Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

# *Analyzing Quadtree Partitions with Different Thresholds on Various Type of Images in Fractal Compression*

[Jeet Kumar and Manish Kumar]

***Abstract*—The paper analyzes the effect of quadtree partition method with different thresholds on various images in order to compress an image with fractal approach. The paper also addresses the cases where lossless compression cannot be achieved. The traditional fractal image compression methods are based on two different partitions on the same image but this paper provides single quadtree partition method to achieve compression.**

***Keywords*—Quadtree, threshold, fractal and image compression.**

## I. Introduction

The quadtree partition employs the popular and simple image partitioning technique based on a recursive splitting of selected image quadrants (shown in Fig. 1), enabling the resulting partition to be represented by a tree structure in which each non-terminal node has four descendents. The usual top-down construction starts by selecting an initial level in the tree, corresponding to some maximum range block size, and recursively partitioning any block for which a match better than some preselected threshold is not found (or more efficiently, by deciding whether to split a block by examining the variance of its pixels. The alternative bottom-up construction begins with a uniform partition using the smallest block size, and then proceeds to merge those neighbouring blocks for which a more efficient representation is provided by the resulting larger block one level up the quadtree. Compact coding of partition details is possible by taking advantage of the tree structure of the partition [4].

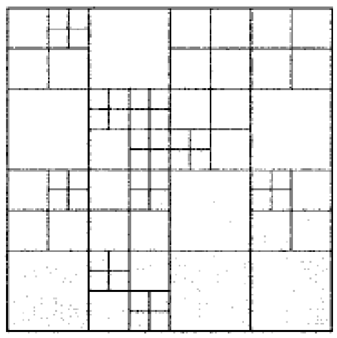
Fig. 1. Quadtree Partition

Jeet Kumar and Manish Kumar
*Shri Ramswaroop Memorial Group of Professional Colleges, Lucknow,* India

Jacquin's original scheme used a variant of the quadtree partition in which the block splitting was restricted to two levels. Instead of automatically discarding the larger block prior to splitting it into four subblocks if an error threshold was exceeded, it was retained if additional transforms on up to two subblocks were sufficient to reduce the error below the threshold [4].
In this paper the threshold is used as a number between 0 and 1. This threshold is multiplied by a number 255 because the values of the pixels in experimental images are in the range 0-255. The product thus obtained is the maximum permissible difference between the minimum and maximum values of pixel in a partition.

## II. Literature Review

Instead of storing an image bit by bit the idea to store the image in the form of contractive transformation was given by Michael Barnsley in 1988 [1]. Barnsley's graduate student Arnaud Jacquin implemented the first automatic algorithm in software in 1992 [2]. Since then the field of fractal image compression has evolved rapidly. Many ideas have been proposed till date towards the improvement of the image compression with fractal approach but still extensive computation requirement for compressing the image and closeness between domain and range blocks are the major issues. Traditionally an image is partitioned into non-overlapping range blocks and domain blocks (non-overlapping constraint is relaxed in domain blocks). Usually sizes of the domain blocks are larger than the range blocks to fulfil the contractive requirement. Some research work also advocated domain blocks of same size as that of range blocks to exploit self-similarity at same scale [3].
The size and shape of range blocks may vary to a great extent, but in many approaches square shaped range blocks are preferred. Although various mechanisms have been proposed for image partition. Some approaches like fixed size partition, quadtree partition, horizontal vertical partition and irregular partition fall under right angled partition category while other partition schemes like triangular and polygonal partition can be used [4].
Usually size of the domain blocks are larger than range blocks [5]. But most of the research is focussed on the fixed square shaped block of size B×B for range and 2B×2B for domain [6-8]. For each range block, i, every domain block is explored with all possible transformations. Therefore this approach needs complete domain pool searching and applying all the

transformations one by one which consumes much time. After this a best matched domain block is selected on the basis of minimum distance [9,10]. At last for each range block the location of the best matched domain block is stored along with the transformation applicable on the particular domain block. Therefore the image is stored as a list of domain block locations and corresponding transformations.

The traditional fractal image compression method described in the previous paragraph is lossy. In fact most of the fractal image compression methods are lossy. Only very few methods are lossless, for example the method given by Korakot Prachumrak et. al. that makes extensive usage of simultaneous equations is lossless [11]. A common characteristic of most images is that the neighbouring pixels are correlated and therefore contain redundant information [12].

## III. Proposed Work

Different applications need different type of compression. Some application can tolerate loss up to a great extent but other cannot tolerate the loss of even a single bit. These different needs can be handled by setting up the threshold value.
Traditionally the fractal image compression was done by dividing a given image into domain blocks and range blocks, i.e., two partitions are required on the same image. The approach given here do not require two different partitions on the same image, rather a single partition will serve the purpose.
The output of quadtree decomposition can be explored for self-similar subblocks. If a subblock with certain pixel values found elsewhere in the image with same size or different size, that block need not to be stored twice. The pixel value of the block can be stored in a sorted array along with the information of its replication, i.e., replication information can be in the form of a triplet (row index for upper left corner, column index for upper left corner, block size).
The work has been conducted on four types of images as shown below.

### A. *Type 1 image*

The image shown in figure 1 has only two gray levels. Moreover these two gray levels occur in the regular or defined regions therefore simplify the compression task.

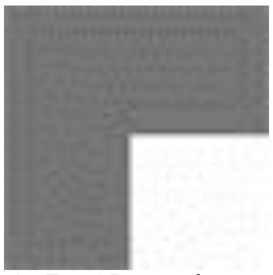
Fig. 2. Type 1 Image with two areas

### B. *Type 2 image*

The image shown in figure 2 has many gray levels. Moreover these gray levels do not occur in the regular or defined regions. Since the image is coarse grained, the compression task is not very much complex.

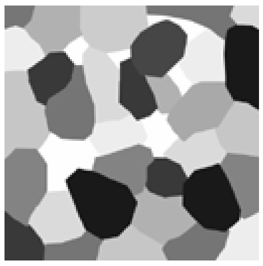
Fig. 3. Type 2, Coarse Grained Image

### C. *Type 3 image*

The image shown in figure 3 has many gray levels. But these gray levels occur in somewhat defined regions. These images are also known as gradient images. Since the gradient follows a pattern, the compression task is not very much complex in this case also.

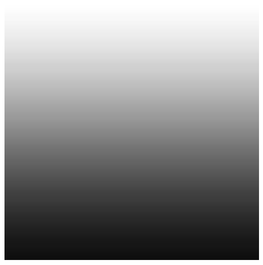
Fig. 4. Type 3, Gradient Image

### D. *Type 4 image*

These types of images are very complex. The image shown in figure 4 has many gray levels. But these gray levels are so irregularly placed that the search of local similarity is very hard. These images are also known as fine grained images.

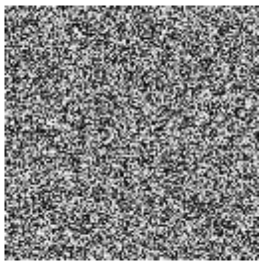
Fig. 5. Type 4, Fine Grained Image

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

## IV. Experimental Results

Experiments have been done on these four types of images as given below.

### A. *Working with type 1 image*

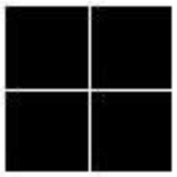
Fig. 6(a). Quadtree partition of type 1 image with 0 threshold

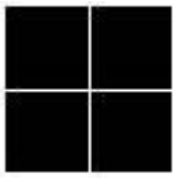
Fig. 6(b). Quadtree partition of type 1 image with 0.1 threshold

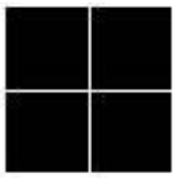
Fig. 6(c). Quadtree partition of type 1 image with 0.2 threshold

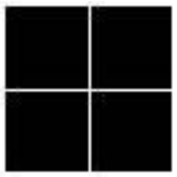
Fig. 6(d). Quadtree partition of type 1 image with 0.3 threshold

TABLE I
NUMBER OF BLOCKS OF VARIOUS SIZES IN TYPE 1 IMAGE

| Image Type | Threshold | Block Size | No of Blocks | Total |
|---|---|---|---|---|
| Type1 | 0.0 | 2×2 | 0 | 4 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 4 | |
| | | 64×64 | 0 | |
| | 0.1 | 2×2 | 0 | 4 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 4 | |
| | | 64×64 | 0 | |
| | 0.2 | 2×2 | 0 | 4 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 4 | |
| | | 64×64 | 0 | |
| | 0.3 | 2×2 | 0 | 4 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 4 | |
| | | 64×64 | 0 | |

### B. *Working with type 2 image*

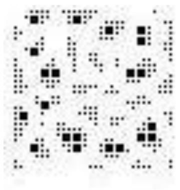
Fig. 7(a). Quadtree partition of type 2 image with 0 threshold

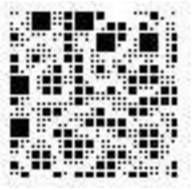
Fig. 7(b). Quadtree partition of type 2 image with 0.1 threshold

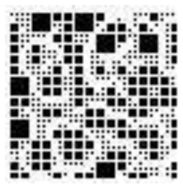
Fig. 7(c). Quadtree partition of type 2 image with 0.2 threshold

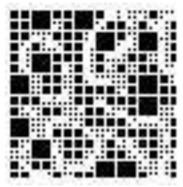
Fig. 7(d). Quadtree partition of type 2 image with 0.3 threshold

TABLE 2
Number of Blocks of Various sizes in Type 2 Image

| Image Type | Threshold | Block Size | No of Blocks | Total |
|---|---|---|---|---|
| Type2 | 0.0 | 2×2 | 231 | 252 |
| | | 4×4 | 21 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.1 | 2×2 | 310 | 405 |

| | | | | |
|---|---|---|---|---|
| | | 4×4 | 89 | |
| | | 8×8 | 6 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.2 | 2×2 | 295 | 408 |
| | | 4×4 | 106 | |
| | | 8×8 | 7 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.3 | 2×2 | 271 | 390 |
| | | 4×4 | 107 | |
| | | 8×8 | 12 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |

## C. *Working with type 3 image*

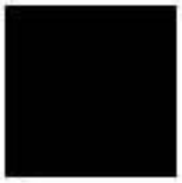
Fig. 8(a). Quadtree partition of type 3 image with 0 threshold

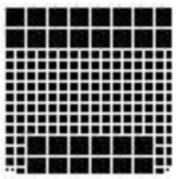
Fig. 8(b). Quadtree partition of type 3 image with 0.1 threshold

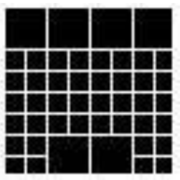
Fig. 8(c). Quadtree partition of type 3 image with 0.2 threshold

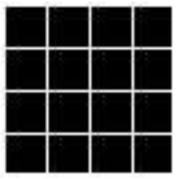
Fig. 8(d). Quadtree partition of type 3 image with 0.3 threshold

TABLE 3
Number of Blocks of Various sizes in Type 3 Image

| Image Type | Threshold | Block Size | No of Blocks | Total |
|---|---|---|---|---|
| Type3 | 0.0 | 2×2 | 0 | 0 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.1 | 2×2 | 6 | 176 |
| | | 4×4 | 142 | |
| | | 8×8 | 28 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.2 | 2×2 | 0 | 46 |
| | | 4×4 | 0 | |
| | | 8×8 | 40 | |
| | | 16×16 | 6 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.3 | 2×2 | 0 | 16 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 16 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |

## D. *Working with type 4 image*

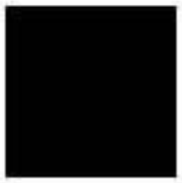
Fig. 9(a). Quadtree partition of type 4 image with 0 threshold

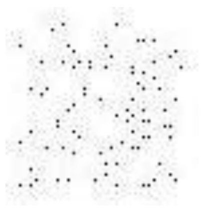
Fig. 9(b). Quadtree partition of type 4 image with 0.1 threshold

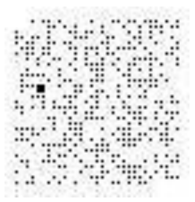
Fig. 9(c). Quadtree partition of type 4 image with 0.2 threshold

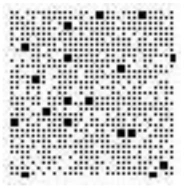

Fig. 9(d). Quadtree partition of type 4 image with 0.3 threshold

TABLE 4
Number of Blocks of Various sizes in Type 4 Image

| Image Type | Threshold | Block Size | No of Blocks | Total |
|---|---|---|---|---|
| Type4 | 0.0 | 2×2 | 0 | 0 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.1 | 2×2 | 91 | 91 |
| | | 4×4 | 0 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.2 | 2×2 | 421 | 422 |
| | | 4×4 | 1 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |
| | 0.3 | 2×2 | 742 | 760 |
| | | 4×4 | 18 | |
| | | 8×8 | 0 | |
| | | 16×16 | 0 | |
| | | 32×32 | 0 | |
| | | 64×64 | 0 | |

## V. Conclusion

Images of type 1 are very easy to compress and provides a high compression ratio. We have lesser number of quadtree partitions therefore need lesser space, moreover change in the threshold do not leave much effect.

Maximum blocks in the images of type 2 are small, rarely we have larger blocks in this case. Self-similar blocks can be obtained at smaller scale. Compression outcomes are moderate in this case. Relaxation in tolerance does not leave much effect in this case also.

In type 3 images a gradual mandatory change is observed in pixel values, therefore zero threshold does not provide any lossless fractal compression on square blocks. If we relax the tolerance a little bit (threshold=0.1) we have a huge number of smaller blocks. Lossless fractal compression may be obtained if we work on rectangular partitions.

A lossless fractal compression of type 4 images is very complex. We do not have much to do until the tolerance criterion is relaxed.

## References

[1] Michael Barnsley, "Fractals Everywhere", Academic Press, Inc., 1988.
[2] Arnaud E. Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations", IEEE Transactions on Image Processing, Vol.1, No.1, January 1992.
[3] Yao Zhao and Baozong Yuan," A Novel Scheme for Fractal Image Coding", Institute of Information Science Northern Jiaotong University, Beijing 100044, P.R.China, May 2001.
[4] Brendt Wohlberg and Gerhard de Jager, "A Review of the Fractal Image Coding Literature", Member, IEEE, December 1999.
[5] Gaoping Li, "Fast Fractal Image Encoding Based on the Extreme Difference Feature of Normalized Block", College of Computer Science & Technology, Southwest University for Nationalities, Chengdu, China, 2009.
[6] Jinshu Han, "Fast Fractal Image Encoding Based on Local Variances and Genetic Algorithm", Dezhou University, China, 2009.
[7] Ying Zhao, Jing Hu, Dongxiang Chi and Ming Li, "A Novel Fractal Image Coding based on Basis Block Dictionary", School of Electronics and Information, Shanghai dian ji University, Shanghai, China, 2009.
[8] Yang Liu and Jin-guang Sun, "Face Recognition Method Based on FLPP", Liaoning Techinical University, Huludao Liaoning, China, 2010.
[9] D. Loganathanff, J. Amudha and K.M. Mehata, "Classification and Feature Vector Techniques to Improve Fractal Image Coding", Electrical and Electronics Engineering, Amrita Institute of Technology and Science, Coimbatore, INDIA, 2003.
[10] Shen Furao and Osamv Hasegawa,"An Effective Fractal Image Coding Method Without Search", Japan, 2004.
[11] K. Prachumrak, A. Hiramatsu, T. Fuchida and H. Nakamura, "Lossless Fractal Image Coding ", Croatia, 2003.
[12] Sachin Dhawan, "A Review of Image Compression and Comparison of its Algorithms", India, 2011.

About Author (s):

The first author of the paper is working as Assistant Professor in the department of Computer Applications at Shri Ramswaroop Memorial Group of Professional Colleges, Lucknow for more than 10 years. The author has achieved MCA, M.Phil. & M. Tech. degrees and pursuing Ph. D. (CS). The author has also qualified UGC-NET with JRF in Computer Science and Applications and got published more than 10 papers in the reputed forums like Springer etc.