript i indicates the transformation of ith range block; x_i and y_i represent the pixel coordinates, and z_i represents the pixel intensity at point (x_i, y_i). a_i, b_i, c_i, d_i, e_i, f_i, are geometrical transform parameters, where a_i, d_i are scaling factors, b_i, c_i are rotation factors and e_i, f_i are translation quantities. While s_i and o_i are pixel intensity transform parameters with s_i the contrast factor, and o_i the intensity shift.

The set of parameters a_i, b_i, c_i, d_i, e_i, f_i, s_i, o_i defines, hence, a transformation between a range block and a domain block.

One cannot expect to get a collection of maps, which generate exactly the original image, since images are not composed of pieces as puzzles, that can be transformed to reconstruct exactly. However it can be expected in general, that another image is generated, where the difference between the generated image and the original image is small. This implies in a lossy compression technique.

Since domain blocks are larger than range blocks, a transformation from a domain block to a range block always shrinks the distance between any two pixels of the domain block. Thus, transformations extracted by this method is always contractive. Then, there exists a unique attractor, which is approximately the original image, for the collection of transformations. This is guaranteed by Contractive Map Theorem [1, 2].

If each range block is well matched by a domain block, the reconstructed image will be sufficiently similar to the original image. This is guaranteed by the Collage Theorem [1, 2].

We summarize next, the main steps of the classical fractal image compression algorithm.

Step 1: Partition the original image into a collection of non-overlapping range blocks $\{R_i\}_{i=1}^{N}$; and

Step 2: a collection possibly overlapping of larger sub-images (domain blocks) $\{D_i\}_{i=1}^{M}$.

Step 3: The domain blocks are filtered and sub-sampled using pixel averaging so that it occupies the size of range blocks, that is,

$$f_{i,j}^{(D')} = (f_{2i,2j}^{(D)} + f_{2i,2j-1}^{(D)} + f_{2i-1,2j}^{(D)} + f_{2i-1,2j-1}^{(D)})/4$$

Step 4: For each range block R_i, search the domain pool (including contracted domain blocks with eight self-symmetrical transformations) until a contracted domain block with a smallest mean square difference of the range block is reached; Once the contracted domain block is found, the affine transformation parameters a, b, c, d, e, f are determined immediately. The two additional

parameters s and o are determined by the regression method proposed by Fisher [6].

Step 5: Store this mapped transformation coefficients.

Step 6: Repeat Step 4 for all other range blocks.

3 Fractal Hierarchical Color Block Coding (FHCBC)

Fractal Hierarchical Color Block Coding (*FHCBC*) algorithm consists on two processings: *Hierarchical Color Block Coding* (*HCBC*) and one-color plane fractal coding.

The general idea behind *HCBC* is to exploit the strong correlation among three-color components. *HCBC* is based on quadtree partitioning [14, 19], which subdivides recursively a region of an image into four equal blocks if a given criterion is not met within that region. It continues to divide each subdivision until a predefined criterion is met or a pre-defined minimum block size is reached.

The structure of the quadtree depends on the inter-pixel correlation of the image that it represents. Areas with the highest correlation, that is, little variation of pixels' trichromatic coefficients, compose the high level of the quadtree, which represent the large uniform blocks. Areas which varies the most fall into the lowest level of the quadtree. Notice that this recursive image splitting imply, that larger blocks achieve higher compression ratio, while small blocks preserve details of non-correlated region.

Specifically, given a color image to be encoded, *HCBC* first partitions the image into sub-image blocks, where to each such block, pixels' trichromatic coefficients has no variation or a small one. Then, each block is represented by a mean value of all pixels' trichromatic coefficient ratios and these mean values together with the information of blocks, such as position and size of a block, are stored as the first part of the compressed image. After that, *HCBC* composes a one-color plane image (gray-level image) S with weighted sum of the three-color components, and in sequence, the second part of processing starts, i.e., the composite image S is compressed by invoking the fractal coding algorithm. Then, a set of transformations obtained at this stage is stored as the second part of compressed image.

At the decoding process, fractal coding algorithm is first applied to get a one-color plane image, say S', where $S' \approx S$. Then, *HCBC* is invoked to restore the three-color planes by utilizing S' and the mean values of trichromatic coefficients previously memorized. The following figure illustrates the whole process of *FHCBC*.

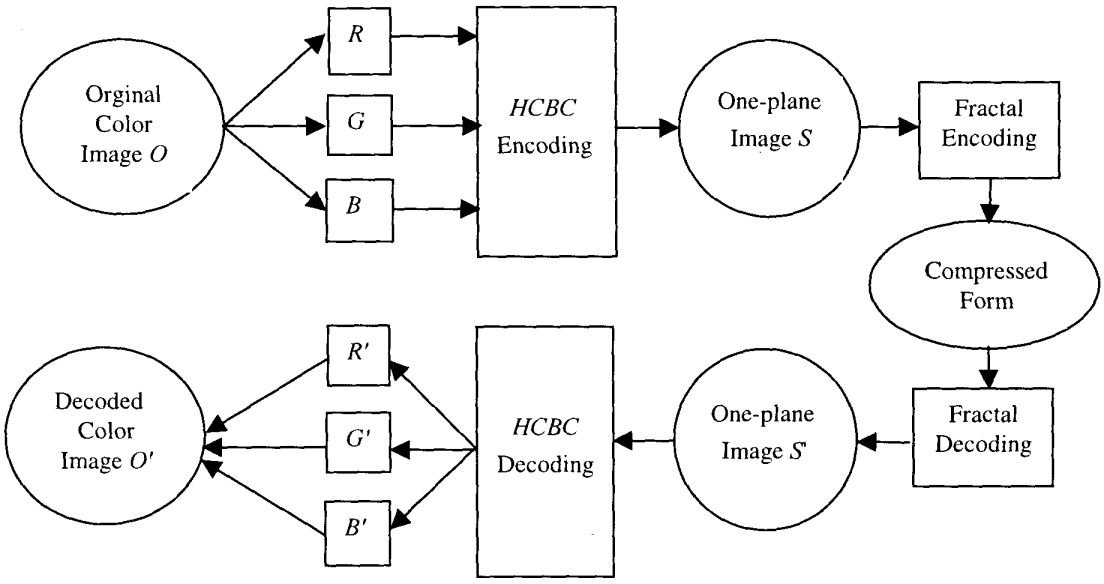

Figure 1: Block diagram of *FHCBC*

The encoding and decoding algorithm of *FHCBC* is given next. The equations used by this algorithm are shown just after the algorithm.

Fractal Hierarchical Color Block Encoding Algorithm

Step 1: Read a color image O with size $N \times N$.

Step 2: Calculate the trichromatic coefficients for each pixel according to equation (1)

Step 3: Decompose the image recursively by quadtree partitioning.

1) Partition the image in 4 non-overlapped blocks with size $\frac{N}{2} \times \frac{N}{2}$.

2) For each such a block, calculate the average trichromatic coefficient and variances according to equations (2)~(3).

3) The process of division continues, if equation (4) (with a pre-defined tolerance) is not satisfied. When this occurs, the block will be divided into another four quadrant sub-blocks, and for each of them, the same process 2), 3), is repeated again. The process of quarter halts if any one of the two conditions are satisfied:

   a) the pre-defined maximum dividing level is reached;

   b) or, equation (4) is satisfied.

4) If all of the blocks have been processed (recursively returned), the whole process terminates.

Step 4. Use equation (5) produce a new image, S, with a value of S_{ij} associated to each pixel. The image S and the mean value of trichromatic coefficient ratio of each block are stored as the first part of compressed image.

Step 5. Use fractal coding algorithm (see Section 2) to compress the composite image S. Store the parameters of affine transformations as the second part of compressed image.

Fractal Hierarchical Color Block Decoding Algorithm

Step 1. Use the fractal decoding algorithm and the second part of compressed image to decode image S', where $S_{ij} \approx S_{ij}$.

Step 2. Use equation (6) and the first part of compressed image to get $R_{ij}', G_{ij}', B_{ij}'$, where $R_{ij} \approx R_{ij}'$, $G_{ij} \approx G_{ij}'$, $B_{ij} \approx B_{ij}'$.

Step 3. Use $R_{ij}', G_{ij}', B_{ij}'$ to reconstruct image O', where $O_{ij} \approx O_{ij}'$.

Equations Used by the Algorithm FHCBC

$$X_{ij} = \frac{R_{ij}}{R_{ij} + G_{ij} + B_{ij}}$$

$$Y_{ij} = \frac{G_{ij}}{R_{ij} + G_{ij} + B_{ij}} \qquad (1)$$

$$Z_{ij} = \frac{B_{ij}}{R_{ij} + G_{ij} + B_{ij}}$$

$$M_X^{I_r} = \frac{1}{m^2} \sum_{i=I_x}^{m+I_x} \sum_{i=I_y}^{m+I_y} X_{ij}$$

$$M_Y^{I_r} = \frac{1}{m^2} \sum_{i=I_x}^{m+I_x} \sum_{i=I_y}^{m+I_y} Y_{ij} \qquad (2)$$

$$M_Z^{I_r} = \frac{1}{m^2} \sum_{i=I_x}^{m+I_x} \sum_{i=I_y}^{m+I_y} Z_{ij}$$

$$V_X^{I_r} = \frac{1}{m^2} \sum_{i=I_x}^{m+I_x} \sum_{j=I_y}^{m+I_y} \left| X_{ij} - M_X^{I_r} \right|$$

$$V_Y^{I_r} = \frac{1}{m^2} \sum_{i=I_x}^{m+I_x} \sum_{j=I_y}^{m+I_y} \left| Y_{ij} - M_Y^{I_r} \right| \qquad (3)$$

$$V_Z^{I_r} = \frac{1}{m^2} \sum_{i=I_x}^{m+I_x} \sum_{i=I_y}^{m+I_y} \left| Z_{ij} - M_Z^{I_r} \right|$$

$$V_X^{I_r} \le tol \quad and \quad V_Y^{I_r} \le tol \quad and \quad V_Z^{I_r} \le tol \qquad (4)$$

$$S_{ij} = R_{ij} X_{ij} + G_{ij} Y_{ij} + B_{ij} Z_{ij} \qquad (5)$$

$$R_{ij}' = \frac{M_R^{I_r}}{(M_R^{I_r})^2 + (M_G^{I_r})^2 + (M_B^{I_r})^2} S_{ij}'$$

$$G_{ij}' = \frac{M_G^{I_r}}{(M_R^{I_r})^2 + (M_G^{I_r})^2 + (M_B^{I_r})^2} S_{ij}' \qquad (6)$$

$$B_{ij}' = \frac{M_B^{I_r}}{(M_R^{I_r})^2 + (M_G^{I_r})^2 + (M_B^{I_r})^2} S_{ij}'$$

where R_{ij}, G_{ij}, B_{ij} represents red, green and blue components of pixel (i, j), respectively. X_{ij}, Y_{ij}, Z_{ij} are called *trichromatic coefficients* of pixel (i, j), which describe the percent of each color component in their

sum. M_X^L, M_Y^L, M_Z^L represents the mean value of trichromatic coefficient X, Y, and Z on block L, respectively. m is the block size; L_x, L_y are x- and y-coordinates of the first pixel of block L. V^L, V_Y^L, V_Z^L represents variances of trichromatic coefficient X, Y, and Z on block L, respectively. S represents composite image to be encoded. S' is the decoded S, i.e., $S \approx S'$. R'_{ij}, G'_{ij} and B'_{ij} are decoded red, green and blue components of pixel (i, j), respectively, i.e., $R'_{ij} \approx R_{ij}$, $G'_{ij} \approx G_{ij}$, $B'_{ij} \approx B_{ij}$.

The real key to color image processing is to first choose the right color model for the job. With all the definitions given above, we can answer the reason for choosing the *RGB* model in *HCBC*. Intuitively, the variance of trichromatic coefficients is small when the three-color planes are strongly correlated, in such a case, larger blocks can be obtained, implying as consequence, that higher compression ratio can be achieved. Table 1 [11] shows the correlation and variance among three-color planes of three representative color models. From the table, we find that color planes in *RGB* model have highest correlation (indicated by non-diagonal values of covariance) than the other models. On the other hand, in *KL* and *YCbCr* models, signal energy (indicated by variance) strongly concentrates on one of the three color components, and in this case, the other two color planes would be almost uniform, while the energy plane would have high pixel variance, and hence, the correlation among the three-color planes would be very low. This is confirmed by Table 1. Thus, the *RGB* model was chosen by *HCBC*.

	Covariance			Variance
R	1.0000	0.9930	0.9784	34.23%
G	0.9930	1.0000	0.9916	35.02%
B	0.9784	0.9916	1.0000	30.75%
	1.0000	0.0000	0.0000	99.25%
KL	0.0000	1.0000	0.0000	0.66%
	0.0000	0.0000	1.0000	0.09%
Y	1.0000	-0.4198	0.2710	98.24%
Cb	-0.4198	1.0000	-0.9424	0.64%
Cr	0.2710	-0.9424	1.0000	1.12%

Table 1: Second-order statistic of *RGB*, *HSI*, *KL* and *YCbCr* images

Another question is interesting to be noticed why homogenous blocks are obtained by the evaluation of the variance of pixels' trichromatic coefficients instead of directly evaluating variance of pixels values of each color plane. This effect occurs because if all of the three-color planes have an uniform pixel value distribution, they have necessarily uniform trichromatic coefficient ratios, with the converse assertion is not always true. For example, suppose that a block with four pixel have values R(10, 10, 10, 10), G(40, 40, 40, 40), B(50, 50, 50, 50), one can see that each one of the three-color plane generates an uniform image. In this case, the trichromatic coefficient ratio (0.1 : 0.4 : 0.5) is a constant too. On the other hand, if a block of four pixels has the values: R(1, 2, 3, 4), G(4, 8, 12, 16), B(5, 10, 15, 20), the trichromatic coefficient ratio is still uniform (0.1 : 0.4 : 0.5), however, neither one of the three-color plane has uniform pixel values. This means that larger blocks can be obtained by detecting uniformity of trichromatic coefficients rather than by detecting uniformity of pixel values. This observation may explain why higher compression ratio can be achieved by the suggested method.

4 Experimental Results

This section presents the results of computer simulations preformed on a 512×512 color images *Lena* by *FHCBC* and *SFC*. In both cases of the fractal coding experiments, the domain block size is set to 16×16, while the range block size is set to 8×8.

Figure 2-a) shows the original color image *Lena*. Fig. 2-b) and Fig. 2-d) are reconstructed image by *FHCBC* and *SFC* respectively. The reconstruction quality is quite good by both methods. However, it is difficult to judge which method generates the best quality based only on subjective observation

From Fig. 2-f), one can see that large blocks can be obtained by *FHCBC* coding. As explained before, this effect improves the compression ratio.

Although the three-color planes are not shown here, it is still possible to realize that the signal energy is concentrated on the red color plane, i.e., it is a dominant color plane. This implies that the decoding error is mainly decided by the red color plane. In this case, if the green or the blue color plane were chosen to perform the *FHCBC* coding, the reconstructed image would not repeat the same performance. This explains why *FHCBC* coding is performed on a composite image S (obtained by equation (5)). Notice that S is, indeed, a weighted average image on the three-color components, and this avoids the selection of a very low energy color plane in either case.

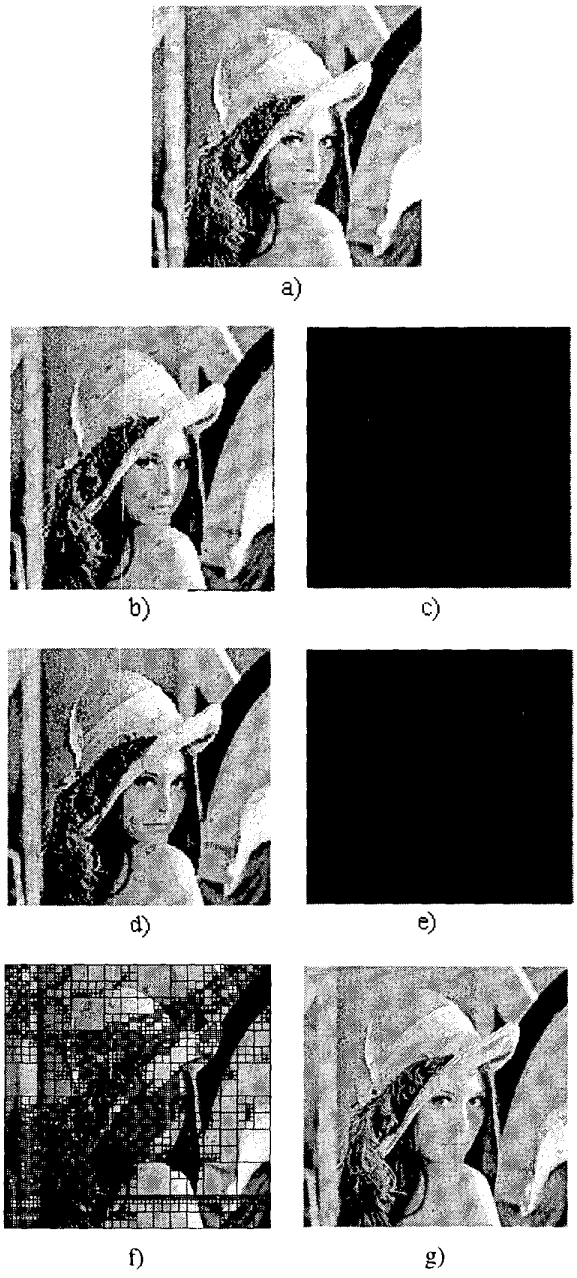

Figure 2: Results of computer simulations on the standard color image *Lena*. a) Original image *Lena* 512×512×24; b) Reconstruct image by *FHCBC* coding; c) Difference image between images a) and b); d) Reconstructed image by *SFC* coding; e) Difference image between images a) and d); f) Result of quadtree partition by *HCBC*; g) Composite image by *HCBC*.

The most important index to evaluate a compression method is the Compression Ratio (CR). It is defined as the total number of bits needed to store the original image by the total amount of bits required to store the compressed image. From Table 2, one can see that the compression ratio of *FHCBC* is higher than that of *SFC*. This can be qualitatively understood in the following way, for each range block, *SFC* must memorize three transformations, each one for a different color plane. On the other hand, two types of information need to be memorized by *FHCBC*, one is a transformation for each range block and the other is the mean values of trichromatic coefficients of a partitioned block. Thus, if the number of blocks obtained by the quadtree partition of *HCBC* is not significantly greater than the number of range blocks of the fractal coding, the compression ratio of *FHCBC* is, necessarily, higher than that of *SFC*. Specifically, we get CR_{SFC}: $CR_{\text{FBHCBC}} \approx 2 : 3$, when the number of blocks partitioned by *HCBC* is almost equal to the number of range blocks.

	Lena 512×512×24		
	CR	PSNR	Time (min)
SFC	21.94	30.69 (R) 29.78 (G) 30.58 (B) 30.35 (M)	624
FHCBC	36.34	29.01 (R) 28.34 (G) 30.34 (B) 29.13 (M)	204

Table 2: CR, PSNR and compression time for each experiment

In order to evaluate the fidelity between the original image and the decoded image, we calculate the *Peak-Signal-to-Noise-Ratio* (*PSNR*) for each method. The *PSNR* is defined by

$$PSNR_R = 10\log_{10}\left(\frac{255\times255}{(O_R - O_R')^2}\right) \qquad (9)$$

$$PSNR_G = 10\log_{10}\left(\frac{255\times255}{(O_G - O_G')^2}\right) \qquad (10)$$

$$PSNR_B = 10\log_{10}\left(\frac{255\times 255}{(O_B - O_B')^2}\right) \qquad (11)$$

where $PSNR_R$, $PSNR_G$, $PSNR_B$ are *PSNR*s for each one of the red, green and blue color plane. O_R, O_G, O_B are color planes of the original images and O'_R, O'_G, O'_B are the color planes of the reconstructed image. Here, each *PSNR* is the same as used in [11, 12, 16] for gray-level images. The mean *PSNR* among three-color planes is:

$$PSNR_M = \frac{\mathrm{PSNR}_R + \mathrm{PSNR}_G + \mathrm{PSNR}_B}{3} \qquad (12)$$

The *PSNR*s calculated during the experiments are shown in Table 2 too. From the table, one can see that the *PSNR*s of *FHCBC* and *SFC* have the same level, implying that the fidelity of *FHCBC* is not better than that of *SFC*. However, the encoding time of *FHCBC* is significantly shorter than *SFC*. This is due to the fact that the quadtree partitioning process (*HCBC*) of *FHCBC* is almost immediate, and hence, the encoding time is decided just by the part of fractal coding. Fractal coding runs only once in *FHCBC*, but it must run three times in *SFC*, each one for a different color plane. Therefore, the time spent by *SFC* is approximately three times longer than that needed for *FHCBC*, and the results presented in Table 2 confirm this observation.

5 Conclusions

The main results obtained by this work can be summarized by the following items.

1) *FHCBC* can achieve higher compression ratio than *SFC*. This is because *SFC* only eliminates intra-color-plane redundancy, while *FHCBC* eliminates both intra-color-plane and inter-color-plane redundancy.
2) *FHCBC* takes shorter encoding time than *SFC*. The encoding time of *FHCBC* is approximately one-third of *SFC*.
3) *FHCBC* has the same level reconstruction quality as *SFC*. However, from the experiments on other images, we found that the reconstruction quality is not stable in both method. It is heavily dependent on the images to be encoded.

The general conclusion is: *FHCBC* has a better performance than *SFC*.

Although *HCBC* algorithm is developed and used, in this paper, to improve fractal color image compression, one can easily see that it is really independent from fractal coding, i.e., *HCBC* is a more general scheme for color image coding, which can be used to work with fractal image compression algorithm as well as with other different one-component image compression techniques. The reason why we choose fractal coding to work with lays on the coding time. In comparison with other image coding techniques, current fractal coding algorithm takes extreme long time to encode a gray-level image and clearly, longer computational time is needed to encode a color image. For fractal color image compression, usually there are two ways to treat this problem. One is to improve the fractal coding algorithm itself (for a review, see [15]), another way is to transform three-color planes into a one-component image by using the correlation among them. One can see that this work follows exactly the second idea. As a conclusion, more benefits on savings of the encoding time can be obtained when *HCBC* works with fractal coding rather than with other image coding algorithms.

In this work, a homogeneous block is obtained by the quadtree partitioning of *HCBC* if the variance of pixels' trichromatic coefficients is within a certain threshold value. However, if the variance does not satisfy the given threshold at the ultimate level (smallest block size) of a given partition, a relatively large loss may occur. This effect diminishes the reconstruction quality. In order to avoid this situation, we can further divide the trichromatic coefficient ratio (a three-value ratio in this work) into two, two-value ratios and a multi mean value scheme has to be devised to get a better approximation.

References

[1] Barnsley M. F., "Fractals everywhere", *Academic Press, Inc.*, 2nd Edition, 1993.

[2] Barnsley M. F. & Hurd, L. P., "Fractals Image Compression", *AK Peters Ltd.*, 1993.

[3] Bell, S. B. M., "Fractals: a fast, accurate and illuminating algorithm", *Image and Vision Computing*, **13**(4), 253-257, 1995.

[4] Chellappa R. & Sawchuk A. A., "Digital image processing and analysis: Vol 1: Digital Image Processing", Silver Spring, MD, *IEEE computer Society Press*, 1985.

[5] Clarke R. J., "Digital compression of still images and video", London, *Academic Press*, 2nd printing, 1996.

[6] Fisher Y., "Fractal image compression", *SIGGRAPH'92 Course Notes*, 1992.

[7] Gersho A. & Gray R. M., "Vector quantization and signal compression", Norwell, Massachusetts, *Kluwer Academic Publishers*, 4th printing, 1995.

[8] Gonzalez R. C. & Woods R. E., "Digital Image Processing", Reading, MA, *Addison-Wesley*, 1992.

[9] Hart J. C., "Fractal image compression and recurrent iterated function systems", *IEEE Computer Graphics and Applications*, 25-33, 1996.

[10] Huang J. & Wang Y., "Compression of color facial image using feature correction two-stage vector

quantization", *IEEE Trans Image Processing*, **8**(1), 102-109, 1999

[11] Jacobs E. W., Fisher Y. & Boss R. D., "Image compression: A study of the iterated transform method", *Signal Processing*, **29**, 251-263, 1992.

[12] Jacquin A. E., "Image coding based on a fractal theory of iterated contractive image transformations", *IEEE Trans. Image Processing*, **1**(1), 18-30, 1992.

[13] Kim C.-S., Kim R.-C. & Lee S.-U., "A fractal vector quantizer for image coding", *IEEE Trans. Image Processing*, 7(11), 1598-1602, 1998.

[14] Klinger A. & Dyer C. R., "Experiment on picture representation using regular decomposition," *Computer Graphics and Image Processing*, **5**(1), 68-105, 1976.

[15] Li Z.-H., Soma N. Y. & Zhao L. (1999), "A review of fractal image compression", *Anais do WORKCOMP'99*, 23-28, São José dos Campos, Brasil.

[16] Lee C. K. & Lee W. K., "Fast fractal image block coding based on local variances", *IEEE Trans. Image Processing*, 7(6), 888-891, 1998.

[17] Moltedo L., Nappi M., Vitulano D. & Vitulano S., "Color image coding combining linear prediction and iterated function systems", *Signal Processing*, **63**, 157-162, 1997.

[18] Peitgen H. O., Jurgens H. & Saupe D., "Chaos and fractals", *Spinger-Verlag Inc.*, New York, 1992.

[19] Samet H. "The quadtree and related hierarchical data structure," *ACM Computing Surveys*, **16**(2), 188-260, 1984.