53206

TDA Progress Report 42-107

November 15, 1991

p-6

N92-14240

# A Comparison of the Fractal and JPEG Algorithms

K.-M. Cheung and M. Shahshahani
Communications Systems Research Section

*A proprietary fractal image-compression algorithm and the Joint Photographic Experts Group (JPEG) industry standard algorithm for image compression are compared. In every case, the JPEG algorithm was superior to the fractal method at a given compression ratio according to a root-mean-square criterion and a peak signal-to-noise criterion.*

## I. Introduction

Fractal image compression has attracted much publicity in recent years. It has been suggested that one can achieve compression ratios of the order of thousands to one by the application of fractal algorithms. Some researchers successfully generated certain images, with very small databases, by using fractal algorithms. These images generally consisted of natural objects, and the memory requirement for, e.g., a realistic looking tree, was about 120 bytes. However, the applicability of these methods to general image compression and the achievement of the phenomenal compression ratios of thousands to one have been viewed with general skepticism. In order to make a comparative study of the fractal versus Joint Photographic Experts Group (JPEG) standard algorithms, the authors sent ten images to a vendor, Iterated Systems Inc., Norcross, Georgia. These images were compressed by their proprietary fractal algorithms, and reconstructed using their decompression package. The compression ratios were from about five to one to twenty to one. The mean square errors and peak signal-to-noise ratio (SNR) were compared to the corresponding ones for the JPEG algorithms at the same compression ratios. The latter approach proved to be superior in every case according to these criteria.

The theoretical foundation and the practical implementation of the fractal method for image generation is described in Section II. The relevant portion of the JPEG algorithm is briefly described in Section III, and finally in Section IV the results of the comparative study are given.

## II. Fractals

There are various ways of defining fractals. The framework proposed by J. Hutchinson [1] has been the most successful approach for the study of fractals. To describe this method let $\mathcal{S} = \{S_1, \cdots, S_n\}$ be a finite set of affine transformations of $\mathbf{R}^q$. This means that if $x \in \mathbf{R}^q$, then the effect of $S_i = (A_i, v_i)$ on $x$ is given by

$$S_i(x) = A_i(x) + v_i$$

where $A_i$ is a linear transformation of $\mathbf{R}^q$ and $v_i \in \mathbf{R}^q$. It will be assumed that $A_i$'s are nonsingular and $S_i$'s are contracting, i.e., $||S_i(x) - S_i(y)|| < ||x - y||$ for all $x, y \in \mathbf{R}^q$. Then the affine transformation $S_{i_1} \cdots S_{i_k}$ is also contracting and, therefore, has a unique fixed point that will be denoted by $F_{i_1,\cdots,i_k}$. The fractal set $\mathcal{F}(\mathcal{S})$ associated to $\mathcal{S}$ is, by definition,

$$\mathcal{F}(\mathcal{S}) = \text{closure}(\{F_{i_1,\cdots,i_k} \mid \text{all } 0 \leq i_1, \cdots, i_k \leq n$$

$$\text{and } k = 1, 2, 3, \cdots\})$$

$\mathcal{F}(\mathcal{S})$ is a compact subset of $\mathbf{R}^q$. The *self-similarity* property of fractals is expressed by the fundamental equation

$$\mathcal{F}(\mathcal{S}) = \bigcup_{i=1}^{n} S_i[\mathcal{F}(\mathcal{S})]$$

In fact,

**Theorem 1.** From [1], $\mathcal{F}(\mathcal{S})$ is the unique compact set $K \subset \mathbf{R}^q$ with the property

$$K = \bigcup_{i=1}^{n} S_i(K) \qquad (1)$$

Notice that each $S_i(K)$ is a replica of $K$, so that Eq. (1) does indeed express the self-similarity property of fractals. This characterization of fractals is also important in practical applications.

**Example 1.** Let $K$ be a convex polygon in $\mathbf{R}^2$ with vertices $v_1, \cdots, v_n$, and let $S_1, \cdots, S_n$ be the affine transformations $S_i(x) = v_i + \alpha_i(x - v_i)$, where $0 < \alpha_i < 1$. If the $\alpha_i$'s are not too small, then $\bigcup S_i(K) = K$ and, consequently, $\mathcal{F}(\mathcal{S}) = K$ by Theorem 1 of fractal sets. Thus, every convex polygon can be realized as an $\mathcal{F}(\mathcal{S})$ for some $\mathcal{S}$. On the other hand, it is not hard to see that the boundary $\partial K$ of the convex polygon $K$ *cannot* be realized as a fractal set.

To generate a fractal image, one starts with a set $\mathcal{S} = \{S_1, \cdots, S_n\}$ of affine transformations. For every product $S_{i_l} \cdots S_{i_2} S_{i_1} = S = (A, v)$ of length $l \leq k$, for some pre-assigned value $k$, one computes its unique fixed point $F_{i_l}, \cdots, F_{i_1} = F = (I - A)^{-1}(v)$. Note that since $S$ is contracting, $I - A$ is invertible. The coordinates $F_1$ and $F_2$ of $F$ are multiplied by a normalizing factor $N$, and then quantized to the nearest integer to obtain a point $p$ with integral coordinates $(p_1, p_2) \in \mathbf{Z}^2$. A point $q = (q_1, q_2)$ on the screen is white (black) as it is (or is not) of form $(p_1, p_2)$, as described above.

The fractal set $\mathcal{F}(\mathcal{S})$, thus constructed, corresponds to a black-and-white image. In order to introduce grey levels into $\mathcal{F}(\mathcal{S})$, one would like to use the density of the points $\{F_{i_1}, \cdots, F_{i_k}\}$ as a measure of the brightness of the pixels. To do so, it is convenient to regard the procedure for the generation of a fractal set as a random process. In fact, let $p_1, \cdots, p_n$ be positive real numbers such that $\Sigma p_i = 1$, and $x \in \mathbf{R}^q$. Consider the random process $\mathcal{X}$ where $x \to S_i(x)$ with probability $p_i$. This process has a unique stationary distribution $\mu$. The stationarity condition is expressed by the equation

$$\mu = \Sigma\, p_i S_{i*}(\mu) \qquad (2)$$

where $S_{i*}(\mu)$ is the transform of the measure $\mu$ under the affine transformation $S_i$. Note that Eq. (2) is a more precise version of the fundamental self-similarity property expressed by Theorem 1. The measure $\mu$ is the mathematical representation of a grey-scale image on the screen. The support of the measure $\mu$ is the fractal set $\mathcal{F}(\mathcal{S})$, and is independent of the choice of positive numbers $p_i$. Furthermore, if $\mathcal{E}_k$ denotes the discrete probability measure naturally assigned to the set $\{F_{i_1}, \cdots, F_{i_l} \mid l \leq k \text{ and all } i_1, \cdots, i_l\}$, then

$$\mathcal{E}_k \to \mu \quad \text{weakly}$$

if all $p_i = 1/n$.

By introducing an alternative method for generating the fractal set $\mathcal{F}(\mathcal{S})$, one can take advantage of the probabilities $\{p_i\}$ in actual image generation. Consider a realization of the random process $\mathcal{X}$. This can be interpreted as follows: Using a random-number generator, one generates a sequence of integers $\{i_1, i_2, \cdots\}$, where $1 \leq i_k \leq n$ and the integer $j$ is chosen with probability $p_j$. A realization of the process $\mathcal{X}$ is then the sequence of points $S_{i_1}(z), S_{i_2}S_{i_1}(z), \cdots$. Let $\mathcal{M}_k$ be the discrete probability measure assigned to the set $\{S_{i_1}(z), S_{i_2}S_{i_1}(z), \cdots, S_{i_k} \cdots S_{i_2}S_{i_1}(z)\}$. Then, one can show by standard arguments that

**Theorem 2.** With probability 1, $\mathcal{M}_k \to \mu$ weakly.

After possibly multiplying by a factor $N$ and quantizing, one can regard a point $S_{i_l} \cdots S_{i_2}S_{i_1}(z)$ in a realization of the process $\mathcal{X}$ as the point $q_{i_l,\cdots,i_1} = q = (q_1, q_2) \in \mathbf{Z}^2$. One can make a histogram of the number of times the points of the integer lattice $\mathbf{Z}^2$ are generated in this fashion. Grey levels are accordingly assigned to the points so that the points generated more often have higher intensity (are whiter) than those generated fewer times. The additional input consisting of the positive numbers $\{p_1, \cdots, p_n\}$

allows one to have better control over the density of the points, i.e., the grey levels.

In order for any procedure to have practical applications, it must be stable relative to the variation of the parameters. In the case of fractals, one would like the fundamental property of self-similarity, as expressed by Theorem 1, to have the necessary stability. This means that if the transformations $S_i$ are such that $\bigcup_{i=1}^{n} S_i(K)$ is approximately identical with $K$, then the fractal set $\mathcal{F}(\mathcal{S})$ is also approximately identical with $K$. It is not difficult to establish, in a quantitative sense, the validity of this stability, which yields a slight generalization of Theorem 1. It is this mild generalization of Hutchinson's theorem that has been widely publicized by M. Barnsley as the *Collage Theorem* [2].

Just as in Example 1, where Theorem 1 was used to generate convex polygons as fractal sets, other natural-looking objects were generated by appealing to the approximate version and the experimental insight into the effects of the variation of the parameters on $\mu$. In actual applications, it is also convenient to decompose the matrix $A_i$ as a product

$$A_i = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} b\alpha & 0 \\ 0 & b'\alpha \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

where $|bb'| = 1$ and $0 < \alpha < 1$. The advantage of using this decomposition is that the effects of the parameters can be more readily understood. The matrix

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

is a rotation through angle $\theta$. The matrix

$$\begin{pmatrix} b\alpha & 0 \\ 0 & b'\alpha \end{pmatrix}$$

represents scaling by factors $b\alpha$ and $b'\alpha$, and

$$\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$$

can be regarded as a *twist*. The numbers $b$, $b'$, and $\alpha$ should be chosen such that $|b\alpha|$, $|b'\alpha| < 1$. By specializing the parameters $b$, $b'$, and $a$ to 1, 1, and 0, respectively, one can already generate a number of very complex and natural-looking images. The interested reader should view these points as hints or general guidelines for experimentation with fractal image generation.

Using the fractal method, a number of images were generated by one of the authors and others (see, e.g., [2] and [3]). Fractal methods have been used recently at JPL for computer simulation of certain images. Since these images were generated by storing only the parameters of a few affine transformations, one would like to believe that the fractal method can be used for efficient data, or more specifically, image compression. The systematization of the fractal method so that it becomes applicable to general image compression has been attempted by a number of researchers. One approach is to try to find a set $\mathcal{S} = \{S_1, \cdots, S_n\}$ of affine transformations and positive numbers $\{p_i\}$ such that the corresponding stationary distribution is a good approximation to a given image. While in theory this is possible, the number of the affine transformations may be so large that the result will have no practical value. Furthermore, since most objects do not exhibit, even remotely, the self-similarity property that is an essential feature of fractals, this direct approach is probably a futile one. By segmentation of an image, one will have better control over the choice of the affine transformations. However, the startling compression ratios of thousands to one will not be achievable. The most successful attempt of the application of fractals to image compression has been by Iterated Systems Inc. While their methodology is a well-guarded trade secret, the authors have tested the results of their fractal compression scheme against the JPEG baseline algorithm. This will be discussed in more detail in Section IV.

## III. JPEG Algorithms

With the advent of multi-media services offered by the 64-Kbit/sec Integrated Services Digital Networks (ISDN), there is a strong urge to define a standard for applications as diverse as photo-videotex, desktop publishing, graphic arts, color facsimile, photojournalism, medical systems, and many others. The JPEG was formed under the joint auspices of the International Standards Organization (ISO) and the Comité Consultatif International de Téléphone et Télécommunication (CCITT) at the end of 1986 for the purpose of developing an international standard for the compression and decompression of continuous-tone, still-frame, monochrome, and color images.

The JPEG-proposed algorithm has three major components. The first is a baseline system that provides a simple and efficient algorithm that is adequate for most image-coding applications. The second is a set of extended system features that allows the baseline system to satisfy a broader range of applications. Among these optional features are 12-bit/pixel input, progressive sequential and hierarchical build-up, and arithmetic coding. The third is an independent, differential, pulse-code modulation (DPCM) scheme for applications that require lossless compression. The following is a brief description of the JPEG baseline system only. The reader is referred to the JPEG proposal [4] for a complete description of the components and the algorithms.

The JPEG baseline system is a transform-based algorithm consisting of three stages. The first stage is a discrete cosine transform (DCT). The output of the DCT is then quantized and in the final stage the quantized output is encoded by variable length codes. The original image is partitioned into $8 \times 8$ pixel blocks and each block is independently transformed by the DCT. The transform coefficients are then quantized using a user-defined quantization template that is fixed for all blocks. Each component of the quantization template is an 8-bit integer and is passed to the receiver as part of the header information that is required for every image. Up to four different quantization templates can be specified; for example, different quantization templates may be used for the different components of a color image. The JPEG baseline system supplies two default quantization templates: one for the luminance component (the $Y$-component) and the other for the two chrominance components (the $I$ and $Q$ components). The top-left coefficient in the two-dimensional DCT block [i.e., the (0, 0) coefficient] is the DC coefficient and is proportional to the average brightness of the spatial block. The remaining coefficients are called the AC coefficients. After quantization, the DC coefficient is encoded with a lossless DPCM scheme using the quantized DC coefficient from the previous block as a one-dimensional predictor. For the baseline system, up to two separate Huffman tables for encoding the resulting differential signal can be specified in the header information. A default Huffman table for DC encoding is given in the JPEG proposal. The encoding of the quantized AC coefficients uses a combination of runlength and Huffman coding techniques. There are many zeros in the quantized AC coefficients, especially in the high frequencies. The AC coefficients that are close (respectively, far) in location to (0, 0) are the low (respectively, high) frequencies. Typically, high frequencies have low energies. The two-dimensional block of quantized coefficients is transformed into a one-dimensional vector using a zigzag reordering so that the coefficients are arranged in approximately decreasing order of their average energy. This creates a combination of nonzero values at the beginning of the vector and long runs of zeros thereafter. To encode the AC coefficients, each nonzero coefficient is first described by a composite 8-bit value, denoted by $I$, of the form (in binary notation)

$$I = NNNNSSSS$$

The four least significant bits, $SSSS$, define a category for the coefficient amplitude. The values in category $k$ are in the range $(2^{k-1}, 2^k - 1)$ or $(-2^k + 1, -2^{k-1})$, where $1 \leq k \leq 10$. Given a category, it is necessary to send an additional $k$ bits to completely specify the sign and magnitude of a coefficient within that category. The four most significant bits in the composite value, i.e., $NNNN$, give the position of the current coefficient relative to the previous nonzero coefficients. The runlengths specified by $NNNN$ can range from 0 to 15, and a separate symbol $I = 11110000 = 240$ is defined to represent a runlength of 16 zeros. In addition, a special symbol, $I = 0$, is used to code the end of a block (EOB), which signals that all the remaining coefficients in the block are zero. Therefore, the total symbol set contains 162 members (10 categories $\times$ 16 runlength values + 2 additional symbols). The output symbols for each block are then Huffman coded and are followed by the additional bits required to specify the sign and exact magnitude of the coefficient in each of the categories. Up to two separate Huffman tables for the AC coefficients can be specified in the baseline system. A default runlength/Huffman table for AC encoding is given in the JPEG proposal.

## IV. The Comparison

For the comparison of the fractal and the JPEG algorithms, a set of ten $320 \times 200$ 8-bit grey-scale images in Sun raster format was sent to Iterated Systems Inc. Some of these images are often used for image-compression experiments, and others are planetary images. The complete list is (1) an air scene, (2) baboons, (3) a couple in a room, (4) Lena (portrait of a young woman often used in image compression tests), (5) peppers, (6) light effects pattern, (7) the moon, (8) Miranda, and (9) and (10) two images of the planet Saturn. Each of the images was compressed to eleven files of sizes ranging from 3 to 13.5 Kbytes. The reconstruction was done by the decompression software *P.OEM*™ *Developers' Kit—Grayscale Still* developed by Iterated Systems Inc. for this purpose and the format conversion developed by one of the authors. The reconstructed images were then compared to the original to obtain the mean-square-error (MSE) and the peak-SNR values versus the number of bits per pixel (i.e., compression ratio). Peak SNR is defined as $10\log_{10}(255)^2/MSE$, and is measured in dB's. Similar curves were obtained for the JPEG baseline algorithm as implemented at JPL [5]. The results are given in Figs. 1 and 2 for images (5) and (8). It is clear from the figures that the JPEG algorithm is superior to the fractal algorithm by as much as 6 dB at low compression ratios and 2 dB at high compression ratios. The curves presented in this article contain the range of compression ratios obtained by Iterated Systems Inc. for the ten images referred to earlier. It should be pointed out that no conclusion regarding the effectiveness of the fractal method at higher compression ratios is warranted at this time. Qualitatively, at high compression ratios of about 16:1 to 20:1, the fractal scheme exhibits *solar flare*, while the JPEG baseline algorithm suffers from the *tiling effect.* No post-processing was done with the application of the JPEG algorithm. The authors are unaware of any post-processing in the fractal algorithm. The tiling effect can be somewhat alleviated by the application of certain filters which involve weighted averages of nearby neighbors. The authors know of no method for dealing with the solar flare effect.

## V. Conclusion

A comparative study of the JPEG baseline algorithm and the fractal method was conducted by the authors. In terms of the mean square error and peak SNR, the JPEG algorithm was superior in every case.

## References

[1] J. Hutchinson, "Fractals and Self-Similarity," *Indiana University Journal of Mathematics*, vol. 30, no. 5, pp. 713–747, 1981.

[2] M. Barnsley, *Fractals Everywhere*, New York: Academic Press, 1988.

[3] P. Diaconis and M. Shahshahani, "Products of Random Matrices and Computer Image Generation," *Contemporary Mathematics*, vol. 50, pp. 163–173, American Mathematical Society, Providence, Rhode Island, 1986.

[4] W. B. Pennebaker to ISO Group X3L2.8, *JPEG Draft Technical Specification Revision 8*, IBM, Yorktown Heights, New York, August 17, 1990.

[5] F. Pollara and S. Arnold, "Emerging Standards for Still Image Compression: A Software Implementation and Simulation Study," *TDA Progress Report 42-104*, vol. October–December 1990, Jet Propulsion Laboratory, Pasadena, California, pp. 98–101, February 15, 1991.

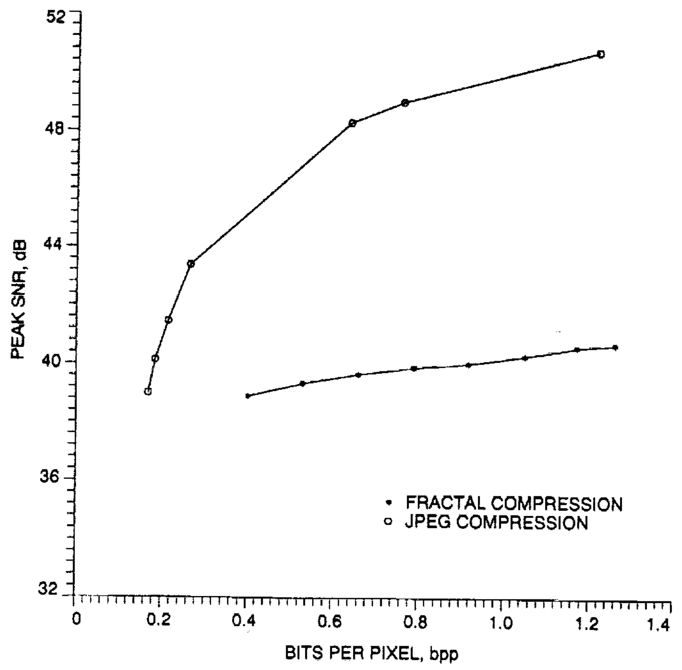

**Fig. 1. Fractal performance on peppers.**

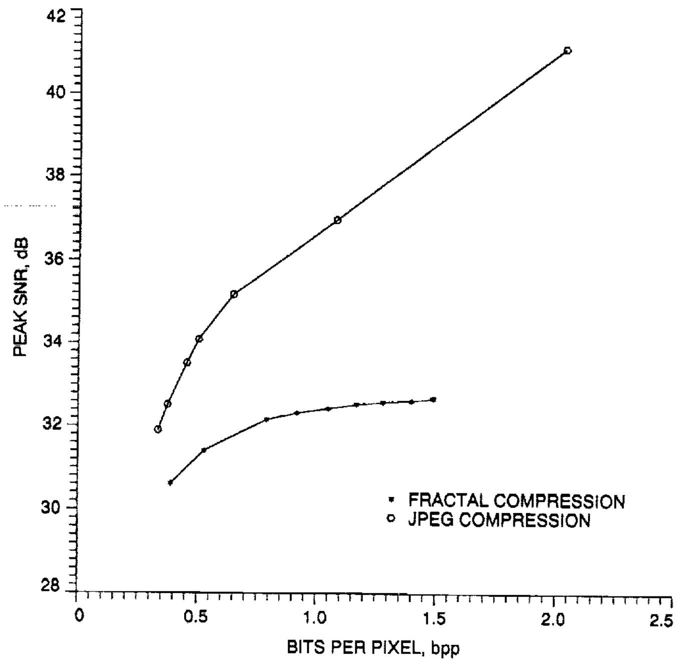

**Fig. 2. Fractal performance on Miranda.**