

Reduction of Domain Blocks Based on a Clustering Method for Fractal Compression

[†]Hsiao-Wen Tin, [†]Shao-Wei Leu^{*}, [‡]Shun-Hsyung Chang

[†] Dept. of Electrical Engineering, National Taiwan Ocean University, Keelung City, Taiwan

Email: {D94530006, B0119^{*}}@mail.ntou.edu.tw

[‡] Dept. of Microelectronic Engineering, National Kaohsiung Marine University, Kaohsiung City, Taiwan

Email: shchang@mail.nkmu.edu.tw

Abstract- *The process of fractal compression generates a number of intermediate artifacts during the encoding phase. These artifacts, known as range blocks and domain blocks, largely determine the coding efficiency. The coding efficiency can be characterized by the processing time required and the quality of the reconstructed image. A reduction in the number of domain blocks is very beneficial to the coding efficiency. The number of domain blocks is a crucial factor when reconstructing an image of high quality. However, deletion of too many domain blocks or simply some blocks with important characteristics can have very negative effect on the quality of reconstructed images. To obtain a good reconstructed image, it must be ensured that the deleted domain block do not carry any important characteristics and should be identified as insignificant before deletion. We apply the clustering method here to group related blocks together to ensure that the deleted blocks are redundant. That is, the retained blocks are significantly different to each other. Thus, the range blocks are compared only to significant domain blocks and not to redundant ones. The reduction of the intermediate data greatly benefits coding efficiency as well. The experimental results show that the peak signal-to-noise ratio (PSNR) of the reconstructed image is close to or around 30dB, that is, the quality of the reconstructed image is visually as good as the original.*

Keywords: Clustering method, fractal compression, domain pool reduction.

1. Introduction

The process of fractal compression has always been computationally time-consuming as well as

resource-demanding. The encoding process of fractal compression first compares each range block with all of the blocks in the domain pool and then determines the best match. In the worst case, it requires $n \times m$ comparisons of range blocks to domain blocks, where n and m are the amounts of range blocks and domain blocks, respectively. The more pieces the image is divided into, meaning larger n and m , the better the quality of the reconstructed image, however, at the expense of longer processing time for encoding. Therefore, a more efficient encoding method is highly desirable in the pursuit of high quality image reconstruction with limited processing time and resource budgets. During the encoding phase of fractal compression, a number of methods to reduce complexity of the encoding process have been proposed, such as *nearest neighbor search* [2]-[4], *clustering methods* [5], [6], and *domain pool reduction* [7].

As we pointed out earlier that, in the encoding phase, a significant amount of processing time is devoted to matching the range blocks with the domain blocks. Therefore, reducing the amount of domain blocks will no doubt cut the processing time and resource requirement significantly. However, if too many domain blocks are deleted, quality of the reconstructed image will suffer. Therefore, how to delete as many domain blocks as possible without hurting the quality of reconstructed images is a critical issue which greatly concerns the overall efficiency of fractal encoding. For example, the original work done by Jacquin, splitting an image into a shade block and a non-shade block, yielded only 11% reduction rate[8], which apparently does not translate into a significant contribution to the saving of processing time and resources. To develop a more efficient and more balanced approach to solving this problem, we have adopted a clustering method to group together those related domain blocks and

devised a rule to effectively eliminate a large amount of unneeded domain blocks. Our approach can delete up to 97% of domain blocks while retaining the peak signal-to-noise ratio (PSNR) of the reconstructed image at more than 30 dB.

The rest of this article is organized as follows. Sec. 2 briefly reviews the basics of fractal coding and a partition clustering method; Sec. 3 describes the proposed method in detail; Sec. 4 presents the experimental results; Sec. 5 gives the conclusion.

2. Previous Work

2.1. Conventional Fractal Coding Scheme

Fractal compression reconstructs the image according to its characteristics: *self-similarity* and *self-affinity* [8]-[10]. The major idea is to represent a compressed image with the transform parameters.

Most of the self-similarities in an image are localized. With fractal compression, the image is split into domain blocks and range blocks. The collection of domain blocks is called *the domain pool*. Each range block may find its associated domain block from the domain pool by using the transform function. Fractal encoding records the corresponding transform function and the position of the associated domain block. Fractal decoding reconstructs the image iteratively with the corresponding transform function and the position of the associated domain block.

With fractal encoding, the image is divided into range blocks R and domain blocks D , where $|R| < |D|$. The number of domain blocks is typically 4 times as many as the number of range blocks. The size of the domain pool will impact the matching efficiency of each range block to its domain block and the quality of the reconstructed image. Range block R_i matches the associated domain block D_i within domain pool D . There exists a corresponding transform function T_i which minimizes $dtr(T_i(D_i), R_i)$, where dtr is the metric distortion measure function and D_i is the sub-sampling of D . A common sub-sampling method is illustrated in formula (1), where D_i is the average of four adjacent points. This formula is sufficient when the domain block is exactly 4 times the size of the range block. In other cases, the domain block must be scaled with equation (2), where S_D is the size of the domain block and S_R is the size of the range block.

$$D_i(x, y) = (D_i(x, y) + D_i(x+1, y) + D_i(x, y+1) + D_i(x+1, y+1)) / 4 \quad (1)$$

$$S = \frac{S_D}{S_R} \quad (2)$$

Using an affine transform formula, we can transform D to a similar range block as shown in formula (3). This is the basic idea of self-similarity searching.

$$\hat{R} = I\{\alpha \cdot (S \circ D) + O\} \quad (3)$$

$$MSE = \frac{1}{m \times n} \sum \sum (R_{i,j} - \hat{R}_{i,j})^2 \quad (4)$$

As can be seen in formula (3), the domain block D is scaled by S , contracted by α , and given a luminance transform O . After the final isometric process I , the affine range block \hat{R} is obtained. Formula (4) is used to calculate the mean square error (MSE) of \hat{R} and R to determine the best matches. A fractal code includes the transform function coefficient and the description of the transform of the domain block to the best matching range block. Fractal decoding reconstructs the image by transforming T_i and the position of D_i with an iterated contractive transformation.

2.2. K-means

K-means is a partition clustering method proposed by MacQueen [11]. The main idea is to cluster all the data L into K partitions, where $K < L$. Using the Euclidean distance, the partitioning starts by finding the minimum summed distances which are from every data point to the center of that partition. The K-means formula is shown in formula (5).

$$E = \sum_{i=1}^K \sum_{j=1}^L |C_j - X_i|^2 \quad (5)$$

where E is the Euclidean distance, k is the number of partition centers, and L is the number of data.

The objective of the K-means' function is to target the center of the partition. Each iteration minimizes the total intra-cluster variance or the squared error function until the best objective function is obtained. The K-means algorithm is as follows:

- Step 1: Calculate initial partitions and the center of each partition.
- Step 2: Cluster data close to the center as one partition.
- Step 3: Recalculate the center and, if new center is better, replace the old one.
- Step 4: Repeat step 2 and 3 until the partitioning is stable or does not change.

3. Domain Pool Reduction

3.1. The algorithm

In fractal encoding, the range blocks are compared to domain blocks in the domain pool for matching information. A large domain pool means more processing and longer processing time are needed. Clustering of domain blocks can help avoid deletion of domain blocks with important characteristics, hence ensure good quality of reconstructed images while keeping the amount of encoded data at low level.

The clustering method splits data into reasonable partitions which have the following two characteristics: (a) the data in the same partition have significant correlation, and (b) the data in different partitions have significant differences. With these characteristics, this paper will apply the clustering method to partition the domain blocks. The objective of clustering is to avoid mis-deletion. Thus, the deletion applies only to similar domain blocks. At the same time, it eliminates comparisons of range blocks with duplicate domain blocks. The number of domain block partitions would affect the coding efficiency and the quality of the reconstructed image. The more partitions there are, the better the image quality. However, fewer partitions result in faster encoding.

This paper applies the K-means method to partition the domain blocks. The cluster coefficients determine the partitions of domain blocks by their correlation. The partitions replace the domain blocks after clustering which decreases the number of domain blocks. The idea of clustering is shown in Figure 1. Assume there are 8 domain blocks and 2 range blocks. The domain blocks are clustered into 3 partitions. The blocks are then replaced by partitions, that is, the domain blocks are reduced to 3 blocks. This reduces the comparison of range blocks with domain blocks from 2-to-8 to 2-to-3.

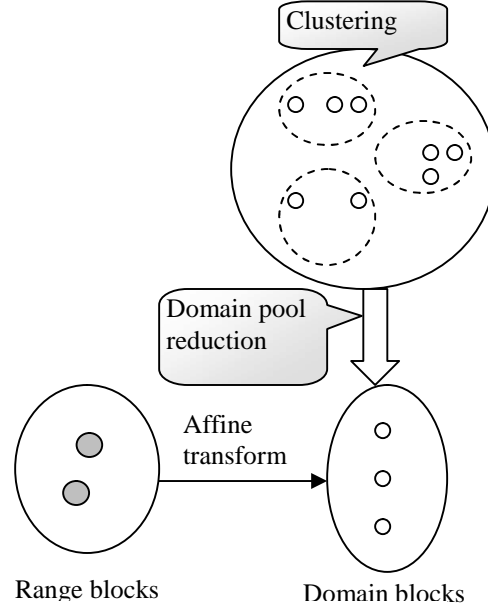


Figure 1. Reduction of domain blocks

The domain blocks are generated sequentially along with the pixels in the image. The clustering coefficients are based on the next 2 characteristics: (a) the distance from each domain block to the first block, and (b) the gray level. The formula to calculate the distance is shown in formula (6).

$$dist((A_x, A_y), (B_x, B_y)) = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2} \quad (6)$$

where B is the domain block to be processed, (B_x, B_y) is its location, A is the first domain block, and (A_x, A_y) is its location.

Improper reduction of domain blocks decreases the quality of the reconstructed image. Both the clustering coefficients and deletion guidelines are very important. The parameters of our algorithm are described as follows:

- Let $\{d_1, d_2, \dots, d_n\}$ be the distance set of n domain blocks, where $d_i = dist(D_1, D_i), i \geq 1$. d_1 is the distance between the first domain block to itself and d_2 is the distance between the second domain block to the first one.
- Let $\{g_1, g_2, \dots, g_n\}$ be the gray level set of n domain blocks and $g_i \in \{0, 1, 2, \dots, 255\}$. g_1 is the gray value of the first block and g_2 is the gray value of the second block.
- Using formula (5), we cluster n domain blocks into D partitions, where

$$1 < |D'| < n.$$

- Let c_1 and c_2 be the centers of two respective partitions.
- Let $\alpha = \{h_1, h_2, \dots, h_n\}$ be the correlation set of n domain blocks, where h_i is the correlation to the coefficients which are d_i and g_i of the i^{th} domain block.

The algorithm for our reduction is described below:

- Step 1: Let P be the original image, D the domain block, and R the range block. We get the sub-sample of the domain block with formula (1).
- Step 2: Cluster the domain blocks into 2 partitions. Randomly pick 2 domain blocks as the centers, c_1 and c_2 .
- Step 3: Calculate the distance d_i from domain block D_i to the first domain block D_0 using formula (6), and record the corresponding gray value g_i . Take d_i and g_i as the clustering coefficients with formula (5). If α meets the reasonable requisition, it replaces c_1 and c_2 and gives the partition D' .
- Step 4: Domain blocks in the same partition are treated as one domain block. Replace the domain block D with the D' in the partitions.
- Step 5: Process the fractal coding with D' and R . After decoding, the algorithm comes out with the reconstructed image P' .
- Step 6: In the worst coding situation, the comparison of the range block with domain blocks will be $R \times D'$, and $|D'| < |D|$.
- Step 7: Evaluate the PSNR of P' using $PSNR(P, P')$.
- Step 8: Repeat Step 3 to 7 until $|(R \times D)| < |(R \times D')|$ and $PSNR(P, P') \geq 30$ dB.

3.2. System Procedures

The system first receives the original image as an input and with fractal compression, splits the image into a range block R and a domain block D . With the clustering coefficients and the K-means method, the system clusters the domain blocks D into partitions. Blocks in the same

partition are treated as one of the same. Thus, the system picks one out of the partition to represent all of the domain blocks and deletes the rest. The number of domain blocks thus reduces to the number of partitions. During encoding, the system compares the range blocks R with the post-reduction domain blocks D' . Using formula (3) and (4), we calculate MSE. If the MSE value is less than the pre-defined threshold V , the range block R matches its domain block D' . Then, the system records the transform coefficients and completes the encoding. While decoding at the receiver end, it reconstructs the image by iterating through the transform function. The flow chart is illustrated in Figure 2.

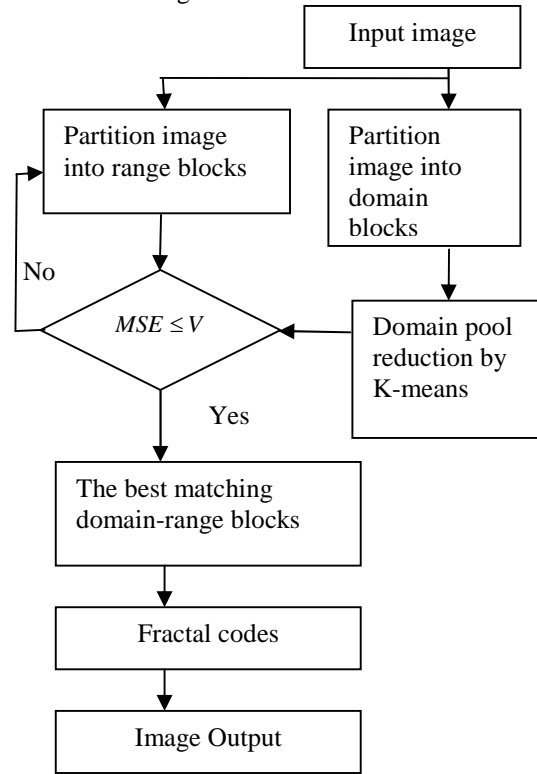


Figure 2. System Procedures

4. Experiments and Results

The sample gray-level medical image 99110 was obtained from the Centers for Disease Control and Prevention. The original file format is tiff. We changed it to a 150 x 148-pixel bitmap file as shown in Figure 3.

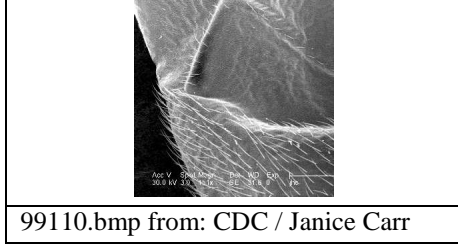


Figure 3. A medical image sample

With the quadtree method proposed by Fisher [12], image 99110 splits into non-duplicate range blocks of size 8×8 . The total number of range blocks is $18 \times 18 = 324$. The total number of duplicable domain blocks of size 16×16 is $135 \times 133 = 17955$.

Our program is based on the theory described in Lu [13]. As the domain blocks are non-duplicate, image 99110 can be split into 5,550 domain blocks. After our proposed cluster reduction method, the amount of domain blocks shrinks down to 128 blocks. The PSNR of the reconstructed image is 31.2017 dB. The number of domain blocks is less than that of Lu's method. The reduction rate is as high as 97%. A comparison of the different methods for image 99110 is shown in Table 1.

Table 1. Number of domain blocks compared

Method	Fisher	Lu	Proposed Method
# blocks	17955	5550	128

Our experiment calculates the number of comparisons between range blocks and domain blocks with the clustering reduction method. Our objective is to reduce the coding data. However, any data reduction would influence the quality of the reconstructed image. We will use the quality of the reconstructed image in our evaluation.

Although the reduction in comparisons saves encoding time, the clustering increases processing time. Our system counts the actual comparisons between range blocks and domain blocks. The count is also one of the valuation objectives.

The quality of reconstructed images may be evaluated by using either subjective or objective fidelity criteria. Naturally, the human eyes stand out as the most convenient tool for evaluation if subjective fidelity criterion is chosen. Meanwhile,

the PSNR is a good choice for evaluation based on objective fidelity. A higher PSNR value means the reconstructed image closely matches the original one. In general, if the PSNR is more than 30 dB, the difference between the original and the reconstructed images is too little to tell by the human eyes. The PSNR function is derived in formula (7).

$$MSE = \frac{1}{N \times M} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} (B(x, y) - A(x, y))^2$$

$$PSNR = 10 \log_{10} \frac{I_{\max}^2}{MSE} \quad (7)$$

where $A(x, y)$ is the gray function of the original image, $B(x, y)$ is the function of the reconstructed image, N and M are the length and the width, and I_{\max} is the maximum gray value of the pixel, preset at 255.

In our experiment, the domain blocks are clustered into 2, 16, 32, 64 and 128 partitions. As an example, a classification diagram is shown in Figure 4. The results are listed in Table 2.

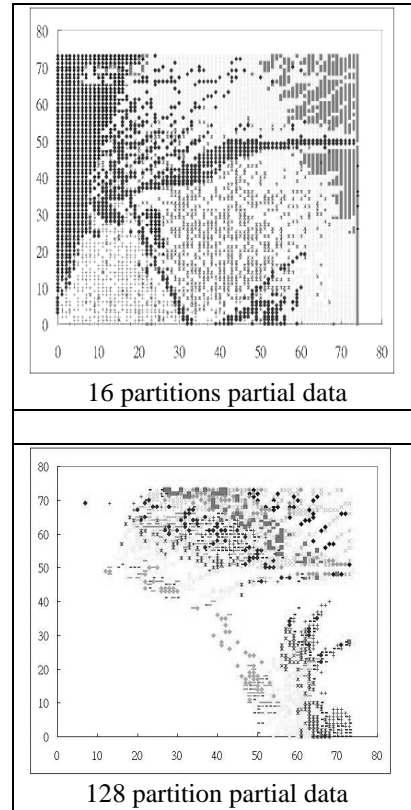
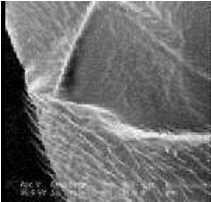
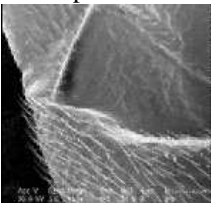
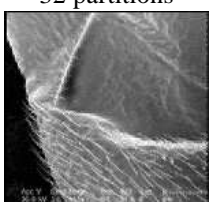
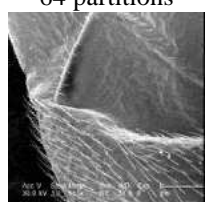
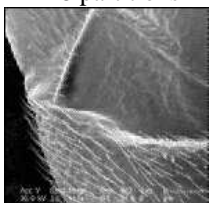


Figure 4. Classification diagram

Table 2. Experimental results

No. of partitions/ reconstructed image	No. of Comparisons	PSNR
2 partitions 	16650	25.8856
16 partitions 	138737	27.6000
32 partitions 	593734	27.9704
64 partitions 	1186949	28.0127
128 partitions 	2340612	31.2017

More clustered partitions means more domain blocks and thus more coding data. From the results in Table 2, the number of domain blocks is the major factor in influencing the PSNR. When the number of partitions increases, the number of gray levels in domain blocks increases. Additionally, the comparison of range blocks and domain blocks increases as well. This results in a larger PSNR after reconstruction. Take image 99110 for example, the PSNR reaches 31.2017dB while the number of partitions is 128. The reconstructed image is visually identical to the original one.

The partitions of domain blocks are a direct ratio to the comparisons. When the partitions

increase, the number of comparisons goes up and the processing time takes longer. Figure 5. Number of partitioned domain blocks vs. number of comparisons Figure 6 shows the number of partitioned domain blocks and the PSNR of the reconstructed image. A direct ratio is observed, that is, the increase of partitions increases the PSNR. This means that the more clustered partitions there are, the better the quality of the reconstructed image.

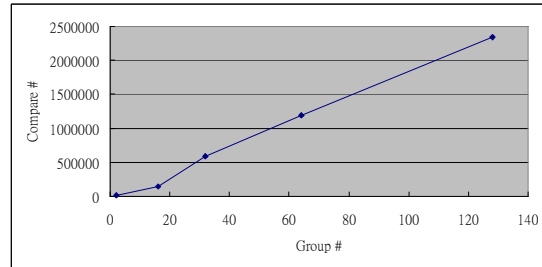


Figure 5. Number of partitioned domain blocks vs. number of comparisons

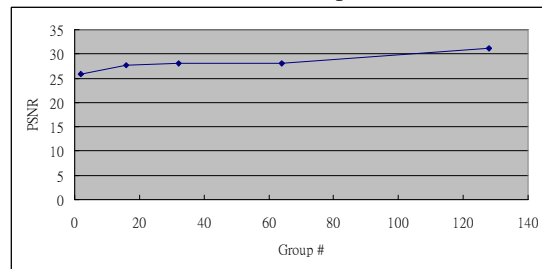


Figure 6. Number of partitioned domain blocks vs. PSNR

5. Conclusion

Although the reduction of domain blocks may decrease the data processing time during coding, mis-deletion may result in a loss of quality in the reconstructed image. To avoid mis-deleting domain blocks, we applied the clustering method to delete duplicate or similar domain blocks. After deletion, the remaining blocks are significantly different to each other. This also reduces the comparison of range blocks and domain blocks during encoding.

While encoding, we assume that there are n range blocks, and m domain blocks. In the worst case scenario, the number of comparisons between range blocks and domain blocks are $n \times m$. This paper improves the processing of this coding data. It clusters the domain blocks into j partitions, where $j < m$. The domain blocks are then reduced down from m to j . This time, in the worst

case scenario, the number of comparisons is $n \times j$, that is, a saving of $n \times (m - j)$ comparisons.

The clustering method partitions a great amount of data according to existing correlations. Data in the same partition may be treated identically. Representing data by partitioning simplifies the complexity of the original data. We applied the K-means method to cluster domain blocks, delete blocks and simplify the domain pool. The distance and gray level are our coefficients. The end result is very acceptable. In the future, we may apply other clustering methods, change the coefficients, or even apply the weighted clustering method to achieve the reduction of domain blocks to achieve a high quality reconstructed image.

References

- [1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering," *ACM Computing Surveys*, vol. 33, no. 3, pp. 264-323, 1999.
- [2] D. Saupe, "Accelerating fractal image compression by multi-dimensional nearest neighbor search," in *Proc. IEEE Data Compression Conference (DCC 95)*, pp. 222-231, 1999.
- [3] D. Saupe, "Fractal image compression via nearest neighbour search," in *Proc. NATO ASI Conf. on Fractal Image Encoding and Analysis*, July 1995.
- [4] C. A. Lang and A. K. Singh, "Accelerating high-dimensional nearest neighbor queries," in *Proc. IEEE International Conference on Scientific and Statistical Database Management (SSDBM'02)*, pp. 109-118, July 2002.
- [5] R. Hamzaoui, "Codebook clustering by self-organizing maps for fractal image compression," in *Proc. NATO ASI Conf. on Fractal Image Encoding and Analysis*, July 1995.
- [6] R. Hamzaoui and D. Saupe, "Combining fractal image compression and vector quantization," *IEEE Trans. Image Processing*, vol. 9, pp. 197-208, 2000.
- [7] D. Saupe, "Lena domain pools for fractal image compression," in *Proc. SPIE Electronic Imaging and Still Image Compression II*, vol. 2669, 1996.
- [8] A. E. Jacquin, *Image coding based on a fractal theory of iterated contractive Markov operators, Part II: Construction of fractal codes for digital images*, Technical Report Math. 91389-17, Georgia Institute of Technology, 1989.
- [9] A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Processing*, vol. 1, pp. 18-30, 1992.
- [10] M. F. Barnsley, "Fractal Image Compression," *Notices of the AMS*, vol. 43, pp. 657-662, 1996.
- [11] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281-297, 1967.
- [12] Yuval Fisher, *Fractal image compression: theory*

and application. Springer-Verlag, New York, 1994
[13] Ning Lu, *Fractal Image*. Academic Press, San Diego, 1997