AN IMPROVED DOMAIN CLASSIFICATION SCHEME BASED ON LOCAL FRACTAL DIMENSION

JAYAMOHAN M.
Department of Computer Science, College of Applied Science, Adoor,
Kerala, India, 691523.
jmohanm@gmail.com

K. REVATHY
Department of Computer Science, University of Kerala,
Thiruvananthapuram, Kerala, 695581.
revathysrp@gmail.com

Abstract

In fractal image compression, most of the time during encoding is spent for finding the best matching pair of range-domain blocks. Different techniques have been analyzed for decreasing the number of operations required for this range-domain matching. Encoding time can be saved by reducing the domain search pool for each range block. Domain blocks can be classified based on local fractal dimension. Fractal dimension is being studied as a measure to analyze the complexity of image portions. This paper proposes application of height balanced binary search trees for storing domain information ordered in terms of the local fractal dimension. The approach is to prepare the domain pool dynamically, by comparing the fractal dimension of range block with that of the domains. Domains with fractal dimension in an interval, evenly covering the fractal dimension of range block alone are given for comparison. We use AVL trees to enlist the domains based on their fractal dimension. The domain pool is prepared at runtime. Since the tree organization is used in the preprocessing phase, the proposed method can be used with any algorithm for fractal compression.

Keywords*:* fractal image compression, fractal dimension, domain classification, AVL tree

1. Introduction

Fractal image compression has been widely accepted as the most promising technique among the alternates for transform coding techniques and wavelets. The constraint that keeps the technique from becoming a popular standard is the huge encoding time compared to other methods like JPEG, wavelet etc. The overhead expense in fractal encoding is due to the huge number of range-domain comparisons required to find a best matching pair.

The concept of fractals was first introduced by Benoit Mandelbrot [1]. Fractals are self-similar patterns. Irrespective of the scale in which a fractal image is viewed, it demonstrates the same amount of details in the same pattern in every part of it. Michael Barnsley introduced the idea of expressing an image as an attractor of a set of chaotic processes [2]. The system of contractive transformations which generates the attractor image has been called iterated function systems (IFS). Collage Theorem states that to find an IFS whose attractor looks like a given image, one must find a set of transformations- contraction mappings- of the images of the given set. This idea inspired researchers to focus on using fractal concepts to represent an image in a reduced form, as a collection of affine transformation coefficients.

But it was computationally infeasible to derive a contraction mapping that defines the entire image as a single collage. Later, Jacquin introduced the idea of partitioned IFS [3]. The image can be partitioned as non-overlapping blocks and each partition can be defined as an attractor of an IFS. The method has proven to be a feasible one.

The most attractive part of fractal compression is that it is free from Gibb's phenomenon [4]. If one can find a best set of affine transformations, the image reconstruction process cannot generate visible artifacts in the

image. Another promising feature of fractal techniques is its resolution independence. The image stored in compressed form can be reconstructed to any size we like, with the same level of details, whereas in other methods the decoder has to duplicate pixel values to scale it up.

In spite of all these unique advantages, what make the industry to rethink on fractal image compression scheme is the intensive computations required in image encoding. The computational complexity of fractal compression is very huge compared to other techniques. The function that eats most of the encoding time is the range-domain matching. Since it is computationally infeasible to obtain a single set of affine transformation for the entire image, the compression algorithms use partitioned iterative function systems. The image is divided into nonoverlapping blocks called ranges. Then the algorithm tries to find a larger image portion, called domain, which is similar to the given range block. The encoder has to search for all possible domain blocks. Here the domain blocks are allowed to be overlapping. So, each range block has to be compared with a large number of domain blocks to find the best match. For example, consider an image of size 256 x 256. Let the image be partitioned into range blocks of size 8x8. This results in 1024 blocks to be encoded. Now, if we choose the domain blocks to be twice larger than range blocks, it leads to 241 x 241 = 58081 domain blocks (since they can be overlapping). So, a plain algorithm requires each range block to be compared with these domains to get a best matching pair. Thus to decide on the 1024 range blocks, it requires the matching function to execute 59,474,944 times. In general, the time complexity of such an encoder will be $\Theta(n^4)$ for an *n* x *n* image.

The encoding time can be reduced by minimizing the number of domains required to match with each range block. Domain classification schemes based on various parameters have been investigated by researchers. [5, 6, 7]. The methods generally focus on selecting a subset from the set of domains, as the candidates for each range block. Reducing the number of domain blocks to be matched based on their local fractal dimension is one among such attempts [8].

2. Domain classification based on Fractal Dimension

Fractal dimension has been used as an efficient indicator of shape information and texture complexity of an image. There are various techniques for measuring fractal dimension of images. The most popular and straight forward method is box counting. According to the box counting theorem [2], the image is assumed to be covered with closed square boxes of size $1/2^n$. Let N(A) denote the number of boxes of side length $1/2^n$, which intersects the attractor. Then A has the fractal dimension D, if

$$D= \lim_{n\to\infty} \frac{\log (N(A))}{\log (2^n)}$$

The value of FD will be stable over a pure fractal image. But natural images are not having pure fractal nature. Hence if we estimate fractal dimensions of local areas of the image, it will be different from the FD of the entire image, and the results may vary depending on the local texture variations.

Sarkar and Choudhari [9] uses an improved approach to determine the fractal dimension of images, known as differential box-counting (DBC), taking into account the graylevel values in each box. The image is viewed as a terrain surface of graylevel plot. Let an *m x m* image be covered with boxes of size *b* x *b*. Let r = *m/b* be the ratio of partitioning. (If box size is half of the image, r = ½). Let *L* be the maximum graylevel and *l* be the minimum graylevel value of pixels in a box (i, j). Then the number of boxes is counted as

$$N_r = \sum n_r(i,j)$$

where $n_r(i,j) = L - l + 1$.

Conci & Campos has improvised the method with minor modifications in DBC [10]. When the method is used, we get the fractal dimension of graylevel images generally in [2.0,3.0].These works [9, 10] show that fractal dimension can be used to identify variations in images. Hence it can be used as a measure for texture similarity.

It has been investigated to use fractal dimension to improve the encoding speed of fractal image compression. According to the method proposed in [8], the domain blocks are classified into two pools based on their fractal dimension. To derive the contractive transformations, the domains of the selected pool only are

considered. The method proposed by A.Conci separates the domains into two pools, one with FD in the interval 2.0 <= FD < 2.5 and the other in 2.5 <= FD <= 3.0. The results shows that the classification based on fractal dimension is effective in improvising the encoding time without compromising the compression ratio or quality, compared to other existing fractal methods.

The method uses a brute-force approach to classify the domains. Two fixed partitions of the domain blocks are created before encoding process starts. The domain pools remain static throughout the compression phase. The fractal dimension of the candidate range block is not considered in preparing the set of domains competing for contractive transformations to it. For example, when the FD of range block is 2.50, domain blocks with FD less than 2.5 alone will be considered for matching. Besides, we cannot expect for an even distribution of local FD values in the interval [2.0, 3.0] in all cases. In the worst case, all domains can fall in the same pool (either in [2, 2.5) or [2.5, 3.0]). We propose an improvement in selecting the domain pool.

3. New method

Instead of dividing the entire domains at the middle of their FD value range, into two fixed pools, the domain pool can be selected dynamically for each range block. The domain blocks with their fractal dimension falling in an interval evenly covering the FD of the candidate range block alone will be pooled for matching. The fractal dimension of all the domain blocks will be computed before starting the encoding process. Though any method can be used for computing fractal dimension, we use the basic box counting method which is the most popular algorithm, for illustration purpose.

Inorder to reduce the time spend in finding a matching domain block, we propose to use AVL tree for holding the domains. AVL tree is height balanced binary search tree, which can be searched in O (log n) time in every case [11, 12, 13].

First the image is divided into range blocks of size *nxn* and domain blocks of size, say *2nx2n*. The local fractal dimension of each domain block is computed. The domain information is inserted into a height balanced binary search tree (AVL tree). This can be called a preprocessing operation. Inorder to find the best matching domain of each range block, the FD of the range block is computed first. An interval of FD values corresponding to the range FD is determined dynamically. Then an inorder traversal is initiated in the AVL tree starting from the node at the lower boundary of the set interval and the matching function is executed for domains in nodes upto the upper bound of the interval.

3.1 *Height balanced trees*

A binary tree is said to be height balanced if the balance factor of every node in the tree lies in [-1,0,1]. That is, the difference between the depth of left subtree and right subtree of any node cannot exceed 1. Adelson-Velskii and Landis introduced the operations for balancing a binary search tree [11]. These operations are known as AVL rotations. This allows building the tree in optimum number of levels. Thus searching in the tree can be achieved in minimum number of comparisons. The number of comparisons required to search for a node in a binary search tree is equal to the number of levels to reach the node. For height balanced binary search

trees, searching can be performed in *O (log n)* time, even for the worst case.

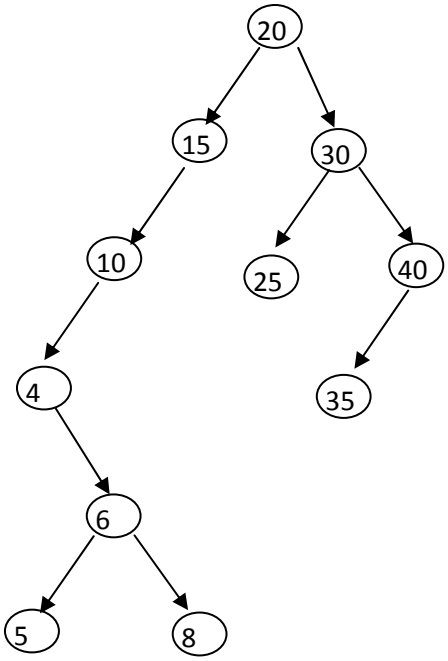

Fig.1. An unbalanced search tree. The number of comparisons required to reach a leaf node lies between [3, 6].

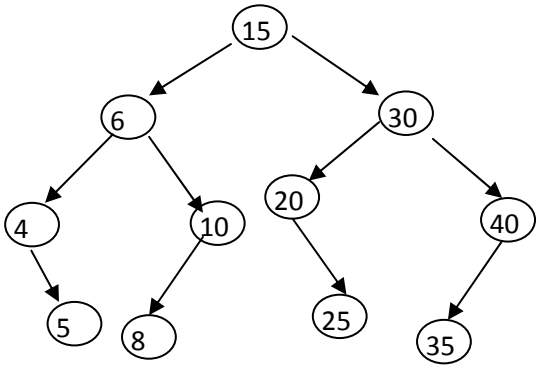

Fig.2.The search tree after balancing. The number of comparisons required to reach a leaf node is either 3 or 4.

When compared to self-balancing binary trees like red-black tree, AVL tree is the one that can be constructed with minimum height. The height of an AVL tree will be strictly less than 1.44 log (n+2) – 1, where n is the number of data nodes. Hence the time required for searching is O (log*n*).

The tree needs to be balanced when an insertion increases the balance factor to cross the given limits. Subtree rotations shall be performed to rebalance the tree. If the balance factor remains [-1, 0, 1] then rotations are not necessary. A maximum of O(log*n*) rotations need to be done in this regard.

The pointers to domains along with their FD values will be inserted in an AVL tree, ordered upon their FD. The fractal dimension of each domain block will be computed using box dimension method. Inorder to find a matching domain for each range block, we first compute the FD of the selected range. Then an interval is fixed that evenly covers the FD of range block. An *inorder* traversal is initiated in the tree and the domains falling in the selected interval alone are considered for matching. The inorder traversal of a binary search tree results in sorted list of nodes. The procedure can be summarized as follows:

1. Divide the image into non-overlapping range blocks.
2. Divide the image into overlapping domains blocks of larger size.
3. Find the fractal dimension of domain blocks and insert each domain to an AVL tree, the position being decided in the order of the fractal dimension.
4. Compute the fractal dimension of the current range block.
5. Fix an interval of FD, for deciding on the domain pool. E.g. it can be at 0.2 distant from range FD. Say, the interval [2.15, 2.55] for a range with FD 2.35.
6. Locate the node with the least FD value in the interval. A general lookup procedure for binary search tree can be used.
7. (a) Test for matching the range-domain pair.
   (b) Perform inorder traversal to get the next node in the domain tree to find the required domains.
   (c) Find the best match from the domains until the node with maximum allowed FD is read from the tree.
8. Repeat from step (4) until all ranges are processed.

3.2. Features

1. The domain pool is selected dynamically based on the fractal dimension of the range considered.
2. The method can be used with any technique to estimate fractal dimension.
3. The pool boundaries are decided dynamically according to the range FD.
4. The domain pool range can be extended if an optimum matching pair is not obtained. The traversal can first be extended to the upper extension. A re-traversal shall be initiated only if a matching pair is not obtained in the upper extension. Thus in rare cases, the searching can be extended to the entire pool, to obtain the best matching domain.

The insertion and search in an AVL tree can be performed in *O (log n)* time, where n is the number of nodes. The worst case time complexity is also *O (logn)*. There is no other search tree structure with better time complexity than AVL trees. Hence the time for finding the required domain is better than conventional methods.

4. Results

The method has been tested for different type of images. For fractal compression, the program has been based on the method proposed by Fisher [14], with minor code optimizations. Fractal dimension was estimated with the improvements suggested by [10] on differential box counting.

Error estimates like PSNR and RMS with the reconstructed images has been found out. Results for ten of different types for Fisher's method are listed in Table 1.

The images were tested for the proposed method. The results are shown in Table 2.

Table 1. Results of basic method

Image	PSNR	RMS
astronomic	27.6532	10.5633
cameraman	26.2497	12.3396
lena	28.3241	9.7791
landscape	23.2673	17.5024
medical	23.9976	16.0916
nayanar	27.0037	11.3841
flowervase	26.2321	12.4251
highcourt	29.6988	8.3676
road	27.0403	10.8268
sachu	30.9867	7.1978

Table 2. Results with proposed method

Image	PSNR	RMS
astronomic	28.0586	10.0825
cameraman	25.7425	13.1643
lena	26.7537	11.7166
landscape	24.161	15.7939
Medical	23.9431	16.1920
nayanar	25.7045	13.2225
flowervase	29.7481	8.3004
highcourt	30.8415	7.3198
road	28.1554	9.9701
sachu	28.8635	9.1911

It is seen that in general more than 70% of the range blocks succeeds to find a matching pair in the set interval itself. In other cases the search pool has been extended by moving back to the parent node of the traversed subtree and traversing to its second child. This increases the number of operations significantly, but

still keeps below that of the basic method. The programs were tested with a Core i5 processor with 2.53GHz clock speed, 3MB cache and 3GB RAM. A comparison of encoding time with A. Conci's method is given in Table 3.

Table 3. Comparison of encoding time

Image (256x256)	Time Average	
	Basic method	Proposed method
Astronomic	0.105466	0.076154
Cameraman	0.102452	0.073908
Lena	0.097601	0.080073
Landscape	0.098654	0.067821
Medical	0.110742	0.079087
Nayanar	0.093967	0.075502
Flowervase	0.096580	0.095205
Highcourt	0.156002	0.012254
Road	0.136588	0.102076
Sachu	0.090254	0.083664

Two images and the reconstructed ones are shown in the figure 3. The difference between the original image and the reconstructed one is also shown.

(i)a

(ii)a

(i)b

(ii)b

(i)c

(ii)c

Figure 3. Images of (i) Lena and (ii) Sachu: (a) original image, (b) decompressed (c) difference image

5. Conclusion

Achieving a significant improvement in the encoding time of fractal image compression will lead the technique to the front of competing compression methods. We have attempted to classify the domain blocks based on the theory that fractal dimension, if estimated properly, can be used as a measure of texture variations. The results show that significant increase in encoding time can be achieved without compromising the quality of decompressed image.

Techniques for estimation of fractal dimension are still being in its infancy. No method has proven to be stable for all type of signals. Classification of domain blocks with a better estimate of the similarity of color images, shall give better results with our method.

Because our method is proposed to use a better data structure to hold the domains prepared for processing, it can be implemented with other domain classification schemes also. The method has no role in improvising the compression ratio.

References

[1] Mandelbrot, B. (1982)*Fractal Geometry of Nature.* Freeman and Co.,SanFrancisco.

[2] Barnsley, M. (1993). *Fractals Everywhere*, 2nd edn. Academic Press, Cambridge.

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

[4] Welstead, S.T. (1999). *Fractal and Wavelet Image Compression Techniques.* SPIE Press, Washington.

[5] Doudal, S. *et.al.* (2011). A reduced domain pool based on DCT for a fast fractal image encoding. Electronic Letters on Computer Vision and Image Analysis 10(1):11-23.

[6] Kovacs, T. (2008): A fast classification based method for fractal encoding. Image and Vision Computing , **26,** pp.1129-1136

[7] Liu, B. ; Yan, Y. (2010): An improved Fractal Image Coding based on Quadtree. Proc. Of 3rd International Congress on Image and Signal Processing, CISP, pp.529-532.

[8] Conci, A. ; Aquino, F. R. (2005): Fractal coding based on image local fractal dimension. Computational and Applied Mathematics , **24**, pp. 83-98.

[9] Sarkar, N. ; Choudhuri, B.B. (1994): An efficient differential box counting approach to compute fractal dimension of image. IEEE transactions. On Syst. Man & Cybernetics, **24,** pp.115, 120.

[10] Conci, A.; Campos, C.F.J. (1996): An efficient Box-counting fractal dimension approach for experimental Image Variation Characterization. Proceedings of IWISP'96-3rd International Workshop in Signal and Image Processing, Elsevier Science, UK, pp. 665-668.

[11] Velskii, A.G; Landis, G.E. (1966): An algorithm for the organization of information. Proceedings of the USSR Academy of Sciences**,** **146**, pp. 263–266. **(**Russian**)** English translation by Myron J. Ricci in *Soviet Math. Doklady*, (1962), **3**, pp. 1259–1263.

[12] Tremblay, J; Sorenson, P.G. (1991). *An introduction to data structures with applications*, 2nd edn. Mcgraw-Hill Education, New Delhi.

[13] Knuth, D. E. (1997). *The Art of Computer Programming, 3rd edn.* Addison-Wesley, London.

[14] Fisher, Y. (1995). *Fractal Image Compression-Theory and Application.* Springer-Verlag, New York.