ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 6, No. 4, 2011, *pp*. 287-294

Speeding Up Fractal Image Compression Using DCT Descriptors

Dr. Loay E. George[1] and Nevart A. Minas[2]

[1]University of Baghdad/ College of Science
[2]Technical Institute of Kirkuk/ Computer Systems Department

(*Received January 25, 2011, accepted May 22, 2011*)

Abstract. The main problem in the Fractal image compression (FIC) method is the long encoding time. In order to speed up the encoding stage, an improved PIFS scheme is introduced in this paper; In the introduced PIFS scheme the input color image is, converted from RGB color space to YUV color space, and then the chromatic bands are down sampled by 2. The zero-mean range and domain blocks are classified using a block descriptor which is determined using a pair of low frequency DCT coefficients due to the energy packing properties of DCT. The DCT descriptors are also used to address the isometric state of each range and domain blocks. The coefficients of the optimal affine approximation are quantized, and then coded using differential pulse code modulation (DPCM) and shift coding. The conducted tests results on Lena's standard image indicated that the encoding time is reduced to (1 sec) while keeping the image quality above the acceptable level and no significant reduction in the compression ratio was occurred.

1. Introduction

The basic idea of fractal image compression technique was introduced by Barnsley [1] in 1980 according to the contractive mapping fixed-point theorem. In his proposed iterated function system (IFS), the contacted transform consisting of a sequence of affine transformations is applied to the entire image. Later Jacquin [2] proposed a partitioned IFS (PIFS) associated with a block-based automatic encoding algorithm where those affine transformations are applied to partitioned blocks. This scheme search process is extremely time-consuming. The main reason is that the similarity matching computation between range and domain blocks is complex. In response to this problem, an effective method is that the image blocks are classified before matching, so as to speed up the encoding process.

The major methods are used the grey statistical classification characteristics, such as variance or mean of image blocks in Jacquin[2] and Fisher[3]. Some methods used the image blocks' low frequency DCT coefficients [4]. Other methods classified the image blocks from different aspects, such as with inner product[5], with moment features[6].

This paper is organized as follows. In Section 2 the basic fractal image encoding method is described. Section 3 introduces, the proposed classification algorithm, in Section 4, the tests of the proposed method are presented and compared to that of other fast classification schemes. In the last section conclusions of the comparison are drawn.

2. Traditional PIFS Encoder

The basis steps of the encoding procedure could be summarized as: "the image, to be compressed, is partitioned into blocks; each block can be approximated by other image block after some scaling operations. The result block approximation is represented by sets of transform coefficients.

The traditional PIFS encoder loads the 24-bit bitmap color image, and decomposes the image data into three 2D-arrays (i.e., Red, Green and Blue). The color bands are converted from RGB color space to YUV space. The two chromatic bands (U and V) are down sampled by 2 to increase the compression gain. These two down sampled components with the Y component are passed, as three separate gray images, to PIFS encoder.

The implemented PIFS-mapping stage implies the following processes:

(a) Range Pool Generation: Each input color band (*WxH*) is partitioned into a non-overlapping blocks of size *m*, using fixed size blocks scheme with block size (*m=LxL*). The number of range blocks N_r is determined as follows:

$$N_r = [W/L][H/L] \tag{1}$$

(b) Domain Pool Generation: The domain array is generated by down sampling the input color-band (by 2) using averaging method. The domain array dimensions are calculated as follows:

$$H_h = H \setminus 2 \ , \quad W_h = W \setminus 2 \tag{2}$$

The domain array is partitioned into overlapping fixed size blocks; using certain jump step size (Δ) for horizontal and vertical shifts. The number of domain blocks in each row and column of the domain pool (i.e., N_{xd} and N_{yd}) and total number of domain blocks (N_d) are determined using the following equations:

$$N_{xd} = \left\lfloor \frac{W_h - L}{\Delta} \right\rfloor + 1, \quad N_{yd} = \left\lfloor \frac{H_h - L}{\Delta} \right\rfloor + 1 \tag{3}$$

$$N_d = N_{xd} N_{yd} \tag{4}$$

(c) PIFS mapping: In order to avoid the computational redundancy, some of involved coding parameters have been predetermined before entering the nested loops, among these parameters are:

(1) The quantization steps of both offset and scale parameters using the following equations:

$$Q_r = \frac{255}{2^{b_r} - 1} \ , \quad Q_s = \frac{s_{\max}}{2^{b_s - 1} - 1} \tag{5}$$

Where,

b_r is number of bits used to represent offset coefficient

b_s is number of bits used to represent sacle coefficient

S_{max} is the highest permissible value of scale coefficients(255).

(2) The average ($\overline{d}$) and variance ($m\sigma^2_d$) of each domain block (d_i):

$$\overline{d} = \frac{1}{m} \sum_{i=0}^{m-1} d_i \tag{6}$$

$$m\sigma_d^2 = \sum_{i=0}^{m-1} d_i^{\,2} - m\overline{d}^{\,2} \tag{7}$$

Where, *m* is the block size

(3) The reference coordinates of each domain block (x_d and y_d).

In the PIFS encoder, the search process implies that all the domain blocks (d_i) listed in the domain pool ($d_i, \ldots, d_n$) should be matched with the considered range block (r_i), to list out the optimal affine approximation (r_i'). The PIFS mappings are done between the range and domain blocks which have zero means. This kind of mapping is described by in the following equation:

$$r_i' = s(d_i - \overline{d}) + \overline{r_i} \tag{8}$$

where,

d_i is the corresponding pixel value in the domain block.

s is the scaling coefficient.

DOI https://doi.org/2024-JICS-22670 | Generated on 2025-04-15 09:43:55

$\bar{r}$ is the mean (or offset) range block.

$\bar{d}$ is the mean of domain block.

The best affine approximation (r_i') is determined using the following least error criteria:

$$\frac{\partial \chi^2}{\partial s} = 0 \tag{9}$$

where,

χ^2 is the total error between the range block (r) and its approximation (r')

s is the optimal scale value

Each domain block is transformed using a set of isometric transforms (i.e., rotation and flipping); the isometric mappings in the encoding process need more computational time to perform the extra matching processes.

The IFS set {consist of the scale(s), offset ($\bar{r}$), symmetry index (sym) of the domain block, besides to its coordinates (x_d,y_d)} is considered as the IFS code set that represents the range block (r) in terms of the domain block (d).

The following steps are used to determine the IFS code set:

(1) The offset ($\bar{r}$) of the range block:

$$\bar{r} = \frac{1}{m}\sum_{i=0}^{m-1} r_i \tag{10}$$

(2) Apply the uniform quantization to the average (offset):

$$I_r = round(\frac{\bar{r}}{Q_r}) \tag{11}$$

$$\tilde{r} = Q_r I_r \tag{12}$$

(3) Determine the scale coefficients and then bound it within the range [-S_{max}, S_{max}].

$$s = \frac{\sum_i r_i d_i - m\bar{d}\bar{r}}{\sum d_i^{\,2} - m\bar{d}^{\,2}} \tag{13}$$

$$s = \begin{cases} -s_{\max} & if\ s < -s_{max} \\ s & if\ -s_{max} \le s \le s_{\max} \\ s_{\max} & if\ s > s_{max} \end{cases} \tag{14}$$

The bounded scale is then quantized:

$$I_s = round\left(\frac{s}{Q_s}\right), \quad \tilde{s} = Q_s I_s \tag{15}$$

(4) The total error (i.e. Chi square (χ^2)) between the actual values of the range block elements and the corresponding approximate values produced by IFS mappings:

$$\chi^2 = \sum_{i=0}^{m-1} \left[(r_i - \tilde{r}) - \tilde{s}(d_i - \bar{d}) \right]^2 \tag{16}$$

(5) The registered χ^2 is compared with the permissible level of error (i.e. threshold value ε) to register the optimal set $(s, \bar{r}, sym, x_d, y_d)_{\text{opt}}$.

(d) The optimal affine approximation set are then coded using differential pulse code modulation (DPCM) and shift coding.

At decoding stage, the approximations could be performed several times, starting with any random image, till reaching the fixed point (i.e., attractor state) image".

3. Usage of DCT Block Indexing in Modified PIFS Encoder

In the enhanced PIFS coding stage, the range and domain pools are generated first, and then the input band is resized before partitioning it into fixed blocks of length (L) to be set as multiples of the block length. This is done by inserting some additional lines and/or columns, with equal spaces between each other, to the images in order to make the width and height of the band as multiple of block length, and they must be removed at the end of the PIFS decoding.

The energy compaction property of the DCT block transform is utilized in this introduced PIFS scheme. Most of the energy of image blocks is hold by the low frequency coefficients due to DCT energy packing property. So, the energy decreases when moving to high frequency part of the DCT domain, taking into consideration that this part of coefficients holds the fine shape information. The even DCT coefficients (such as, C(2,0), C(0,2), …) are invariant to block reflection (i.e., when reflecting the block horizontally or vertically the values of these coefficients will be the same as without reflection). For this reason, its use in this work was ignored, so only the coefficients {C(i,0) or C(0,i)} with i= {1,3} were used to determine the DCT based descriptors.

The low DCT coefficients {i.e., C(i,0) and C(0,i); where i=1 or 3} of each domain block (and later, for each range block) of size (L×L) are calculated using the following equations:

$$C(i,0) = \sum_{y=0}^{L-1} \sum_{x=0}^{L-1} c(x,y) \cos\left(\frac{i\pi}{2L} (2x+1) \right) \tag{17}$$

$$C(0,i) = \sum_{y=0}^{L-1} \sum_{x=0}^{L-1} c(x,y) \cos\left(\frac{i\pi}{2L} (2y+1) \right) \tag{18}$$

Where,

i ∈ {1,3} and c(x,y) is the domain or range element.

Table .1: The truth table for the eight blocks classes

Block's Class Index	Boolean Criteria		
	$\lvert C(i,0)\rvert \geq \lvert C(0,i)\rvert$	$C(i,0) \geq 0$	$C(0,i) \geq 0$
0	T	T	T
1	T	T	F
2	T	F	T
3	T	F	F
4	F	T	T
5	F	T	F
6	F	F	T
7	F	F	F

The two DCT coefficients i.e., {C(i,0) and C(0,i)} for each domain and range block are used to predict

the isometry index of the block using the Boolean criterion shown in table (1).

The required symmetry process on the domain block is determined by comparing the symmetry indexes of the range and matched domain block using the lookup table (2) instead of trying the 8 isometric operations to find the best possible match with the range block.

Beside the isometry index, another block descriptor is introduced to classify the domain and range blocks; it is also based on the two DCT descriptors C(i,0) and C(0,i). This descriptor is called DCT-based block descriptor (*R*). Instead of testing all domain blocks to find the best approximate for each range block, only the domain blocks have similar DCT-based block descriptor (*R*) to that of the range block, are tested to find the optimal IFS code. So, the descriptor (*R*) is used as

a classifier to classify the domain and range blocks. R Factors should be affine invariant, such that "If two blocks (range and domain) satisfy the contractive affine transform then their factors (R_d and R_r) should have similar values; this doesn't means that any two blocks have similar R factors are necessarily similar to each other" [6].

Table 2. The required symmetry operation to convert the block's isometry status

		New Block State Index(domain)							
		0	1	2	3	4	5	6	7
Old Block State Index (range)	0	0	6	4	2	5	1	3	7
	1	6	0	2	4	1	5	7	3
	2	4	2	0	6	3	7	5	1
	3	2	4	6	0	7	3	1	5
	4	5	1	3	7	0	6	4	2
	5	1	5	7	3	6	0	2	4
	6	3	7	5	1	4	2	0	6
	7	7	3	1	5	2	4	6	0
0≡ No Operation; 1≡ Rotation_90; 2≡ Rotation_180; 3≡ Rotation_270; 4≡ Reflection; 5≡ Reflection+ Rotation_90; 6≡ Reflection+Rotation_180; 7≡ Reflection+Rotation_270;									

In this paper, two types of *R*-descriptors have been tested (i.e., the ratio descriptor, and the relative difference). For each type, the two sets of DCT coefficients, {i.e., C(i,0) and C(0,i)} were taken into consideration. So, for each type of *R*-descriptors two possible descriptors were determined. For the first type (i.e., ratio descriptor) one of the following two descriptors were used:

$$FR1 = \begin{cases} \left| NoBin\frac{C_{01}}{C_{10}} \right| & if \ \ |C_{10}| \geq |C_{01}| \\ \left| NoBin\frac{C_{10}}{C_{01}} \right| & if \ \ |C_{01}| > |C_{10}| \end{cases} \tag{19}$$

$$FR3 = \begin{cases} \left| NoBin\frac{C_{03}}{C_{30}} \right| & if \ \ |C_{30}| \geq |C_{03}| \\ \left| NoBin*\frac{C_{30}}{C_{03}} \right| & if \ \ |C_{03}| > |C_{30}| \end{cases} \tag{20}$$

While, for the second type (i.e., relative difference) one of the following two descriptors were used:

$$SR1 = \left| NoBin * \frac{C_{01}^{\ 2} - C_{10}^{\ 2}}{C_{01}^{\ 2} + C_{10}^{\ 2}} \right| \tag{21}$$

$$SR3 = \left| NoBin * \frac{C_{03}^{\ 2} - C_{30}^{\ 2}}{C_{03}^{\ 2} + C_{30}^{\ 2}} \right| \tag{22}$$

The domain blocks are sorted in an ascending order according to their R-descriptor values to speed up the matching task between range and domain blocks. The possible value of all R-descriptor values is specified from the range [0, N_{Bins}]. If the best match between domain and range blocks in the exact class to find error less than the predefined threshold ε_1 is failed, then the matching should be restarted with domain blocks belonging to neighbor classes whose R factors lay within the range of values $[I_r \mp WinSiz]$. The matching error between these domain blocks and the range block is checked using other predefined error (ε_2) as its value is higher than the first predefined error (ε_1).

4. Test Results

The proposed methods have been tested by using color Lena image (256x256 pixels, 24 bits); the encoding parameters were set as follows:

1. The block length was 5 with jump step 1.
2. The maximum permissible scale (S_{max}) was 3.
3. The number of bins (N_{Bins}) of the DCT descriptor was 100 and search window size (*WinSiz*) was 1.
4. Permissible error threshold value (ε_1) was 1.
5. Permissible error threshold value (ε_2) was 1.2.
6. The number of quantization bits of the offset coefficients (*rNoBit*) was 8, and of the scale coefficients (*sNoBit*) was 6.

The proposed system was established using Visual Basic 6.0 and implemented on a Dell PC, equipped with Pentium Dual CPU and 1.60GHz processor. The encoding time was 1.02 second, PSNR was 31.66dB, and the obtained compression ratio was 13.533.

In the figures listed in this section, the following notations are used:

1. Trad: Denotes the traditional PIFS encoder.
2. Sym: Denotes the PIFS encoder that based on the symmetry descriptor only.
3. FR1, FR2, SR1, and SR2: Denotes the enhanced PIFS encoders that based on symmetry descriptor with the block descriptor defined by equations (19-22) respectively.

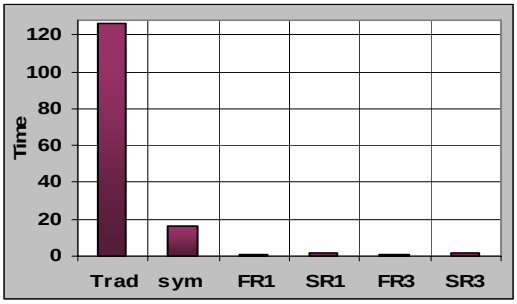

Fig. 1: The encoding time of the six PIFS schemes

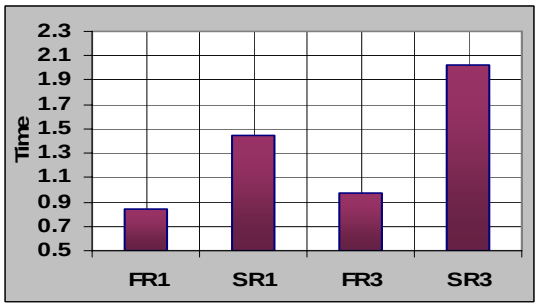

Fig. 2: The encoding time of the enhanced PIFS schemes

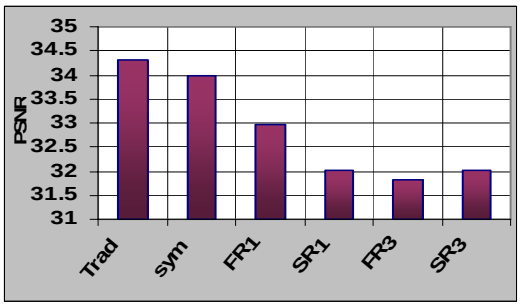

Fig .3: The attained PSNR of the six PIFS schemes

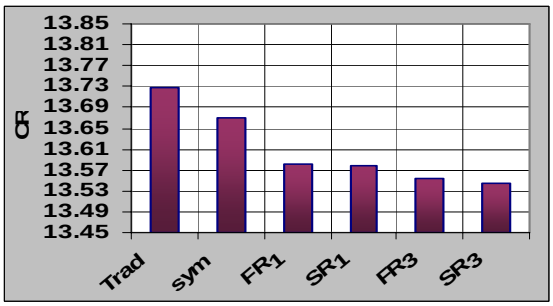

Fig. 4: The attained CR of the six PIFS schemes

The test results shown in figures (1) and (2) shows that the traditional system is the slower PIFS encoding scheme is decreased by 8 with little degradation in PSNR with the scheme that based on the

symmetry descriptor only. The enhanced PIFS schemes are relatively fast. The schemes FR1 or FR3 take less compression time than the SR1 or SR3schemes. Also, the descriptors based on C(1,0) and C(0,1) coefficients (i.e. FR1 and SR1) lead to compressed images with a quality higher than that obtained using C(3,0) and C(0,3) coefficients (i.e. FR3 and SR3).

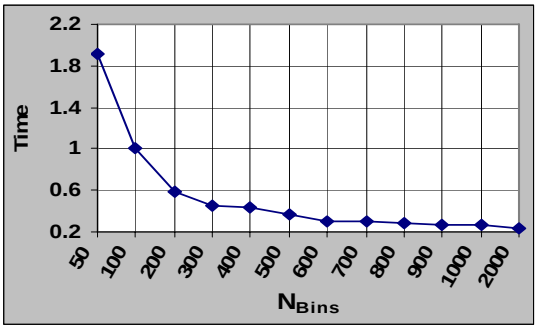

Fig. 5: The effect of N_{Bins} on the encoding Time

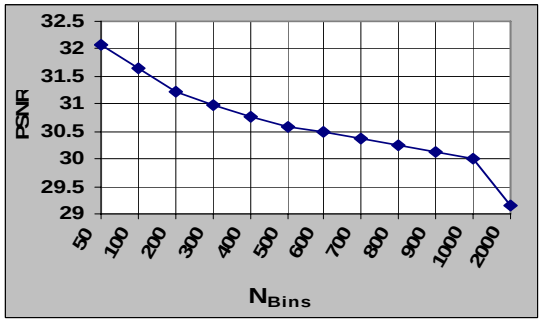

Fig. 6: The effect of N_{Bins} on the encoding Time

Table. 3: The comparison results of our proposed PIFS scheme and of some previous compression Systems (all applied on Lena image 256x256)

Research	Image type	Time (Sec)	PSNR (dB)	CR
[2]	Grey Scale	142	29.95	13.25
[7]	Color(8bit)	60	32	15
[6]	Grey scale	7	29	4.57
[9]	Grey scale	11	30.89	13.87
[8]	Color (24bit)	65	29.05	7.48
[4]	Grey scale	19.09	27.71	9.23
our	Color(24bit)	1.02	31.66	13.533

The test results in the figures (5) and (6) led us to consider N_{Bins} =100 as the best selection, because it leads to good compromised compression results. Table (3) summarizes between the attained results by the proposed compression system and those presented in other PIFS systems.

5. Conclusions

The analysis of the conducted tests has stimulated that the energy packing attribute of DCT coefficients is very useful in classifying the domain and range blocks and isometric process prediction, by using only the low frequency DCT coefficients. Using the color model YUV leads to good compression ratio when it is compared with the case of using RGB. Further work is needed to enhance the DCT filter method by using hierarchal partition schemes (*Quadtree* or *Horizontal-Vertical partition*) to attain higher compression ratios. Non uniform quantized classification descriptor can be used to address the domain and range blocks.

6. References

[1] M. F. Barnsley. *Fractals Everywhere.* Academic Press, 1993.

[2] A. E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformation**.** *IEEE, Image Processing*. 1992, **1**(1): 18-30.

[3] Y. Fisher. Fractal Image Compression. *SIGGRAPH Course Notes*. 1992.

[4] Duh D.J., Jeng J.H., and Chen S.Y.Speed Quality Qontrol for Fractal Image Compression**.** *Imaging Science Journal*. 2008, **56**(2): 79-90(12).

[5] MA Yan & LI Shun-bao. Fast fractal image compression algorithm based on inner product. *Computer Engineering*. 2003, **29**(2): 25-27.

[6] George L. E. Fast IFS Coding for Zero-Mean Image Blocks Using Moments Indexing Method*. Journal of science*.

2006, **6**(1): 8-14.

[7] Grebenik V. V. *Fast Hybrid Fractal Image Compression Algorithm for Color Images***.** IEEE. 1998, **1**: 804-806.

[8] Thakur N. V. Color Image Compression with Modified Fractal Coding on Spiral Architecture**.** *Journal of Multimedia*. 2007, **2**(4): 55-66.

[9] Jin-shu H. Fast Fractal Image Compression by Pixel Peaks and Valleys classification**.** *Journal of Communication and Computers*. 2007, **4**(3).