Electronic Letters on Computer Vision and Image Analysis 5(1):30-40, 2005

A Fast Fractal Image Compression Method Based Entropy

M. Hassaballah [*] , M.M. Makky [*] and Youssef B. Mahdy [+]

**Mathematics Department, Faculty of Science,South Valley University, Qena, Egypt*
+ Faculty of Computers & information, Assuit University, Assuit, Egypt

Received 9 December 2004; accepted 22 February 2005

Abstract

Fractal image compression gives some desirable properties like resolution independence, fast decoding, and very competitive rate-distortion curves. But still suffers from a (*sometimes very*) high encoding time, depending on the approach being used. This paper presents a method to reduce the encoding time of this technique by reducing the size of the domain pool based on the Entropy value of each domain block. Experimental results on standard images show that the proposed method yields superior performance over conventional fractal encoding.

Key Words: Fractal Image Compression, Complexity Reduction, Entropy.

1 Introduction

With the ever increasing demand for images, sound, video sequences, computer animations and volume visualization, data compression remains a critical issue regarding the cost of data storage and transmission times. While JPEG currently provides the industry standard for still image compression, there is ongoing research in alternative methods. Fractal image compression [1,2] is one of them. It has generated much interest due to its promise of high compression ratios at good decompression quality and it enjoys the advantage of very fast decompression. Another advantage of fractal image compression is its multi-resolution property, i.e. an image can be decoded at higher or lower resolutions than the original, and it is possible to "zoom-in" on sections of the image. These properties made it a very attractive method for applications in multimedia: it was adopted by Microsoft for compressing thousands of images in its Encarta multimedia encyclopaedia [3].

Despite of all the above properties of fractal image compression, the long computing in the encoding step still remains the main drawback of this technique. Because good approximations are obtained when many domain blocks are allowed, searching the pool of domain blocks is time consuming. In other word, consider an N x N image and n x n range blocks. The number of range blocks is $(N/n)^2$, while the number of the domain blocks is $(N-2n+1)^2$. The computation of best match between a range block and a domain block is $O(n^2)$. Considering n to be constant, the computation of complexity search is $O(N^4)$.

Correspondence to: mhasaballah@yahoo.com or mah220763@hotmail.com

Recommended for acceptance by Maria Petrou
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

Several methods have been proposed to overcome this problem. The most common approach for reducing the computational complexity is the classification scheme. In this scheme range and domain blocks are grouped in classes according to their common characteristics. In the encoding phase, only blocks belonging to the same class are compared, thus saving a lot of computation while keeping the performance in terms of image quality quite close to that of exhaustive search. Jacquin [2] proposed a discrete feature classification scheme based on Ramamurthi and Gersho approach [4]. The domain blocks are classified according to their perceptual geometric features. Only three major types of block are differentiated: shade blocks, edge blocks, and midrange blocks. In the Fisher's classification method [5], a given image block is divided into four quadrants. For each quadrant, the average and the variance are computed. According to certain combination of these values, 72 classes are constructed. This method reduces the searching space efficiently. However, it required large amount of computations and, the arrangement of these 72 classes is complicated.

In clustering methods [6,7] the domain blocks are classified by clustering their feature vectors in Voronoi cells whose centers are designed from the test image or from a set of training images. For each range block, matches are sought in the neighboring classes only. Another discrete feature classification based on mean was proposed by Hurtgen and Stiller [8]. The feature vector is constructed by comparing the sub-block mean of each quadrant to the block's mean. In this approach the search area for a domain block is restricted to a neighborhood of the current range. All the above approaches can only reduce the factor of proportionality in O(N) the time complexity for a search in the domain pool, where N is the size of the domain pool.

A different approach is to organize the domain blocks into a tree- structure, which could admit faster searching over the linear search. This approach is able to reduce the order of complexity from O(N) to O(log N). The idea of tree-structured search to speed up encoding has long been used in the related technique of Vector Quantization [9]. Caso et al. [10] and Bani-Eqbal [11] have proposed formulations of tree-search for fractal encoding.

In the feature vector approach introduced by Saupe in [12,13] a small set of d real-valued keys is devised for each domain which make up a d-dimensional feature vector. These keys are carefully constructed such that searching in the domain pool can be restricted to the nearest neighbors of a query point, i.e., the feature vector of the current range. Thus the sequential search in the domain pool is replaced by multi-dimensional nearest neighbor searching, which can be run in logarithmic time. Unfortunately, the feature vector dimension is very high, *i.e*. equal to the number of pixels in the blocks. This limits the performance of this approach as the multi-dimensionality search algorithms. Moreover large amounts of memory are required. Some attempts to solve this problem are presented in [14].

Complexity reduction methods that are somewhat different in character are based on reducing the size of the domain pool. Jacobs et al.'s method uses skipping adjacent domain blocks [15]. Monro [16] localizes the domain pool relative to a given range based on the assumption that domain blocks close to range block are well suited to match the given range block. Saupe's Lean Domain Pool method discards a fraction of domain blocks with the smallest variance [17]. The latest survey on the literature may be found in [18-20].

In this paper a new method to reduce the encoding time of fractal image compression is proposed. This method is based on removing the domain block with high entropy, ε from the domain pool. In this way, all the useless domains will be removed from the pool achieving a more productive domain pool. The proposed method can be extended to speed up the hybrid fractal coders and improve their performance.

The rest of this paper is organized as follows. Section 2, briefly describes fractal image coding and the baseline algorithm. In Section 3, definition of entropy and using it in the proposed method to reduce the encoding time of fractal image compression is presented, followed by experimental results and discussion in Section 4. The conclusions of the present work are summarized in Section 5.

2 Fractal Image Coding

2.1 Principle of Fractal Coding

In the encoding phase of fractal image compression, the image of size N x N is first partitioned into non-overlapping range blocks R_i, $\{ R_1, R_2, ... R_p \}$ of a predefined size BxB. Then, a search codebook (domain

pool Ω) is created from the image taking all the square blocks (domain blocks) D_j, $\{D_1, D_2, ...D_q\}$ of size 2Bx2B, with integer step L in horizontal or vertical directions. To enlarge the variation, each domain is expanded with the eight basic square block orientations by rotating 90 degrees clockwise the original and the mirror domain block. The range-domain matching process initially consists of a shrinking operation in each domain block that averages its pixel intensities forming a block of size BxB.

For a given range R_i, the encoder must search the domain pool Ω for best affine transformation w_i, which minimizes the distance between the image R_i and the image $w_i(D_i)$, (*i.e.* $w_i(D_i) \approx R_i$). The distance is taken in the luminance dimension not the spatial dimensions. Such a distance can be defined in various ways, but to simplify the computations it is convenient to use the Root Mean Square RMS metric. For a range block with n pixels, each with intensity r_i and a decimated domain block with n pixels, each with intensity d_i the objective is to minimize the quality

$$E(R_i, D_i) = \sum_{i=1}^{n} (sd_i + o - r_i)^2 \tag{1}$$

Which occurs when the partial derivatives with respect to s and o are zero. Solving the resulting equations will give the best coefficients s and o [5].

$$s = \frac{n\sum_{i=1}^{n} d_i r_i - \sum_{i=1}^{n} d_i \sum_{i=1}^{n} r_i}{n\sum_{i=1}^{n} d_i^2 - (\sum_{i=1}^{n} d_i)^2} \tag{2}$$

$$o = \frac{1}{n}(\sum_{i=1}^{n} r_i - s\sum_{i=1}^{n} d_i) \tag{3}$$

With s and o given the square error is

$$E(R_i, D_i)^2 = \frac{1}{n}\left[\sum_{i=1}^{n} r_i^2 + s(s\sum_{i=1}^{n} d_i^2 - 2\sum_{i=1}^{n} d_i r_i + 2o\sum_{i=1}^{n} d_i) + o(on - 2\sum_{i=1}^{n} r_i)\right] \tag{4}$$

If the denominator in Eq. (2) is zero, then s =0 and $o = \frac{1}{n}\sum_{i=1}^{n} r_i$.

The parameters that need to be placed in the encoded bit stream are s_i, O_i, index of the best matching domain, and rotation index. The range index *i* can be predicted from the decoder if the range blocks are coded sequentially. The coefficient s_i represents a contrast factor, with $|s_i| \leq 1.0$, to make sure that the transformation is contractive in the luminance dimension, while the coefficient o_i represents brightness offset.

At decoding phase, Fisher [5] has shown that if the transforms are performed iteratively, beginning from an arbitrary image of equal size, the result will be an attractor resembling the original image at the chosen resolution.

2.2 Baseline fractal image coding algorithm

The main steps of the encoding algorithm of fractal image compression based on quadtree partition [5] can be summarized as follows:

Step 1: Initialization (domain pool construction)

Divide the input image into N domains, D_j

For (j =1; j $\leq$ N; j ++)

Push D_j onto domain pool stack Ω

Step 2: Choose a tolerance level ℓ_C;

Step 3: Search for best matches between range and domain blocks

```
For ( i =1 ; i ≤ num_range ; i ++ )  {
    min_error = ℓ_C ;
    For ( j =1 ; j ≤ num_domain; j ++ ) {
        Compute s, o;
        If (0 ≤ s< 1. 0)
          If ( E(R_i , D_j ) < min_error) {
              min_error = E(R_i , D_j );
              best_domain[i] =j ; }
        }
     If  (min_error = = ℓ_C )
         Set R_i uncovered and partition it into 4 smaller blocks;
      Else
        Save_coefficients(best_domain, s, o);
}
```

In this algorithm, parameter ℓ_C settles the fidelity of the decoded image and the compression ratio. By using different fidelity tolerances for the collage error, one obtains a series of encodings of varying compression ratios and fidelities. For a range block if ℓ_C is violated for all the domain blocks, that is the range block is uncovered, the range block is divided into four smaller range blocks, and one can search for the best match domains for these smaller range blocks. At the end of **step 1** the domain pool Ω has N domain (*i.e.* all domains).

3 The Proposed Method

3.1 Entropy

Assume that there exists a set of events S={ x_1, x_2,... x_n }, with the probability of occurrence of each event $p(x_i) = p_i$. These probabilities, P={ p_1, p_2,... p_n }, are such that each $p_i \geq 0$, and $\sum_{i=1}^{n} p_i$ =1. The function,

$$I(x_i) = -\log p_i \tag{5}$$

is called the amount of self-information associated with event x_i. This function is a measure of occurrence of the event x_i. The function *I* focuses on one event at a time. In most situations, however, and certainly in the context of data compression, one has to look at the entire set of all possible events to measure

content over the entire set. An important concept introduced by Shannon is *entropy* associated with a set of events, which takes the form:

$$H(p_1, p_2, ...p_n) = H(s) = -\sum_{i=1}^{n} p_i \log p_i \quad (6)$$

Entropy can be defined as the average self-information that is, the mean (expected or average) amount of information for an occurrence of an event x_i. In the context of coding a message, entropy represents the lower bound on the average number of bits per input value. The function *H* has the following lower and the upper limits:

$$0 = H(1,0,0,...0) \le H(p_1 p_2, ...p_n) \le H(\frac{1}{n}, \frac{1}{n}, ...\frac{1}{n}) = \log n \quad (7)$$

In other words, if the events are equally likely, the uncertainty is the highest since the choice of an event is not obvious. If one event has probability 1 and the others probability of 0, the choice is always the same, and all uncertainly disappears.

3.2 The Entropy Based Encoded Algorithm

Eq. (1) is a full search problem and as mentioned previously is computationally intensive. One of the simplest ways to decrease encoding time of this full search problem is to decrease the size of the domain pool in order to decrease the number of domains to be searched. The proposed method reduces the encoding time of fractal image compression by performing less searches as opposed to doing a faster search, by excluding many of domain blocks from the domain pool. This idea is based on the observation that many domains are never used in a typical fractal encoding, and only a fraction of this large domain pool is actually used in the fractal coding. The collection of used domains is localized in regions with high degree of structure [17]. Figure (1) shows the domain blocks of size 8x8 that are actually used in the fractal code of Lena image. As expected the indicated domains are located mostly along edges and in the regions of high contrast of the image.

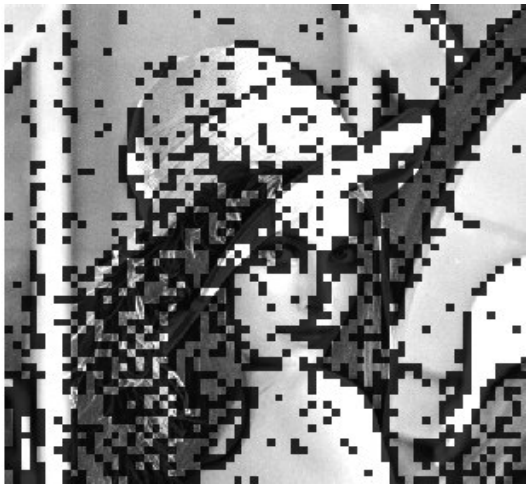

Fig. (1): Domains of size 8x8 that are used for fractal coding of 512x512 Lena are shown in black.

Analyzing the domain pool, there is a very large set of domain blocks in the pool with high entropy, which are not used in the fractal code. Thus, it is possible to reduce the search time by discarding a large fraction of high entropy blocks, which affect only a few ranges. For these ranges a sub-optimal domains with smaller entropy may be found. In this way, the domain pool is constructed from blocks with the lowest

entropy instead of all domains. In this case, the encoding time is heavily reduced by a priori discarding those domains from the pool, which are unlikely to be chosen for the fractal coding. Eq. (6) is used to calculate the entropy value for each domain block. According to this value a decision is taken to determine if this domain can become a part of the domain pool or not. A parameter ε will control the domain entropy value in the implementation, with ε being a quality parameter since it determines the size of the domain pool. The proposed method can only reduce the factor of proportionality in the O(N) complexity, where N is the domain pool size. But one can use the Tree approach [21] on the resulting efficient domain pool after removing all useless domain blocks, which is able to fundamentally reduce the order of encoding time from O(N) to O(log N).

The baseline algorithm mentioned above is modified in such a way that the domain pool Ω contains only domain blocks which have a certain entropy value. The main steps of the modified encoder algorithm of fractal image compression can be summarized as follows:

Step 1: Initialization (domain pool construction)

Choose parameter ε ;

Divide the input image into *N* domains, D_j

For (j =1; j ≤ *N*; j ++) {

Ent =entropy (D_j);

If (Ent ≤ ε)

Push D_j onto domain pool stack Ω }

Step 2: Choose a tolerance levels ℓ_C ;

Step 3: Search for best matches between range and domain blocks

```
For ( i =1 ; i ≤  num_range ; i ++ ) {
    min_error = ℓ_C ;
    For ( j =1 ; j ≤  num_domain; j ++ )  {
        Compute s, o;
        If (0 ≤ s< 1. 0)
          If ( E(R_i , D_j ) < min_error) {
              min_error = E(R_i , D_j );
              best_domain[i] =j ;          }
        }
     If  (min_error = = ℓ_C )
        Set R_i uncovered and partition it into 4 smaller blocks;
     Else
       Save_coefficients(best_domain, s, o);
}
```

At the end of **step 1** the domain pool has *num_domain* domain according to ε value.

4 Experimental Results and Conclusions

This section presents experimental results showing the efficiency of the proposed method. The performance tests carried out for a diverse set of well-known images of size 512x512 gray levels with 8bpp, on a PC with Intel Pentium III 750 MHz CPU and 128MB memory under windows 98 operating system using Visual C++6.0 programming language and the time is measured in seconds. Moreover, The scaling

coefficient (contrast) restricted to values between 0 and 1 in order to avoid searching domain pool twice (*i.e.* allowed only positive scaling factors in the gray level transformation). To ensure a compact encoding of the affine transformation, the value of contrast and brightness are quantized using 4 and 6 bits for contrast and brightness, respectively, hence the compression ratio is 95% and 89% for fixed range size and quadtree partitions respectively. This study focuses on the implementation issues and presents the first empirical experiments analyzing the performance of benefits of entropy approach to fractal image compression. First, the performance of the proposed method with fixed range size partition is examined. The size of the range block is set to be 8x8 pixel, and hence the domain size is 16x16, with domains overlapping *i.e.* the domain step L (distance between two consecutive domains) is divided by 4. The result is shown in table (1). Second, the same experiment is carried out with well-known technique of quadtree partitioning, allowing up to three quadtree levels. The average tolerated error between the original image and its uncompressed version is set to be ℓ_C =2.0. The results are shown in table (2).

The results in tables (1) and (2) show that the encoding time scales linearly with ε . This is expected since the major computation effort in the encoding lies in the linear search through the domain pool. For the case without domain pool reduction ε =0 (full search) there is no savings in the encoding time as shown in Fig. (2). Also, in the case of fixed range size partition the loss in quality of the encoding in terms of fidelity is larger than for quadtree partition. This is caused by the fact that some larger range can be covered well by some domains, which are removed from the domain pool at larger values of ε (e.g. $\varepsilon \geq 2.5$). As a consequence some of these ranges are subdivided and their quadrants may be covered better by smaller domains than the larger range.

This simple entropy approach leads to very significant savings in encoding time and is similar to the approach used in [5]. With fixed range size partition, it causes only negligible or no loss in the equality of image, thereby reducing by 2 the encoding time (at ε =2.5). In the quadtree case, when ε =3.8 the encoding time of Hill image is 80.03 sec while the PSNR is 38.63 dB. For comparison, the baseline (full search) required 1304.45 sec and the PSNR achieved is 39.14 dB. This represented a speed up factor of over 16 at the expense of a slight drop of PSNR of 0.51 dB. Generally, the speed-up in terms of actual encoding time is almost 7 times while the loss in quality of the image is almost 0.83 dB. This compares well with Saupe's Lean Domain Pool Method, which achieved comparable speedup of 8.9 at the expense of a drop of 1.7dB for Lena image [18]. Also, with Chong Sze [22], which achieved a speed-up of 9.3 with 0.87 dB loss for the same image. Figures (3), and (4) show examples of reconstructed images, which were encoded using the entropy method with fixed range size and quadtree partitions.

Finally, the proposed method seems to be applicable in situations where extremely fast encodings are desired and some quality degradation can be tolerated (*e.g.* by choosing $\varepsilon \geq 3.8$). For example, Fig. (5) shows that the Peppers image is coded in 2.93s with a quality of 33.56 dB (while the full search encoding time is 749.63s with a quality of 40.50 dB). This means that the image fidelity is still acceptable at least for some applications where high fidelity is not an absolute requirement.

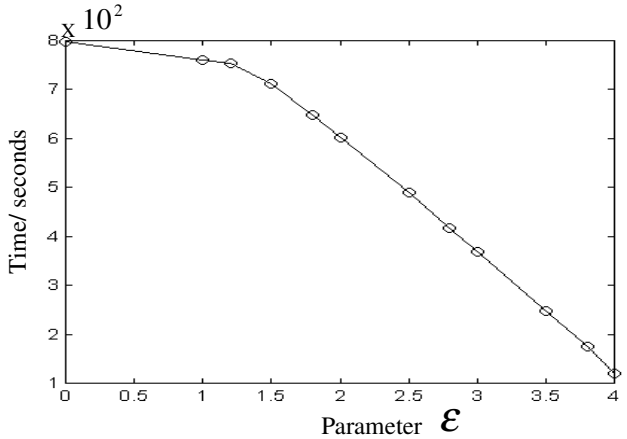

Fig. (2): Encoding time versus epsilon for Lena 512x512 image.

Table 1: Performance of fixed range size coding of four test images.

ε	Lena		Peppers		Boat		Hill	
	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR
0	124.99	36.34	119.65	37.51	122.59	28.31	119. 56	34. 93
1	106.41	36.34	111.81	37.49	100.20	28.28	115.86	34.93
1.2	100.34	36.32	110.08	37.46	92.26	28.25	105.86	34.92
1.5	87.68	36.30	98.37	37.39	79.04	28.18	86.06	34.86
1.8	76.02	36.23	86.54	37.34	64.06	27.99	65.67	34.78
2	68.56	36.12	74.99	37.29	58.77	27.98	51.77	34.65
2.5	50.36	35.98	55.01	37.23	45.42	27.76	21.67	34.37
2.8	40.45	35.95	40.03	37.08	39.55	27.73	12.38	34.23
3	32.89	35.73	31.99	36.98	35.50	27.62	8.67	33.89
3.5	18.26	35.36	13.69	36.56	22.86	27.38	5.38	33.46
3.8	10.82	34.83	6.03	35.59	15.93	27.22	4.52	33.16
4	5.80	34.39	3.01	34.50	11.56	26.89	4.16	33.15

Table 2: Performance of quadtree partition coding of four test images.

ε	Lena		Peppers		Boat		Hill	
	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR
0	797.78	40.66	749.63	40.50	1151.26	34.23	1304.45	39.14
1	760.91	40.65	745.91	40.51	1144.31	34.12	1323.31	39.12
1.2	753.86	40.65	743.14	40.52	1201.96	34.27	1313.14	39.12
1.5	712.72	40.64	746.93	40.51	1180.62	34.23	1318.08	39.09
1.8	647.72	40.59	736.98	40.50	981.00	34.2	1192.09	39.09
2	601.98	40.56	629.49	40.49	880.75	34.27	1062.43	39.05
2.5	489.06	40.48	553.58	40.45	632.66	34.28	677.08	38.96
2.8	417.90	40.39	477.20	40.49	541.81	34.17	442.98	38.85
3	367.91	40.36	398.01	40.43	494.21	34.11	317.13	38.75
3.5	246.4	40.12	236.96	40.32	408.76	33.79	121.95	38.69
3.8	174.00	39.88	127.97	39.98	327.58	33.71	80.03	38.63
4	120.18	39.83	64.09	39.71	250.91	33.53	65.82	38.51

Fixed range size partition
Encoding time: 5.8s.
Quality: 34.39dB

Quadtree partition
Encoding time: 120.18s.
Quality:39.83 dB.

Fig. (3): Lena 512x512 image.

Fixed range size partition
Encoding time: 3.01s.
Quality: 34.50dB

Quadtree partition
Encoding time: 64.09s.
Quality:39.71dB.

Fig. (4): Peppers 512x512 image.

Fig. (5): Peppers 512x512 image encoded in 2.93s by the proposed method and the PSNR of the reconstructed image is 33.56dB.

5 Conclusions

In this paper a parameterized and non-adaptive version of domain pool reduction is proposed, by allowing an adjustable number of domains to be excluded from the domain pool based on the entropy value of the domain block, which in turn reduced the encoding time. Experimental results on standard images showed that removing domains with high entropy from the domain pool has little effect on the image quality while significantly reduces the encoding time. The proposed method is highly comparable to other acceleration techniques. Next step in our research is to use the proposed method to improve the speed of hybrid coders (gaining better results than JPEG) that are based on fractal coders and transform coders so as to improve their performance.

References

[1] M. Barnsley and A. Jacquin. Applications of Recurrent Iterated Function Systems to Images. SPIE Visual Communications and Image Processing, Vol. 1001,

[2] A. E. Jacquin. Image Coding Based on a Fractal Theory of Iterated Contractive Image Transform. IEEE trans. on Image Processing, Vol. 1, N

[3] M. Barnsley and L. Hurd. Fractal Image Compression. on Image Processing: Mathematical Methods and Applications. pp. 183-210, Clarendon Press, Oxford, 1997.

[4] B. Ramanurthi and A. Gersho. Classified Vector Quantization of Image. IEEE Trans. Communication, COM-34, Vol.11, pp. 1105-1115, 1986.

[5] Y. Fisher. Fractal Image Compression: Theory and Applications. Springer-Verlag, New York, 1994.

[6] R. Hamzaoui. Codebook Clustering by Self-Organizing Maps for Fractal Image Compression. In NATO ASI Conf. Fractal Image Encoding and Analysis, Trondheim, July 1995. Fractals, Vol. 5, Supplementary issue, April 1997.

[7] R. Hamzaoui and D. Saupe. Combining Fractal Image Compression and Vector Quantization. IEEE Trans. on Image Processing, 9(2), pp.197-208, 2000.

[8] B. Hurtgen and C. Stiller. Fast Hierarchical Codebook Search for Fractal Coding of Still Images. Proceeding of SPIE, Vol. 1977, pp. 397-408, 1993.

[9] L.M. Po and C.K. Chan. Adaptive Dimensionality Reduction Techniques for Tree-Structured Vector Quantization. IEEE *Trans*. on Communications, Vol. 42, No. 6, pp. 2246-2257, June 1994.

[10] G. Caso, P. Obrador and C.-C. Kuo. Fast Methods for Fractal Image Encoding. Proc. SPIE Visual Communication and Image Processing'95, Vol. 2501, pp.583-594, 1995.

[11] B. Bani-Eqbal. Enhancing the Speed of Fractal Image Compression. Optical Engineering, Vol. 34, No. 6, pp.1705-1710, June 1995.

[12] D. Saupe. Accelerating Fractal Image Compression by Multi-dimensional Nearest Neighbor Search. In Proc. Data compression Conference, March 28-30, 1995.

[13] D. Saupe. Fractal Image Compression via Nearest Neighbor Searching. In Conf. Proc. NATO ASI, Fractal Image Coding and Analysis Trondheim, July 1995.

[14] C.S. Tong and W. Man. Adaptive Approximation Nearest Neighbor Search for Fractal Image Compression. IEEE Trans. on Image Processing, 11(6), pp.605-615, 2002.

[15] E.W. Jacobs, Y. Fisher, and R.D. Boss. Image Compression: A study of the Iterated Transform Method. Signal Process, Vol. 29, pp. 251-263, 1992.

[16] D.M. Monro and F. Dudbridge. Approximation of Image Blocks. in Proc. Int. Conf. Acoustics, Speed, Signal Processing, Vol. 3, pp.4585-4588, 1992.

[17] D. Saupe. Lean Domain Pools for Fractal Image Compression. Proceedings IS&T/SPIE 1996 Symposium on Electronic Imaging: Science & Technology Still Image Compression II, Vol. 2669, Jane 1996.

[18] M. Polvere and M. Nappi. Speed-Up in Fractal Image Coding: Comparison of Methods. IEEE Trans. on Image Processing, Vol. 9, No. 6, pp.1002-1009, June 2000.

[19] B. Wohlberg and G. Jager. A review of the Fractal Image Compression Literature. IEEE Trans. on Image Processing, Vol. 8, No. 12, pp. 1716-1729, Dec. 1999.

[20] D. Saupe and R. Hamzaoui. Complexity Reduction Methods for Fractal Image Compression. In I.M.A. Conf. Proc. on Image Processing; Mathematical methods and applications, Sept., J.M. Blackedge (ed.), 1994.

[21] X. Gharavi-Alkhansari, and T.S. Huang. Fractal Image Coding Using Rate-Distortion Optimized matching Pursuit. Proc. *SPIE*, pp. 265-304, 1996.

[22] C.S. Tong, and M. Pi. Fast Fractal Encoding Based on Adaptive Search. IEEE Trans. on Image Proc.10, No.9, pp.1269-1277, Sept. 2001.