Proc Natl. Sci Counc. ROC(A)
Vol 20, No 2, 1996 pp 185-193

(Scientific Note)

A Study of Fractals on Image Processing

HSIN-TONG KAO AND LING-LING WANG[†]

Institute of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R O C

(Received September 26, 1994, Accepted October 24, 1995)

ABSTRACT

The fractal image compression scheme is shown to be powerful in image compression In this method, an image is not represented in an array of pixels; instead, a relatively compact set of numbers, called iterated function system (IFS) codes, is used to code the segments that compose the image In this paper, a speed improvement in the fractal compression technique by using the N-step matching algorithm is proposed, and the feasibility of image processing based on IFS codes is presented. This scheme includes noise reduction, edge detection, and edge magnification or demagnification, which are jointly implemented with the decompression procedure instead of being treated as separate operations. In addition, texture segmentation can be achieved by using the IFS codes as features. Experimental results are presented to show the feasibility of the proposed approach.

Key Words: fractal image compression, iterated function system, edge detection, texture segmentation

I. Introduction

Since the pioneering work of Mandelbrot (1982), there has been considerable interest in the use of fractals to model natural phenomena, such as clouds, plants, landscapes, coastlines, and even whole planets. The great advantage of such a method is that a small set of parameters, called iterated function system (IFS) codes, can be used to specify an apparently complex image. This led Barnsley to use fractals to compress images (Barnsley and Hurd, 1993). This new technique is called fractal image compression (Barnsley and Alan, 1988; Barnsley and Hurd, 1993; Fisher, 1992; Louisa, 1993). It has attracted the attention of researchers in the area of image compression because of its high compression ratio, which is up to 10,000: 1 for special fractal images.

For fractal images, such as the Cantor set or Snowflake, several approaches have been proposed to compute the corresponding IFS codes. Kawamata *et al.* (1992) regarded IFSs as time-variant state-space digital filters and applied digital signal processing techniques to systematically determine IFS codes. Pei *et al.* (1993) considered the wavelet transform and scale space filtering as special cases of general scale shift mapping (SSM). The SSM was used as a tool to characterize the geometrical complexity of fractals. The results obtained from scale shift mapping were then used to estimate the IFS codes of a class of fractal images. To reduce the computational cost, Pei *et al.* (1992) provided a novel algorithm to decode a fractal image from IFS codes. It is suitable for parallel implementation and has no transient behavior in the decoding of IFS codes.

To compress a natural image, we first partition the image into blocks, and the affine transformations, explained in a later section, are then used to express the relations between blocks of the image. The parameters of these transformations are called local IFS codes, which approximate the natural image and are used for image decompression or reconstruction. The amount of storage for these codes is much smaller than that of the original image. The larger the partitioned blocks are, the fewer IFS codes there are, and the worse is the reconstructed image.

Encoding of natural images by fractals is computationally intensive. In this paper, speed improvement in the fractal image compression technique by using the *N*-step matching algorithm is proposed. Additionally, characteristics of the local IFS codes that

---

[†]To whom all correspondence should be addressed.

are meaningful and useful for some image processing tasks are described. These tasks include edge detection, edge magnification or demagnification, and texture segmentation. The edges of an image can be extracted by slightly modifying the IFS codes in the decompression process. Magnified or demagnified non-jagged edges can be easily obtained in decompression, and texture segmentation can be achieved through classification of the parameters of local IFS codes. The experimental results indicate that IFS codes are promising features for image processing.

In this paper, detailed descriptions of local IFS codes and fractal image compression techniques are given in Section II. In Section III, applications of fractals or local IFS codes on image processing are proposed, and experimental results are presented. Conclusions and suggestions for further researches appear in the last section.

II. Fractal Image Compression

This section introduces some basic concepts in fractal geometry and in the implementation of fractal image compression.

1. Fractal Geometry

Affine transformations and IFSs are central to fractal image compression and are briefly discussed in this subsection.

A transformation is a function which takes points from one space to another. In fractal geometry, transformations operate on a metric space (X, δ), where X is a space, and δ is a metric defined on X. Let $(\mathcal{R}^2, \delta)$ be a metric space, where $\mathcal{R}$ is the set of real numbers, and δ is a metric on $\mathcal{R}^2$. An affine transformation defined on $(\mathcal{R}^2, \delta)$ is in the form of

$$w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix},$$

where a, b, c, d, e, and f are real numbers, and $(x, y) \in \mathcal{R}^2$. If an affine transformation w has the property

$$\delta(w(x), w(y)) \le s \bullet \delta(x, y), \text{ for any } x, y \text{ in } \mathcal{R}^2,$$

where s is a constant, and δ is a metric to measure the distance between pairs of points in $\mathcal{R}^2$, then the transformation w is said to be contractive, and s is called the contractivity factor because points move closer together after the transformation. When an affine transformation is performed on a metric space, there may exist points which remain unchanged after the transformation. These points are called fixed points. If an affine transformation is contractive, it can be proved that this transformation has fixed points (Mandelbrot, 1982).

An IFS consists of a metric space together with a finite set of contractive affine transformations. The set of fixed points of these contractive affine transformations is called the attractor of the IFS. The attractor constitutes a binary fractal image, and the parameters of the contractive affine transformations in the IFS are called IFS codes. The attractor of an IFS can be obtained with the aid of the deterministic algorithm or the random iteration algorithm (Barnsley and Hurd, 1993).

The affine transformations of an IFS on a metric space (X, δ) transform points in the space X to X. A fractal image is formed by affine transformations of its whole self. However, a natural image does not appear to contain affine transformations of its whole self; it appears to consist of copies of properly transformed "parts" of itself. So local transformation is defined. A local transformation on a space X is one whose domain is a subset of the space X, instead of all the points of the space; a global one is defined on all the points of the space. By extending the IFS concept from "global" to "local" and restricting the rotation transformations to eight affine symmetry rotations (Barnsley and Hurd, 1993), as shown in Fig. 1, it is not difficult to develop schemes for natural image compression.

2. Compression and Decompression

This section deals with compression and decompression of a natural gray-level image by using the fractal scheme (Jacquin, 1992; Monro, 1993; Monro and Dudbridge, 1992). An image is coded into local IFS codes in the compression process. In the decompression process, these codes are used to reconstruct the original image. The detailed procedures are discussed in the following.

A. Encoding

To compress a natural gray-level image, we first define a space on which the affine transformations operate. The space is $X=(I^2, f)$, where I^2 is the spatial domain of an image, and f is the intensity function of the image. We define a metric δ as follows:

$$\delta(I_1, I_2) = \sum_{(x,y)\in I_1, I_2} |f_1(x,y) - f_2(x,y)|^2,$$

where I_1 and I_2 are two blocks of an image, and $f_1(x, y)$ and $f_2(x, y)$ denote the gray levels of pixel (x, y) on blocks I_1 and I_2, respectively. It is obvious that $(X,$

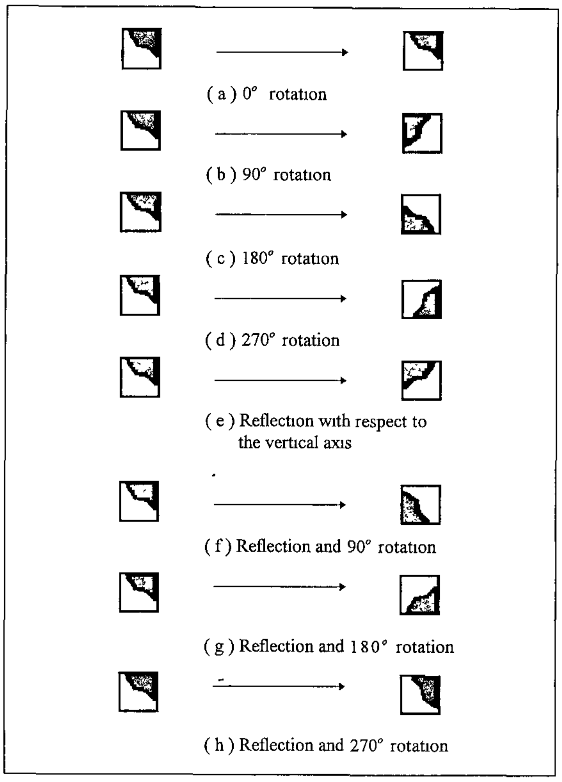

(a) 0° rotation
(b) 90° rotation
(c) 180° rotation
(d) 270° rotation
(e) Reflection with respect to the vertical axis
(f) Reflection and 90° rotation
(g) Reflection and 180° rotation
(h) Reflection and 270° rotation

Fig. 1. Eight affine symmetry rotations.

δ) is a metric space. The metric δ is used to determine the distance between pairs of blocks and is used as a measure of block similarity.

A natural image does not appear to contain affine transformations of itself, but it seems to be composed of copies of properly transformed parts of itself. To find the local IFS codes of an image, we partition the image into N non-overlapped blocks, called range blocks. For each range block R_i, we wish to select a portion of the image, called domain block D_i, such that the result of applying some affine transformation over D_i is most similar to the range block. The affine transformation should be contractive; hence the domain block is larger than the range block by, in general, four times. For gray-level images, the affine transformation w between a range block and a domain block is modified as follows:

$$w\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}a&b&0\\c&d&0\\0&0&s\end{pmatrix}\begin{pmatrix}x\\y\\z\end{pmatrix}+\begin{pmatrix}e\\f\\o\end{pmatrix}, \qquad (1)$$

where z denotes the gray-level value of pixel (x, y), and s and o are concerned with the gray-level transformation between the two blocks. The transformation w is contractive if $s<1$.

To find the most similar domain block for each range block R_i, $i=1, 2, ..., N$, we search through all the overlapping domain blocks of the image to find a domain block D_i which satisfies the minimum of the quantity:

$$\delta(R_i,\ w_i(D_j)) \text{ for } j=1, 2, ..., M. \qquad (2)$$

To compute δ, the domain block should be subsampled to the same size as the range block. There are 8 ways to rotate one square block to another, as shown in Fig. 1. Minimizing. Equation (2) means finding a domain block D_i that most looks like R_i after applying some rotation as shown in Fig. 1 and a gray-level transformation. A choice of D_i and the corresponding rotation, along with the parameters s_i and o_i, determine a transformation w_i in the form of Eq. (1). Let R_i and the subsampled and rotated D_i contain n pixel intensities $q_1, q_2, ..., q_n$ and $p_1, p_2, ..., p_n$, respectively. From Eq. (2), parameters s_i and o_i of w_i are selected to minimize the quantity:

$$T=\sum_{j=1}^{n}(s_i\times p_j+o_i-q_j)^2. \qquad (3)$$

The minimum of T occurs when its partial derivatives with respect to s_i and o_i are zero. Hence, we have

$$s_i=(n(\sum_{j=1}^{n}p_jq_j)-(\sum_{j=1}^{n}p_j)(\sum_{j=1}^{n}q_j))$$

$$/(n\sum_{j=1}^{n}p_j^2-(\sum_{j=1}^{n}p_j)^2)$$

$$o_i=(\sum_{j=1}^{n}q_j-s_i\bullet\sum_{j=1}^{n}p_j)/n.$$

If $n\Sigma_{j=1}^{n}p_j^2-(\Sigma_{j=1}^{n}p_j)^2=0$ (that is, R_i is a plain block), then s_i=0 and $o_i=(\Sigma_{j=1}^{n}q_j)/n$. Once we have the collection of transformations $w_1, w_2, ..., w_N$, the encoding procedure finishes.

B. Decoding

The decoding procedure decompresses an image from a set of local IFS codes. An arbitrary image is initialized first in the decoding procedure. Its content is unimportant; it can be data, a natural image, or anything. This initial image is the same size as the original image in encoding. In the first iteration of decoding, the initial image is partitioned into non-overlapping blocks which are the same size as the range blocks in encoding, and which are also called range blocks. For each range block, we read the corresponding local IFS codes (that is, the affine transformation parameters from the stored packed file), and locate the domain block that is the same size as the domain block in encoding. Then, the domain block is subsampled into the same size as the range block and its content is mapped to the range block by the affine transformation. Once the content of each range block is replaced by the transformed content of its corresponding domain block, the reconstructed image in the first iteration is obtained. In the second iteration, the reconstructed image obtained in the first iteration is used as the initial image, and the same process carried out in the first iteration is performed. The obtained reconstructed image in this iteration is then used as the initial image in the later iteration. The process is repeated until the difference between the initial image and the reconstructed image is indiscernible, and the last reconstructed image is the decoded image. For simplicity of programming, blocks are usually rectangular or triangular in shape. Experimental results of compressing 256×256 images are shown in Figs. 2-4. In these examples, the blocks in the shape of triangles are chosen. The sizes of triangular range blocks and domain blocks are a quarter of 16×16 and 32×32, respectively. The compression ratios in these examples are about 16.2:1 when no data-packing method is used. The compression ratio is determined by the ratio of the memory required to store the IFS codes to the memory required to store the original image. The peak signal-to noise ratio (PSNR) of the decompression image is about 26.19. In order to reach a high image compression ratio, IFS codes may be stored into a packed file by using, for example, Kraft's coding theorem (Barnsley and Hurd, 1993).

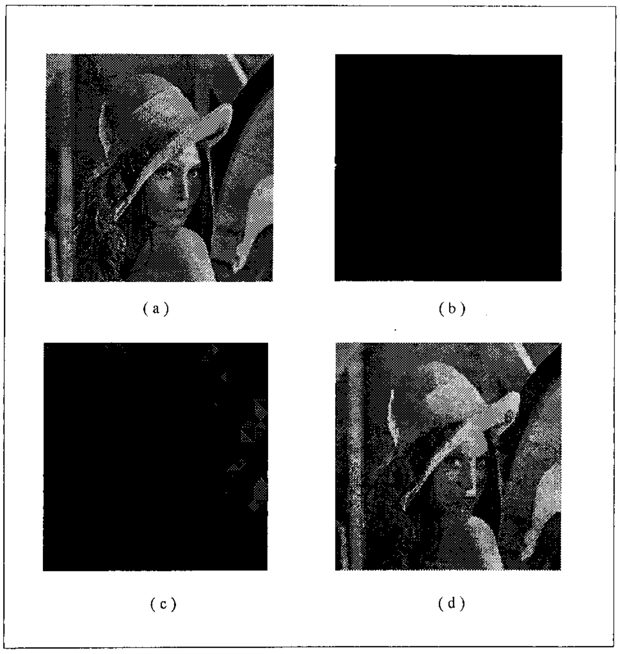


Fig. 2. An example of fractal image compression: (a) original image; (b) initial image; (c) image reconstructed after 1 iteration; (d) image reconstructed after 20 iterations.

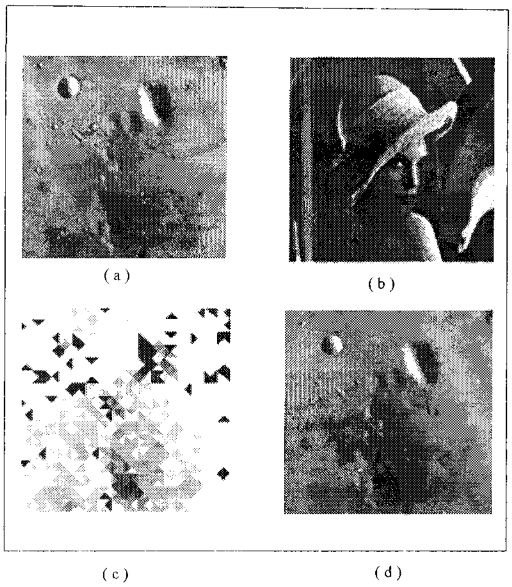


Fig. 3. An example of fractal image compression: (a) original image; (b) initial image; (c) image reconstructed after 1 iteration; (d) image reconstructed after 10 iterations.

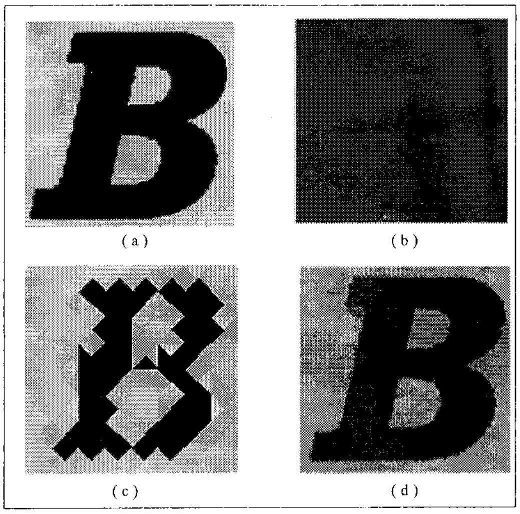


Fig. 4. An example of fractal image compression: (a) original image; (b) initial image; (c) image reconstructed after 1 iteration; (d) image reconstructed after 10 iterations.

III. Fractal-based Image Processing

After compressing an image using the fractal scheme, we obtain a set of local IFS codes of this image. In this section, we wish to study whether there exist characteristics of these codes which are useful for image processing.

1. Speed Improvement by N-Step Matching Method

In the encoding process, it is most computationally intensive to search the most similar domain block for each range block. To speed up this work, the N-step matching method (Koga *et al.*, 1981; Lee *et al.*, 1994) is used. This method uses coarse-to-fine searching to find the location of the best matching. In the first step, an approximate location is detected from several trial locations that are coarsely spaced. In the second step, a more accurate location is detected from fewer trial locations that are less coarsely spaced around the approximate location. The third step is a repetition of the second step, and the distance between trial points becomes smaller than before. This process continues until all N steps have been evaluated.

The N-step algorithm with N=3 is described briefly through an example in Fig. 5. In the first step, let the coordinates of the starting point be (i, j). Around the point (i, j), $(2\times3+1)^2$ trial points, which are three relatively equally differing grids indicated by the circle marks in Fig. 5 are tested. For each trial point (p, q), we use the value of T in Eq. (3) as a measure of dissimilarity between the image blocks located in (i, j) and (p, q), respectively. After computing the value of T for each trial point, we choose the point with the minimum T to be the starting point of the next step. In this example of Fig. 5, the point found is $(i+3, j+3)$. In the second step, point $(i+3, j+3)$ is the starting point; $(2\times2+1)^2$ trial points, which are indicated by the rectangle marks in Fig. 5, are two equally differing grids around the starting point. In this step, the point with minimum T is $(i+3, j+5)$. In the third step, the starting point is $(i+3, j+5)$, and $(2\times1+1)^2$ trial points, which are indicated by the cross marks in Fig. 5, are checked. The coordinates of the point $(i+2, j+6)$ have the minimum T value, and so it is considered to be the location of the best matching.

This method has been implemented on a DEC 3000 machine. Experimental results of taking N as 5, 10, 15, and 20 in the N-step algorithm for image compression are shown in Fig. 6, where the images are of size 256×256. The encoding time for each case is also given in this figure. The smaller the N value was, the less was the encoding time used, and the worse was the reconstructed image. Note that if exhaustive searching was used to find the most similar domain block of each range block, then the encoding time was about 3 hours.

2. Noise Reduction

If an image is corrupted with noise, what will be the result after reconstruction from the local IFS codes of the noisy image? We find that the noise has been blurred in the reconstructed image. If the noise in an image is not extensive, then it won't affect the searching for the most similar domain block for each range block. That is, the coefficients of the affine transfor-

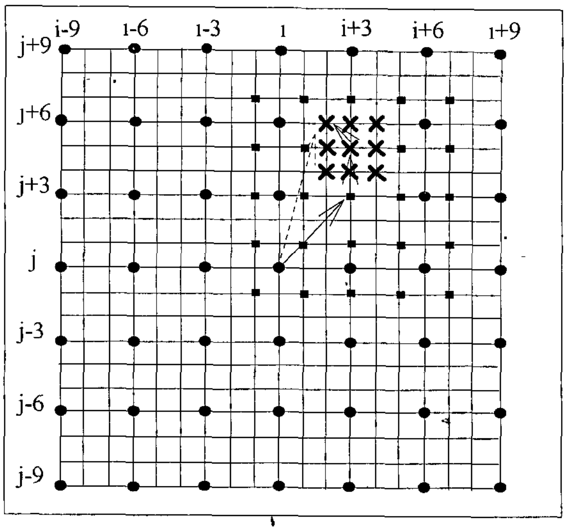
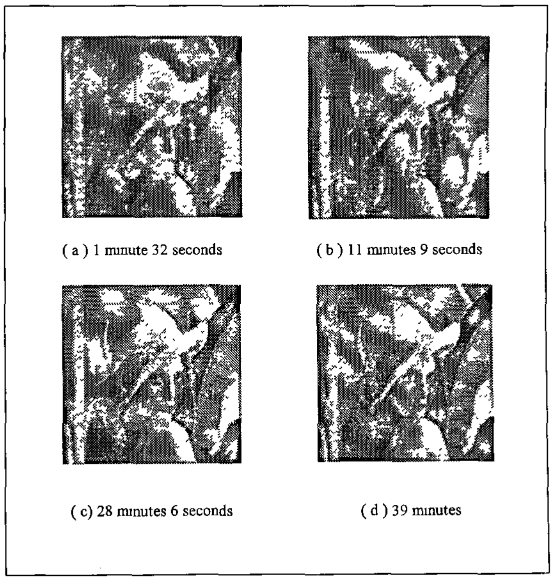


Fig. 5. An example which illustrates the N-step algorithm with N=3. The points found in the three steps are $(i+3, j+3)$, $(i+3, j+5)$ and $(i+2, j+6)$, respectively. Point $(i+2, j+6)$ is the final matching result.



(a) 1 minute 32 seconds

(b) 11 minutes 9 seconds

(c) 28 minutes 6 seconds

(d) 39 minutes

Fig. 6. Results of using the N-step matching algorithm to speed up compression: (a) N=5, PSNR=21.82 dB; (b) N=10, PSNR=23.70 dB; (c) N=15, PSNR=24 83 dB; (d) N=20, PSNR=25.06 dB.

mations found are almost the same as those found in compressing the original noiseless image. Hence, the reconstructed image will be almost no different from the original noiseless image. This is the noise reduction effect of the fractal image compression scheme, as shown in Fig. 7. If much noise appears in an image, the reconstructed image is somewhat noisy, but the noise has been blurred, as shown in the example in Fig. 8.

3. Edge Detection

If a given image is encoded to a set of local IFS codes, a straightforward approach to extracting the edges of the image might be to decode the image first and then use an edge detector, such as the Sobel edge detector, to extract its edges. That is, image decoding and edge extraction are treated as two separate operations. This would result in an increase of the computational requirements.

In fact, edge detection tasks can be implemented simultaneously with the decompression procedure through the following steps. First, the IFS codes o of all range blocks are set to an arbitrary small gray-level value o', such as $o'=5$. The selection of o' does not affect the result significantly, but o' cannot be 0. Second, we set the IFS codes s to 0.0 for certain range blocks whose s values are far from 1, or not within a specific interval, such as [0.90, 1.20]. This interval is specified by the user; the wider the range is, the more edges are detected edges. Third, we perform the decoding procedure with an arbitrary initial image, and then choose as the edge points those pixels whose gray levels are approximately the value o' multiplied by the number of iterations, n, that have been performed in decoding.

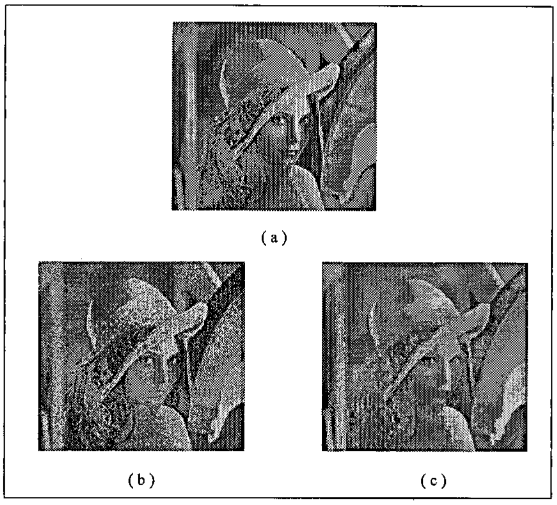


Fig. 7. An example of noise reduction: (a) original image; (b) image with Gausian noise; (c) reconstructed image.

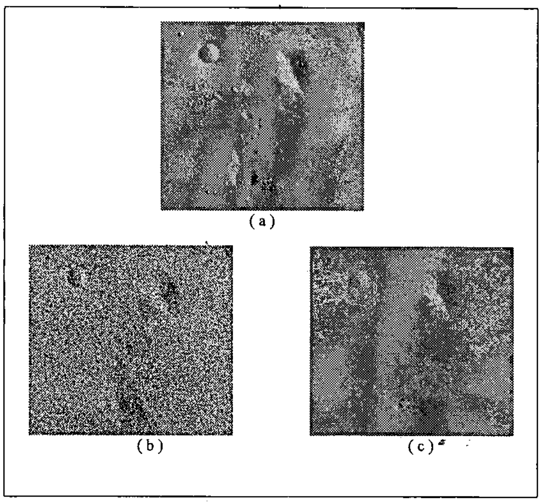


Fig. 8. An example of noise reduction: (a) original image; (b) image with Gausian noise; (c) reconstructed image.

Before decoding, if the s value of a range block was about 1.0, we found from the experiments that this range block included edge points in general. For those range blocks whose s values were far from 1.0, their s values were reset to 0 in the first step. Hence, after one iteration of processing in decoding, their gray levels were $g \cdot s+o'=o'$, where g is the gray-level value of the initial image. After n iterations, their gray levels were also approximately o'. Then, those pixels with gray levels which were approximately $o' \cdot n$ were the edge points. Figure 9 illustrates the edge detection results of several gray-scale images using the fractal scheme.

4. Edge Magnification and Demagnification

If we select the sizes of the range and domain blocks in decoding to be four times the sizes of the range and domain blocks in encoding, respectively, then the reconstructed image will be four times larger than the original reconstructed image. On the other hand, if the blocks in decoding are of smaller in size, then the reconstructed image becomes smaller. So, if we wish to magnify or demagnify the edges of an image, we select the appropriate sizes of the range and domain blocks, and use the procedure mentioned in the previous section to modify the local IFS codes in decoding. An example is shown in Fig. 10. Obviously, magnification or demagnification of edges in this way is a feasible way to generate non-jagged edges.

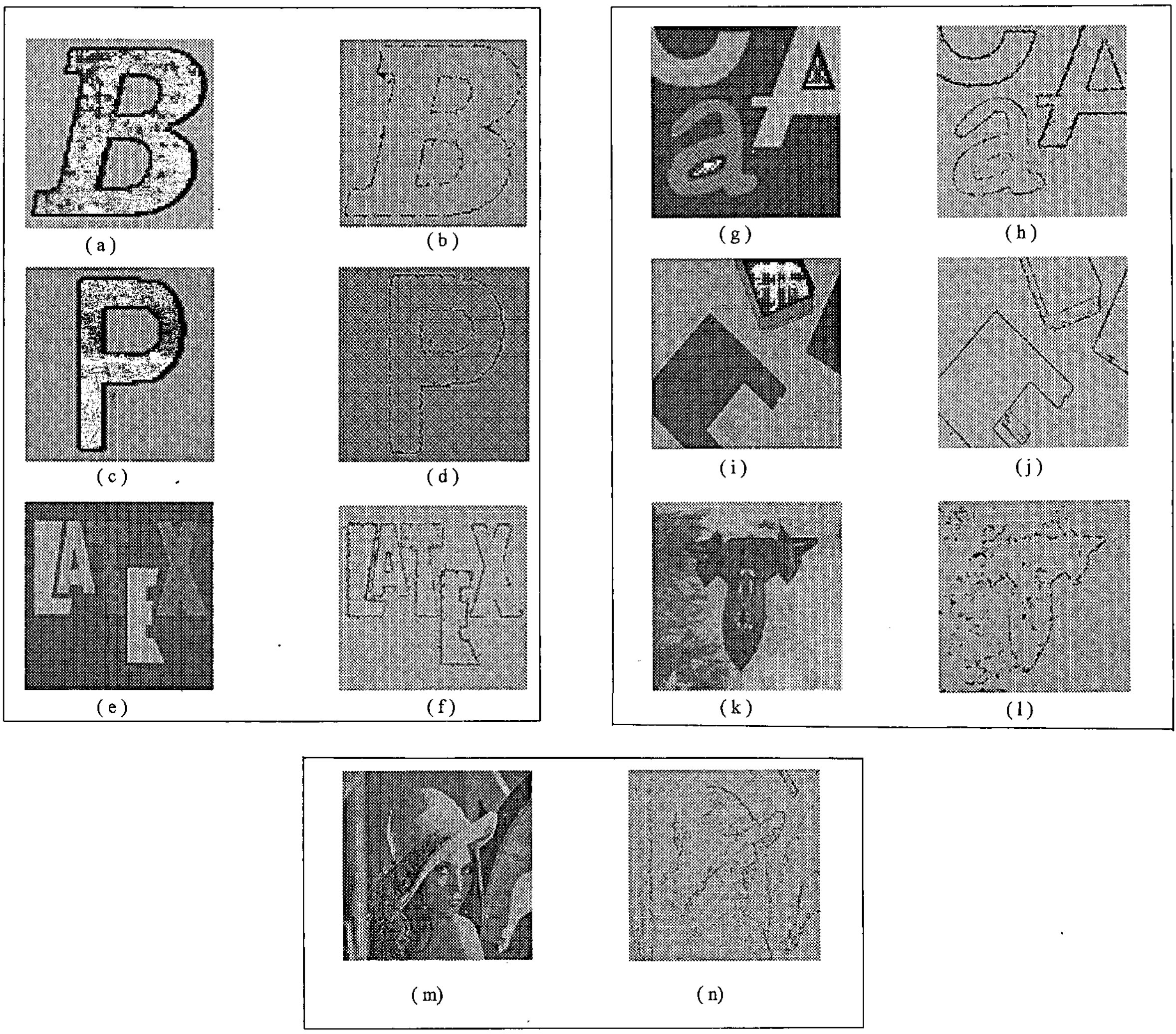


Fig. 9. Edge detection (Image size: 256×256).

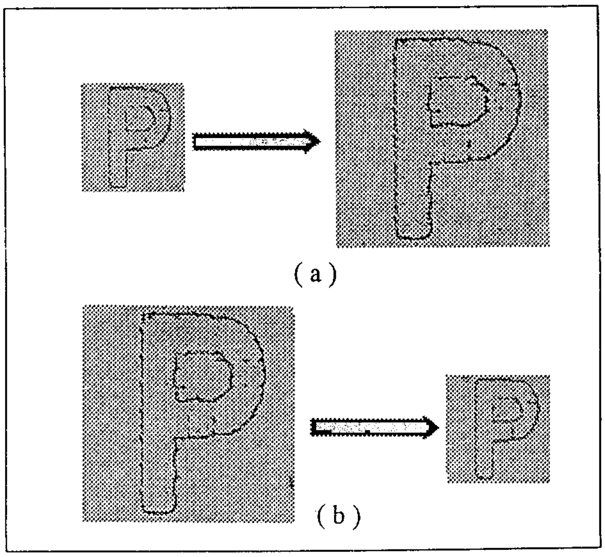
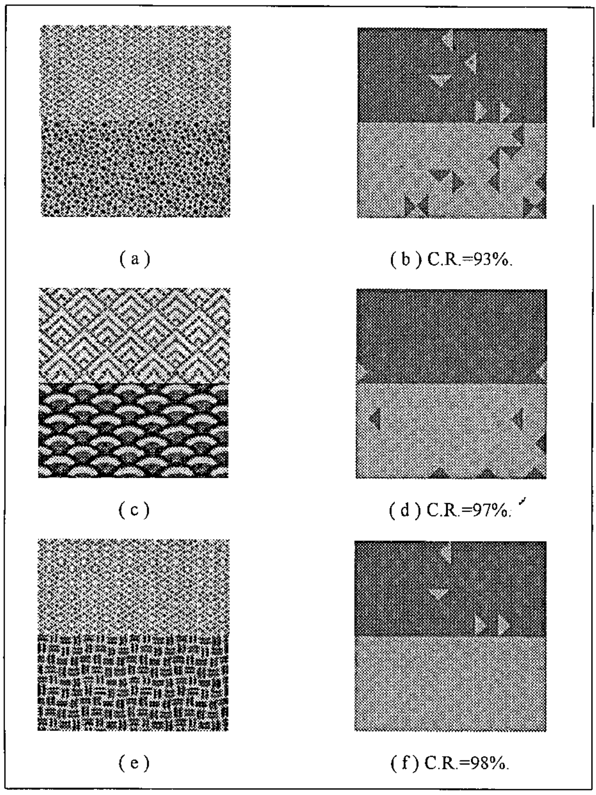


Fig. 10. Edge magnification and demagnification: (a) edge magnification; (b) edge demagnification.

5. Texture Segmentation

Another application of fractals is texture segmentation. New features of fractals based on IFS codes are proposed here. In the proposed approach, the IFS codes *s* and *o* of each range block, obtained in the encoding procedure, are directly used as features, and the c-means algorithm (Devijver and Kittler, 1982), for example, is used as the classification scheme to classify the range blocks. Some examples of block segmentation for textures are given in Fig. 11, where each range block is triangular in shape. Shown in Fig. 11(a), (c), (e), (g), (i) and (j) are six original images. Their corresponding segmentation results by using the c-means algorithm with a given priori c are shown in Fig. 11(b), (d), (f), (h), (j) and (k) accordingly, where blocks classified in the same cluster are denoted by the


(a)
(b) C.R.=93%.
(c)
(d) C.R.=97%.
(e)
(f) C.R.=98%.

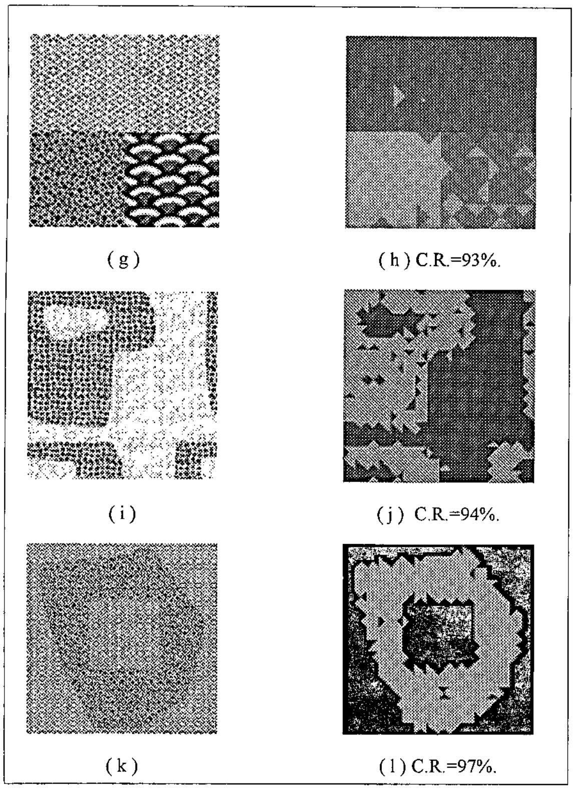

(g)
(h) C.R.=93%.
(i)
(j) C.R.=94%.
(k)
(l) C.R.=97%.

Fig. 11. Texture segmentation (C.R. denotes classification rate; the image size of (a), (c), (e) and (g) is 128×128 and that of (i) and (k) is 256×256).

same gray level to illustrate the result of segmentation. The average classification rate in this example is 95.33%, which is the ratio of the number of blocks correctly classified to the total number of blocks. The primitive blocks for encoding/decoding or classification in the example are triangles. Hence, the boundary between different textures may have a triangle blocking effect. The local IFS codes can be considered good features for coarse texture segmentation. If improvements are expected, we can use smaller primitive blocks in classification or use another texture feature for finer segmentation.

IV. Conclusions

In image processing and recognition, the amount of storage for an image is large, and so is the computational cost. However, if an image is represented by local IFS codes, the storage space is reduced; thus, the amount of computation required for the image can also be reduced. In this paper, we have studied the feasibility of processing images on the basis of fractals or local IFS codes. After compressing an image, some image processing tasks can be implemented simultaneously with the decompression procedure. In other words, when the image is reconstructed, the image processing tasks also are completed. These processing tasks include noise reduction, edge detection, edge magnification or demagnification. In addition, texture classification can be achieved by means of the features of local IFS codes. Some experimental results have shown that the IFS codes are promising features for texture segmentation.

Further researches may be directed to the following topics. First, a parallel processing environment can be designed to speed up the fractal image compression scheme, especially block matching in encoding. Second, more characteristics of local IFS codes may be found, and thus, additional applications of fractals in image processing can be studied.

Acknowledgment

This work was supported partially by the National Science Council, the Republic of China, under Grant NSC 82-048-E-007-304.

References

Barnsley, M. F. (1993) *Fractals Everywhere*, 2nd Ed., Academic Press, San Diego, CA, U.S.A.

Barnsley, M. F. and D. S. Alan (1988) A Better Way to Compress Images. *BYTE*, **13**(1), 215-223.

Barnsley, M. F. and L. P. Hurd (1993) *Fractal Image Compression*, AK Peters, Wellsley, Massachusetts, U.S.A.

Chen, S. F. (1993) *Fractal-based image analysis*. Master Thesis.

National Tsing Hua University, R.O.C.

Devijver, P. A. and J. Kittler (1982) *Pattern Recognition*, Prentice-Hall, London, U.K.

Fisher, Y. (1992) Fractal Image Compression. *SIGRAPH* 92' Course Notes.

Jacquin, A. E. (1992) Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, **1**(1), 18-30.

Kawamata, M., H. Kanbara, and T. Higuchi (1992) Determination of IFS codes using scale-space correlation function. PP. 219-233. *Proc. of IEEE Workshop on Intelligent Signal Processing and Communication Systems*, Taipei, Taiwan, R.O.C.

Koga, T., K. Linuma, A. Hirano, and T. Ishiguro (1981) Motion-Compensated Interframe Coding for Video Conference. pp. G5.3.1-G5.3.5. *Proc. IEEE National Telecommunication Conference*, New Orleans, Louisiana, U.S.A.

Lee, P. S., Y. E. Shen, and L. L. Wang (1994) Model-based location of automated guided vehicles in the navigation sessions by 3D computer vision. *Journal of Robotic Systems*, **11**(3), 181-195.

Louisa, A. (1993) Fractal image compression. *BYTE*, **18**(11), 195-202.

Mandelbrot, B. B. (1982) *The Fractal Geometry of Nature*, W. H. Freeman and Co., San Francisco, CA, U.S.A.

Monro, D. M. (1993) Class of fractal transforms. *Electronic Letters 18th*, **29**(4), 362-363.

Monro, D. M. and F. Dudbridge (1992) Fractal block coding of images. *Electronic Letters 21st*, **28**(11), 1053-1055.

Pei, S. C., C. C. Tseng, and C. Y. Lin (1993) Wavelete transform and scale-space filtering of fractal images. IPPR Conf. on Computer Vision, Graphics and Image Processing, pp. 50-57. Nantou, Taiwan, R.O.C.

Pei, S. C., C. C. Tseng, and C. Y. Lin (1992) A novel decoding algorithm for IFS codes. IPPR Conf. on Computer Vision, Graphics and Image Processing, pp. 121-128. Nantou, Taiwan. R.O.C.

碎形在影像處理上的研究

高興同　王玲玲[†]

清華大學資訊科學研究所

摘　要

碎形影像壓縮方法是一種新穎的影像壓縮技術，並且在影像壓縮的領域上，表現出它出色的處理能力。此種影像壓縮方法可產生壓縮率超過一萬比一的情況。在這方法下，一張影像不再以一序列的像素矩陣來表示，而是以一組排列相當緊密的數碼，稱作反複函數系統數碼，來代表影像的各個分割區的資訊，然後再由這些分割區的資訊建構出完整影像。在本篇論文中，我們提出利用N步搜尋方法來加速碎形影像的壓縮速度，並探討在反複函數系統數碼上作影像處理之可行性。當一張影像被壓縮成反複函數系統數碼後，一些影像處理的工作就可以接著進行。這些工作包括雜訊減少，邊界偵尋，邊界之放大與縮小等，都可以在碎形影像壓縮之解壓縮的過程中同時進行。此外，反複函數系統數碼對紋理分割工作而言，是一組良好的特徵值，故可直接用來作紋理分割工作。由論文中所提供實驗結果顯示，我們所提出的方法的確具有可行性。