영상 압축을 위한 유사성 함수 연구

주 우 석[†] · 강 종 오[††]

요 약

프랙탈 영상압축은 블록단위의 비교방식을 사용함으로써 기존의 압축방법에 비해 획기적으로 증대된 압축율을 보인다. 소프트웨어에 의한 실시간대 재생이 가능함에도 불구하고 프랙탈 영상압축의 실용화에 관건이 되는 것은 압축에 소요되는 시간이다. 본 논문에서는, i) 영상내부의 차원정보 추출 및 군집화에 의해 유사블럭 탐색 횟수를 줄임으로써 압축에 소요되는 시간을 최소화시키는 알고리즘과 ii) 영상내부의 휘도 및 명암대비 정보를 사용하여 재생영상의 화질을 증가시키는 알고리즘을 제안하고 검증함으로써, 고속이면서도 상대적으로 고화질을 유지할 수 있는 방법을 제시하였다.

Study on the Similarity Functions for Image Compression

Woo-Seok Joo[†] · Jong-Oh Kang[††]

ABSTRACT

Compared with previous compression methods, fractal image compression drastically increases compression rate by using block-based encoding. Although decompression can be done in real time even with softwares, the most serious problem in utilizing the fractal method is the time required for the encoding. In this paper, we propose and verify i) an algorithm that reduces the encoding time by reducing the number of similarity searching on the basis of dimensional informations, and ii) an algorithm that enhances the quality of the restored image on the basis of brightness and contrast information. Finally, a method that enables fast compression with little quality degradation is proposed.

1. Introduction

Fractal image compression is one of the most promising image compression methods today. The method can show a compression rate ranging up to a thousand. Despite such high compression rate, it can also reduce the loss of restored images to a reasonable and acceptable amount. This advantage makes it useful for storing huge amount of image data with minimum memory space. Such drastic reduction in memory requirements can greatly reduce network transmission time, enabling real time processing of networked multimedia applications.

Fractal image compression is applicable both for static images and motion pictures. It is emerging as a new standard image compression technique.

Originally developed by Mandelbrot, fractal theory treats nature merely as a feedback system. By itera-

※This research was supported by Korea Science and Engineering Foundation with idenfitication no. 951-0902-106-2.
† 정 회 원 : 명지대학교 컴퓨터공학과 부교수
†† 정 회 원 : 명지대학교 컴퓨터공학과 대학원
논문접수 : 1997년 2월 24일, 심사완료 : 1997년 7월 28일

tively applying certain feedback rule, the shape of the nature, known as a fractal, can easily be determined. Evolution from original shape to final shape is governed by an orderly feedback rule. As the chaos theory claims, the final fractal shape is deterministically regular although apparently chaotic. In fractal image compression, any image is regarded as a fractal. Resemblances among parts composing an image are used to extract the feedback rule. During compression, the only information saved is the feedback rule. Consequently, one part of the image can be decompressed from other parts by iteratively applying the saved feedback rule.

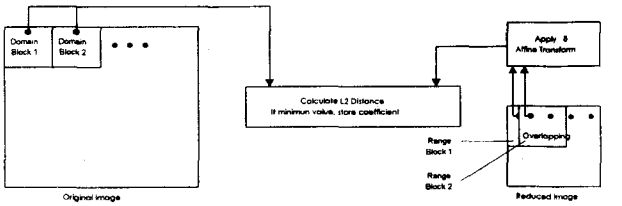
In this paper, we present two new algorithms for the extraction of the feedback rule. The first one is designed for the reduction of the compression time, and the second one for the enhancement of the restored image quality. In addition, each algorithm is compared with previous fractal compression methods in terms of its performances. The remainder of this paper is organized as follows: Section 2 provides the review of the previous fractal compression methods. Section 3 describes the concept and method behind our first algorithm, followed by performance evaluation. Section 4 provides the concept and method behind our second algorithm, followed by performance evaluation. Performance of the combination of the two algorithm is also evaluated. Finally, we provide our concluding remarks in section 5.

2. Preliminaries and Previous Work

Being one of the various block based coding methods, fractal coding claims that the iteration of simple deterministic mathematical procedures can generate images with infinitely intricate geometries. This claim, known as IFS(Iterative Function System) [1, 2, 4, 10] is mathematically based on the contraction mapping principle as follows: A mapping f is a contraction of the metric space X, provided that there is a constant $c(0 \le c < 1)$, such that for all x, y in X one has that $d(f(x), f(y)) \le c \cdot d(x, y)$. The distance d between elements is measured in Hausdorff distance. Let $a_0, a_1, a_2, \ldots$, be a sequence of elements from a metric space X defined by $a_{n+1} = f(a_n)$, with f being a feedback process. The contraction mapping principle states that i) there is a unique attractor $a_\infty = \lim_{n \to \infty} a_n$, and ii) a_∞ is invariant such that $f(a_\infty) = a_\infty$.

Hutchinson applied the theory to the area of image coding by defining operator W, [13, 14]. Denoting an image as A, the Hutchinson operator W is defined as $W(A) = W_1(A) \cup W_2(A) \cup W_3(A) \cup \ldots \cup W_N(A)$, where the feedback transformation W_i are contractions. As a union of individual transformation, the operator W is also proven to be a contraction mapping. If we let $A_{k+1} = W(A_k)$, $k = 0, 1, 2, 3, \ldots$, the generated image sequence will converge to a distinguished image, known as the attractor A_∞ of IFS. It is invariant under further transformations: $W(A_\infty) = A_\infty$. This means that final decoded image produced by iteratively applying the transformation W is convergent and unique. However, to apply the theory to the image coding, the inverse problem exists. The problem is how to derive the transformation W during the encoding.

Originally proposed by Barnsley and implemented by Jacquin et al. [3, 4, 5, 7, 8, 9], PIFS(Partitioned Iterated Function System) encoding is done as follows: i) subdivide original image into square blocks of finite size. ii) measure similarities between the blocks. iii) determine the transformation functions between the most similar block pair. The transformation function and the number of the corresponding block pair make up the encoded file.



(Fig. 1) Relation between Domain Block and Range Block

PIFS subdivides the original image into non-overlapping domain blocks as in (Fig. 1). Also, the same image is reduced by a factor of 2, and range blocks are defined on this reduced image. The size of the domain block and range block is the same. But the range blocks are made to overlap as it shifts right, so that every possible block can appear. For each domain block, similarity searching is performed to find the most similar range block.

PIFS uses affine transformation functions [15, 16]. The inherent linearity of the affine transformations makes the contraction property maintained during decoding process. To simplify further, the transformations are limited to identity, translation, rotation, and reflection. In practice, there are eight possible transformations depending on the angle of the rotation and axis of the reflection. The affine transformation W_i is defined as $W_i=\begin{bmatrix}x'\\y'\end{bmatrix}=\begin{bmatrix}a_i & b_i\\c_i & d_i\end{bmatrix}=\begin{bmatrix}x\\y\end{bmatrix}+Q_i$. Pixels at (x, y) of the domain block can be mapped into (x', y') of the range block by the transformation. a_i, b_i, c_i, d_i designate the corresponding type of affine transformation, and Q_i represents the average intensity difference between the blocks.

Similarity searching is the most important factor in PIFS. It determines the encoding speed and the quality of the restored image. All transformations are applied to every range block, and the transformed block with minimum distance from given domain block is selected. This distance, known as l_2 distance, is defined as $\sum_{x,\,y}^{N}(P_{D(x,\,y)}-P_{R(x,\,y)})^2$. Here, N denotes the block length in pixels, and $P_{D(x,\,y)}$ denotes the pixel intensity at (x, y) in the domain D. In PIFS, l_2 distance is used extensively to measure the similarity. In a rigorous sense, l_2 distance does not coincide with the Hausdorff distance. But the Hausdorff distance itself is hard to implement, and l_2 distance is used in place of it.

3. Proposed Algorithm I: Dimensional Similarity Algorithm

3.1 Concept and Method

In our dimensional similarity algorithm (DSA, hereafter), we make use of dimensional information during the similarity search. Based on the information, we limit the number of the range blocks to be compared. As a result, the encoding time reduces remarkably. As a side effect, the quality of the restored image degrades somewhat. Compromise between the speed and quality can be made by controlling involved parameters. By carefully adjusting the parameters, DSA can produce large reduction in encoding time with small degradation in image quality.

Fractal dimensions can take floating point values unlike general Euclidean dimension. For instance, the Koch curve in (Fig. 2) is 2.26 dimensional. Such dimensional values characterize and differentiate natural fractal images from artificial Euclidean images. By definition, the iterative feedback transformation is valid only when the dimensions match. In (Fig. 2), block 'a' cannot generate block 'c' under any transformation. In contrast, the union of the affine transformations can generates block 'b', which is also 2.26 dimensional. Extracting the dimension is useful in the similarity search.

Two problems exist in characterizing a block with the corresponding dimension. First, the fractal theory generally defines the dimension on the basis of boundary curve shapes. But the raster block image has regional pixel values. Second, a single block may be composed of multiple curves, which do not necessarily coincide in their dimensions. Moreover, the multifractal[13] curves may be intermingled inside the block complicating the extraction of individual dimension.

To solve the first problem, the box counting dimension is used instead of the more general fractal dimension. It is also a form of the fractal dimension, and we use it in DSA because of its automatic com-

putability. It coincides with the general fractal dimension in most cases, but not always. If we divide an image by a regular mesh with size s_1 and s_2, and let $N(s)$ be the number of grid boxes which contain some of the image, the box counting dimension D_b is defined as $D_b = (\log(N(s_2) - \log(N(s_1))/(\log(1/s_2) - \log(1/s_1))$.

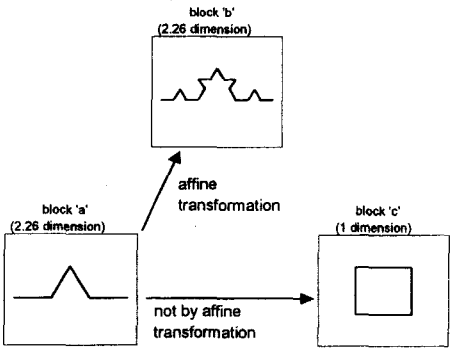
To solve the second problem, we assume that the pixels belonging to a single curve have similar intensities. Under the assumption, the multiple curves can be separated by the pixel intensities. We partition gray level ranges with disjoint set of classes. For instance, 256 gray levels can be partitioned into 4 classes such that $0 \le class_1 \le 63$, $64 \le class_2 \le 127$, and so on. Now, any block can be characterized by a vector with the number of components being equal to the number of classes. Then the block image $I = I_1 \cup I_2 \cup I_3 \cup ... \cup I_N$ where I_i is the subimage of ith class, and N is the number of classification.

Consequently, we define a characterization function $f: R \rightarrow R^N$ as $f(I) = D = (D_b(I_1), D_b(I_2), D_b(I_3), ..., (D_b(I_N))$ where $D_b(I_i)$ is the value of the box counting dimension of $\underline{ith}$ class subimage. Now, a block is characterized by a single vector in the N dimensional vector space whose axes correspond to multifractal elements inside the block. Moreover, we cluster the characteristic vectors by K-means algorithm [11, 17]. Denoting the number of clusters as K, the algorithm repetitively reassigns set element D such that $D \in S_j$ if $\|D - C_j\| < \|D - C_i\|$ for all $i = 1, 2, ..., K$ $i \neq j$. S_j denotes the set of samples whose cluster center is C_j. The clustering is performed in N dimensional vector space. This is shown in (Fig. 3). Once the blocks with similar characteristic vectors are clustered, DSA confines the range block searching only to the cluster whose dimensional values are comparable to the dimension of the given domain block.

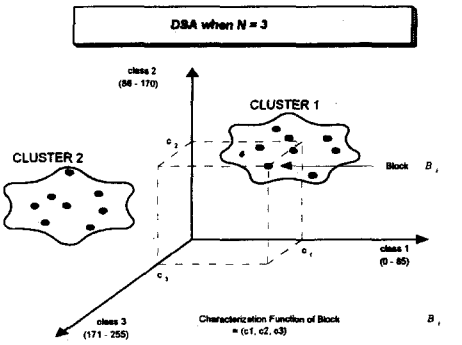
3.2 Performance Evaluation

3.2.1 Pseudocode Comparison

In terms of pseudocode, Barnsley et al. 's algorithm for similarity searching can be written roughly as follows:



(Fig. 2) Koch Curve and Fractal Dimension



(Fig. 3) Vector Space of DSA

```
partition image
adjust threshold
for each domain block
    for each range block
      apply transformations
      for transformed blocks
        calculate l2 distance
        update minimum-distance block
      endfor
    endfor
    Encode by the minimum distance block
endfor
```

In this case, most of the CPU time is spent on

evaluating the statements inside the innermost *for* loop. Recalling the transformations used in Barnsley et al.'s algorithm, the innermost loop is executed eight times. If we let the number of the domain block as N_D and that of the range block as N_R, the algorithm has the time complexity of $O(8\ N_D N_R)$. In DSA, the algorithm is modified as follows:

partition image
adjust threshold
for each domain/range block
  Locate into N dimensional vector space ①
  assort the range blocks into K clusters ②
for each domain block
  find matching cluster ③
  for the range blocks in the matching cluster ④
  .....

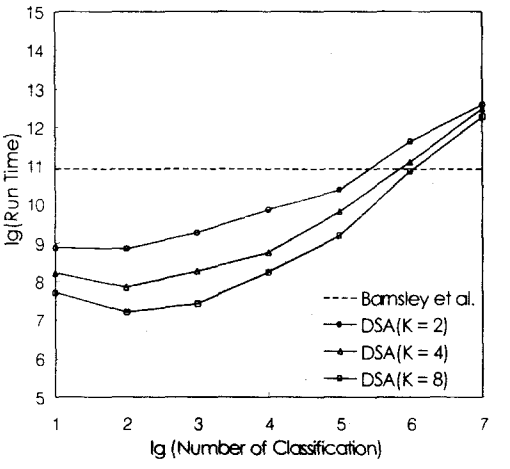
On the average, DSA algorithm can reduce the run time approximately by a factor of K. Statements ① and ② are evaluated just once, and adds no extra complexity in terms of the big Oh notation. Statement ③ can be run in linear time since there can be only K clusters. If the clustered pattern reveals a uniform distribution, the number of range blocks in statement ④ reduces to N_R/K. As a result, DSA will show the time complexity of $O(8\ N_D N_R/K)$, and the run time can be controlled by adjusting the parameter K.

3.2.2 Speed and Quality

Here, we compare DSA with Barnsley et al.'s algorithm with respect to their speed and quality. The encoding time required by Pentium 100MHz CPU was measured in seconds. Let us denote an original image as I, restored one as I', and the pixel intensity at(i, j)th position as $I(i, j)$. Given an wxw image, root mean square error is defined as $RMSE=\sqrt{\frac{1}{w^2}\sum_{i=1}^{w}\sum_{j=1}^{w}(I(i, j)-I'(i, j))^2}$. Peak signal to noise ratio is defined as $PSNR=20\ \log_{10}(255/RMSE)$ [12]. Three 128 X 128 Images, known as Baboon, Lenna, and Pepper are selected as sample inputs. Block size of 4 X 4 pixels is assumed.

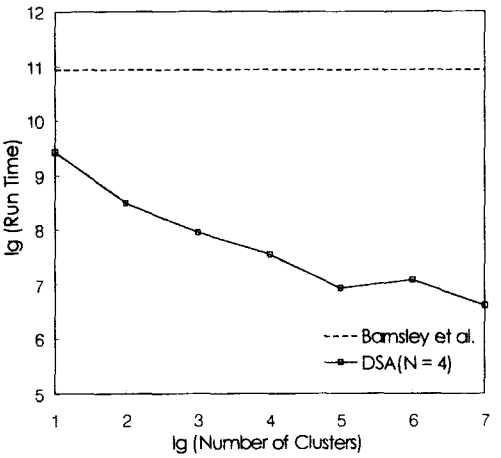
(Fig. 4) shows average encoding time as N, the number of classification, varies. Both axes are plotted in log scale with base 2. The number of clusters, K, is used as a parameter. Taking into account the log scale, the run time reduction is enormous. For instance, if we choose the parameters $N=K=4$, DSA reduces the run time required by Barnsley et al.'s, by a factor or 8.47. Also, as shown in (Fig. 5), the reduction factor grows up rapidly as K increases. However, when N approaches 2^6, the computational overhead caused by multiple vector components dominates our DSA, and the speed is degraded close to that of Barnsley et al.'s.

(Fig. 6) shows the quality behavior of DSA. PSNR of DSA approaches to that of Barnsley et al's, as N approaches 2^7. In 128-dimensional space, almost all blocks are widely dispersed, and uniform clustering cannot be expected. In case where $N=K=4$, DSA degrades PSNR of Barnsley et al.'s by a factor of 0.05, a relatively slight loss compared with the run time reduction factor. However, such degradation in PSNR does not always lead to visual degradation. In a sense, the visual proximity reproduced by the fractal dimension cannot be exactly measured by PSNR alone. Although the searching space is reduced by a factor of 8, inherent

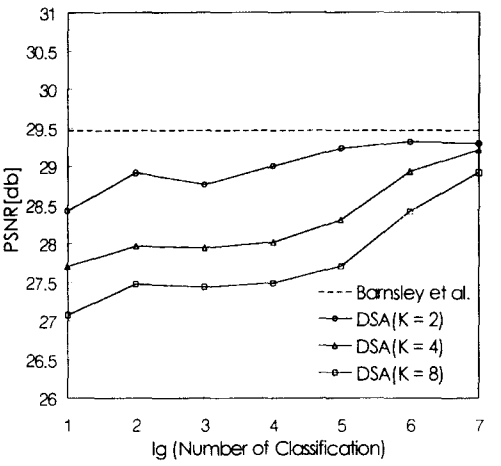


(Fig. 4) Run Time as a Function of N

fractal dimension of DSA still maintains high restoration capability as shown in (Fig. 8).

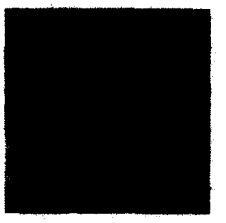


(Fig. 5) Run Time as a Function of K



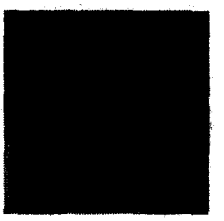
(Fig. 6) PSNR as a function of N

(Fig. 7.1) Original Baboon Image



(Fig. 7.2) Barnsley et al.'s

(Fig. 7.3) Differences



(Fig. 7.4) DSA(N = 4, K = 8)'s

(Fig. 7.5) Differences

4. Proposed Algorithm Ⅱ : Luminous Similarity Algorithm

4.1 Concept and Method

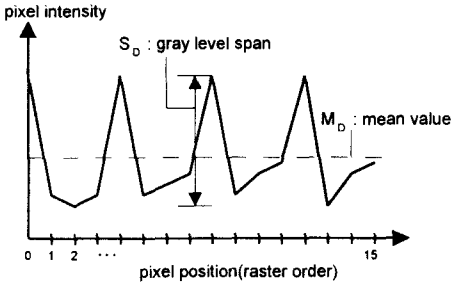
Luminous similarity algorithm (LSA, hereafter) incorporates luminance information into the similarity search. We differentiate brightness and contrast, and apply them separately before the similarity search. The pixel intensity in the original block is modified into another value according to the luminance information. The similarity search is done on the modified block.

Barnsley et al.'s algorithm applies the brightness information into the similarity search. In practice, the algorithm computes the mean pixel intensity for each block. Difference in the mean intensities between given pair of blocks is added or subtracted just before the similarity search. Therefore, it is also based on the modified range block.

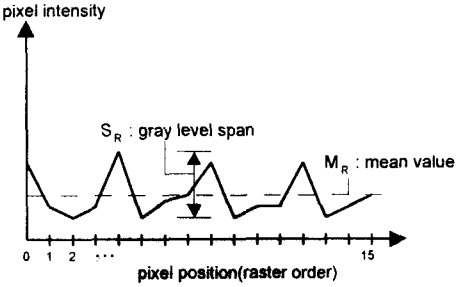
LSA extends Barnsley et al.'s algorithm to accommodate the contrast in addition to the brightness. Given a block, we define a gray level span as the difference between the maximum and minimum pixel intensities inside the block. If the gray level span of

the domain block is S_D, and that of the range block is S_R, every pixel in the range block is made to amplify its pixel intensity by a factor of S_D/S_R. However, in applying such transformation, brightness information must also be incorporated. In terms of (Fig. 8), the algorithm proceeds as follows:6i) the pixel intensities of the range block are decremented by M_R so that M_R coincides with the x-axis, ii) the pixel intensities are multiplied by S_D/S_R, iii) M_D is added to them so that M_R and M_D coincides. Now, l_2 distance evaluation can be proceeded with the modified range blocks.

LSA generally enhances the quality of the restored images. Especially when images have frequent variation in gray level, it works out best. Since both of the luminous transformations are affine, contraction mapping principle is preserved. The mean intensities and the gray level spans are evaluated once for all range blocks. As a result, added computational complexity increases only by a constant term. As for the compression rate, additional few bits are required to store the quantized ratio of the gray level span between the matching domain and range blocks.



(Fig. 8.1) Intensity Distribution of Domain



(Fig. 8.2) Intensity Distribution of Range

4.2 Performance Evaluation

4.2.1 LSA

In general, PSNR of LSA outperforms Barnsley et al.'s as shown in 〈Table. 1〉. Since the similarity searching is done on the modified blocks, the block selected as the matching block in Barnsley et al.'s algorithm may not be the matching block in LSA. The differences in the number of matching blocks are also shown in the table. More than half of the domain blocks have changed their matching range blocks. Direct consequence of this change results in the increase in the quality of restored image as shown in (Fig. 9).

Comparing (Fig. 9.2) and (Fig. 9.4), we find that LSA restores high frequency components far better than Barnsley et al.'s. The difference of (Fig. 9.1) and (Fig. 9.2) is shown in (Fig. 9.3), where most of the

〈Table 1〉 Quality Comparison of LSA and Barnsley et al.'s

Method / PSNR	Block Size : 4 X 4			Block Size : 8 X 8		
	Barnsley et al.	LSA	Block Differences	Barnsley et al.	LSA	Block Differences
PSNR Lenna	28.22	29.40	787/1024	22.92	23.72	197/256
PSNR Pepper	30.98	32.48	848/1024	23.98	24.80	189/256
PSNR Baboon	29.20	29.68	757/1024	24.60	24.25	197/256

outline information is retained. This means that Barnsley et al.'s algorithm relatively lacks in the ability to restore the outline information. On the contrary, LSA exhibits much better performance as shown in (Fig. 9.4) and (Fig. 9.5). The increase in PSNR is mainly caused by the ability of our LSA that traces the rapid change in the gray level.

(Fig. 9.1) Original Lenna Image

(Fig. 9.2) Barnsley et al.'s

(Fig. 9.3) Differences

(Fig. 9.4) LSA

(Fig. 9.5) Differences

4.2.2 DSA in Combination with LSA

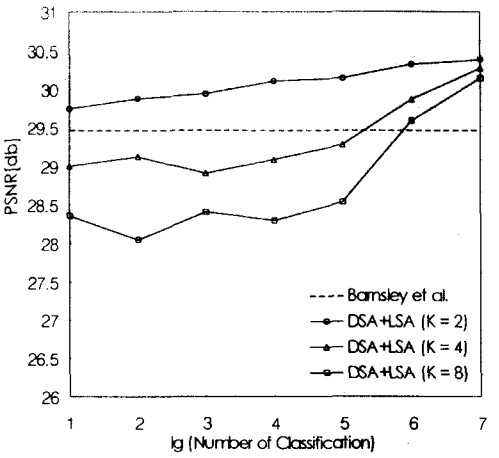
Since the merit of DSA is in its speed, and that of LSA is in its quality, the two algorithms can be combined to complement each other. Such combination can be done procedurally by first applying DSA in finding the dimensionally similar blocks and then applying LSA in modifying the block. The reverse order, LSA followed by DSA, is not used since the dimension defined on the modified block is meaningless. According to their inherent fractal dimensions, DSA sorts the blocks into certain number of clusters. Within each cluster, LSA modifies the pixel intensities according to the contrast and brightness. Performance of the LSA and DSA combination is shown in (Fig. 10) through (Fig. 12).

As (Fig. 11) shows, the encoding time is almost the same as that of DSA alone. Reduction in search space caused by DSA is so predominant that the calculation overhead required by LSA is almost negligible. On the contrary, PSNR generally increases above the level of DSA as shown in (Fig. 10). In case where $K=2$, PSNR continuously outperforms that of Barnsley et al.'s. If we limit the parameter N within the range of 32, the encoding time is also far below the level of Barnsley's. For the case of (Fig. 12.3), the pepper image encoded with parameters $K=N=2$ has the run time reduction factor of 3.6 compared with Barnsley et al's algorithm. Moreover, PSNR also has increased by a factor of 0.03.

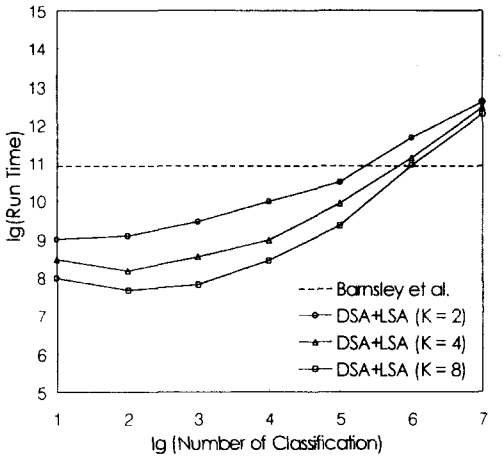
Although the performance largely depends on the specific characteristics and instance of input images, some compromise can always be taken. The compromise between quality and speed can be controlled by the parameters K and N. For instance, if we select the parameters $K=4$, $N=16$, we get the speed up factor of 14.35, and the quality degradation factor of 0.04.

5. Conclusion

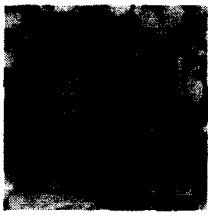
We have presented two efficient algorithms for the fractal encoding. DSA remarkably speeds up the encoding time by reducing the similarity search space. It was possible by clustering range blocks according to their inherent fractal dimensions. LSA enhances the quality of restored image by accommodating both contrast and brightness information inside range blocks. By combining the two algorithms, we could have the freedom to choose the level of speed and quality we need. This can be done by properly selecting the relevant parameters N and K while encoding. Depending on the selection, both the speed



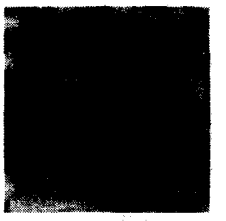
(Fig. 10) PSNR as a Function of N



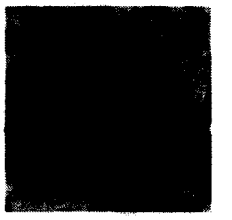
(Fig. 11) Run Time as a Function of N



(Fig. 12.1) Original Pepper Image



(Fig. 12. 2) Barnsley et al.'s
PSNR = 30.98 [db]
Run Time = 1914 [sec]



(Fig. 12.3) LSA Combined with DSA
PSNR = 31.88 [db]
Time = 523 [sec]

and quality can be improved compared with Barnsley et al's. Also, depending on the selection, the speed can be greatly increased at the expense of the slight quality loss expressed in terms of PSNR. Researches on the fractal coding is still relatively in its infancy, and further researches toward the improvements on the compression techniques should follow.

참 고 문 헌

[1] Barnsley, M., *Fractals Everywhere*, Academic Press, Boston, 1992.

[2] Dettmer, R., "Form and Functions: Fractal-based Compression," IEEE Review, pp. 323-327, Sep. 1992.

[3] Barnsley, M., and Sloan, A.D., "A Better Way to Compress Images," BYTE Magazine, pp. 215-223, Jan. 1993.

[4] Barnsley, M., and Hurd, L., *Fractal Image Compression*, AK Peters Ltd., New York, 1993.

[5] Barnsley, M. and Sloan, A., "Method and Apparatus for Processing Digital Data," United States Patent # 5,065,447.

[6] Beaumont, J.M., "Image and Data Compression Using Fractal Techniques," BYTE Magazine, vol.9, no.4, pp.321-328, Oct. 1991.

[7] Jacquin, A.E., "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformation," IEEE Trans. Image Processing, vol.1, no.1, pp.18-30, Jan. 1992.

[8] Monro, D.M., and Dudbridge, F., "Fractal Approximation of Image Blocks," ICASSP, pp. 485-488, 1992.

[9] Monro, D.M., and Dudbridge, F., "Fractal Block Coding of Images," Electronic Letters, vol.28, 11, pp.1053-1055, May, 1992.

[10] Mackeon, A., "Fractal Transform Image Compression," Electronic World and WirelessWorld, pp.203-211, Mar. 1992.

[11] Gonzalez, M., and Woods, R., *Digital Image Processing*, Addison Wesley, New York, 1992.

[12] Deller, J.R., Proakis, J.G., and Hansen, J.H.L., *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company, New York, 1993.

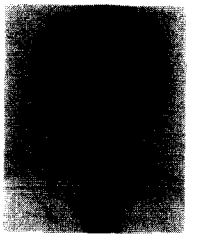
[13] Peitgen, H.O., Jurgens, H., and Saupe, D., Chaos and Fractals, Springer-verlag, New York, 1992.

[14] Peitgen, H.O., Jurgens, H., and Saupe, D., *Fractals for the Classroom*, Springer-verlag, New York, 1991.

[15] Foley, J., van Dam, A., Feiner, S., and Hughes, J., *Computer Graphics*, Addison Wesley, New York, 1990.

[16] Rogers, D.F., *Procedural Elements for Computer Graphics*, McGraw-Hill, New York, 1985.

[17] Tou, J.T. Gonzalez, R.C., *Pattern Recognition Principles*, Addison-Wesley, London, 1977.



주 우 석

1976년~1983년 서울대학교 공과대학 전자공학과(학사)

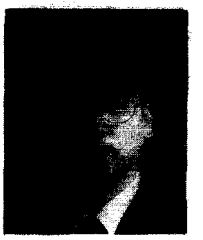
1983년~1985년 한국 IBM, 데이콤 정보통신연구소

1985년~1987년 University of Florida, 컴퓨터공학(석사)

1987년~1991년 University of Florida, 컴퓨터공학(박사)

1992년~현재 명지대학교 컴퓨터공학과 조교수

관심분야:컴퓨터그래픽스, 알고리즘, 데이타베이스



강 종 오

1992년 명지대학교 전자계산학과 졸업(학사)

1996년~현재 명지대학교 대학원 컴퓨터공학과 재학중(석사)

관심분야:컴퓨터 그래픽스, 멀티미디어, 데이터베이스