Fractal Image Compression and Recurrent Iterated Function Systems

John C. Hart
Washington State University

Fractal geometry provides a basis for modeling the infinite detail found in nature. Fractal methods are quite popular in computer graphics for modeling natural phenomena such as mountains, clouds, and many kinds of plants. Linear fractal models such as the iterated function system (IFS), recurrent iterated function system (RIFS), and Lindenmeyer system (L-system) concisely describe complex objects using self-reference. These models hold much promise in computer graphics as geometric representations of detail.

Fractal techniques have recently found application in the field of image compression. The use of fractals for compression has grown into a well-established area of signal processing, but this use sacrifices its "fractal" origins in the search for optimal coding gain.

The sidebar (p. 26) identifies the differences between the fields of fractal image compression and fractal geometry. This article rebuilds the relationship between them to better facilitate the sharing of new results.

Many geometric representations exist for smooth shapes, and each has certain benefits and drawbacks. Computer-aided geometric design has produced many algorithms to convert a given curve or surface description into the most appropriate geometric representation for a given task. Likewise, there are several models for linear fractal shapes and several methods for converting between the representations, such as from L-system to RIFS,[1] from RIFS to L-system,[2] and from L-system and RIFS to constructive solid geometry.[3]

The representation used by fractal image compression has been called *partitioned IFS*[4] or, synonymously, *local IFS*.[5] This article describes a method for converting fractal image compression's partitioned/local IFS to fractal geometry's RIFS. This conversion algorithm allows fractal image compression to represent any input shape as a linear fractal and permits algorithms developed for linear fractals to be applied to a wider variety of shapes.

Recurrent modeling theory

Recurrent modeling is the process of partitioning an object into components and representing each component as a collection of shrunken copies of possibly itself and possibly other components.

Iterated function systems

An iterated function system consists of a set of maps $\{w_i\}_{i=1}^N$ from $\mathbf{R}^n$ into itself. If the maps of an IFS are contractive, each IFS includes a single, compact, nonempty set $A \subset \mathbf{R}^n$, called its *attractor*,[6] defined as the union of images of itself under the IFS maps:

$$A = \bigcup_{i=1}^{N} w_i(A) \tag{1}$$

The Hutchinson operator $\boldsymbol{w}$ is a convenient shorthand notation

$$\boldsymbol{w}(\cdot) = \bigcup_{i=1}^{N} w_i(\cdot) \tag{2}$$

that allows us to simplify the definition of an IFS attractor as

$$A = \boldsymbol{w}(A) \tag{3}$$

Furthermore, the attractor A gets its name from the property that, given any initial nonempty bounded set $B \subset \mathbf{R}^n$, we have

$$A = \lim_{i\to\infty} \boldsymbol{w}^{\circ i}(B) \tag{4}$$

where $\boldsymbol{w}^{\circ i}$ denotes the i-fold composition of $\boldsymbol{w}$ (that is, $\boldsymbol{w}^{\circ i} = \boldsymbol{w} \circ \boldsymbol{w}^{\circ i-1}$).

When every domain element can be expressed as the union of range elements, fractal image compression produces a structure equivalent to an RIFS, enabling it to automatically model arbitrary shapes.

Recurrent iterated function systems

Likewise, a recurrent iterated function system consists of a set of affine transformations $\{w_i\}_{i=1}^N$ and a directed graph G. Each edge $\langle i,j\rangle \in G$ indicates that the composition $w_j \circ w_i$ is allowed. If a RIFS consists of contractive maps, there exists a single compact nonempty attractor[7] $A \subset \mathbf{R}^n$, defined as a collection of possibly over-

Fractal image compression versus fractal geometry

While the term "fractal" appears in both their names, fractal image compression differs greatly from typical applications of fractal geometry. These differences were fleshed out in a recent NATO Advanced Study Institute on Fractal Image Compression and Encoding (July 1995, Trondheim, Norway):

1. Fractal image compression performs well on straight edges and flat surfaces, whereas fractal geometry was designed to represent jagged edges and rough surfaces.
2. Fractal image compression blurs texture in images, whereas fractal geometry enhances texture.
3. Fractal image compression operates on a discrete space, whereas the infinite detail of fractal geometry requires a continuous space.
4. Fractal image compression dices the input into many tiny pieces (and sometimes requires coordinate-dependent functions) to force self-similarity, whereas fractal geometry capitalizes on a shape's natural self-similarities.
5. Fractal image compression works well even for the worst range-domain matches and works better when similarity is identified with respect to another independent image, whereas fractal geometry is based on exact or statistically significant self-reference.

lapping partitions $A_j \subset \mathbf{R}^n$:

$$A = \bigcup_{j=1}^{N} A_j \tag{5}$$

which are each the union of an image of other partitions (including possibly itself):

$$A_j = \bigcup_{\langle i,j\rangle \in G} w_j(A_i) \tag{6}$$

We can keep the components of this partitioning separate by denoting the attractor $A = A_1 \cup A_2 \cup \cdots \cup A_N$ as a vector of sets $\boldsymbol{A} = (A_1, A_2, \ldots, A_N) \in (\mathbf{R}^n)^N$.

The domain of the RIFS maps extends to set vectors $w_j\colon (\mathbf{R}^n)^N \to \mathbf{R}^n$, which are defined

$$w_j(\boldsymbol{A}) = \bigcup_{\langle i,j\rangle \in E} w_j(A_i) \tag{7}$$

Using these maps, the recurrent Hutchinson operator $\boldsymbol{w}\colon (\mathbf{R}^n)^N \to (\mathbf{R}^n)^N$ is defined on set vectors as

$$\boldsymbol{w}(\boldsymbol{A}) = (w_1(\boldsymbol{A}), w_2(\boldsymbol{A}), \ldots, w_N(\boldsymbol{A})) \tag{8}$$

The attractor, consisting of the components defined by Equation 6, is now more concisely defined in set-vector notation as

$$\boldsymbol{A} = \boldsymbol{w}(\boldsymbol{A}) \tag{9}$$

to better resemble Equation 3. Moreover, given any set vector $\boldsymbol{B} = (B, B, \ldots, B) \subset (\mathbf{R}^n)^N$ consisting of bounded nonempty sets $B \subset \mathbf{R}^n$, then

$$\boldsymbol{A} = \lim_{i\to\infty} \boldsymbol{w}^{\circ i}(\boldsymbol{B}) \tag{10}$$

To recap, given any initial nonempty bounded set $B \subset \mathbf{R}^n$, iterating $\boldsymbol{w}$ on the set vector $\{B, B, \ldots, B\} \in (\mathbf{R}^n)^N$ creates a sequence converging to the set vector $\boldsymbol{A}$.

If each partition is the union of images of every partition including itself, then the graph G is complete and the RIFS is simply an IFS. Hence, every IFS is an RIFS. Thus the remainder of this section focuses on properties of the RIFS representation.

Open set property

The degree of overlap of the partitioning is dictated by the open set property. An RIFS $\langle \{w_i\}_{i=0}^{N}, G\rangle$ satisfies the *open set property*[6] if and only if there exists a set-vector $\boldsymbol{U} = (U_1, U_2, \ldots, U_N)$ of open sets $U_i \subset \mathbf{R}^n$ such that its image $\boldsymbol{V} = \boldsymbol{w}(\boldsymbol{U})$ has the property $V_i \subset U_i, \forall i$.

Many algorithms use spatial subdivision to process the infinite detail of an RIFS attractor. When an RIFS satisfies the open set property, and the boundaries of the subdivision agree with the boundaries of the open sets, then spatial subdivision algorithms perform optimally and quickly eliminate from consideration all but the desired portion of the attractor. When the components (A_i) of an RIFS significantly overlap, such algorithms exhaustively search the entire attractor.

Digraph topology

A digraph is *strongly connected* if and only if every pair of vertices is connected by a directed path of edges. A digraph is *weakly connected* if and only if every pair of vertices is connected by a path of edges, regardless of edge direction. We follow the convention that strongly connected and weakly connected are mutually exclusive.

Some RIFS algorithms, such as the chaos game in the next section, require a strongly connected digraph, while others do not even require the digraph to be connected.

The chaos game

The chaos game approximates the attractor of an IFS $\{w_i\}_{i=1}^{N}$ with a point cloud. It starts with any initial point $x^{(0)}$ and generates a sequence of points as

$$x^{(k+1)} = w_j(x^{(k)}) \tag{11}$$

where j is an integer from one to N randomly chosen for each new point in the sequence. This sequence of points is dense in the attractor,[6] though choosing each map with a probability proportionate to its effective change in area (for R^2) provides more uniform coverage of the attractor by the chaos game.[8] Figure 1 shows the simple chaos game algorithm. More advanced versions (not needed in this discussion) use probabilities to balance the distribution of points.

Equation 11 may also be used to render the attractor of a strongly connected RIFS, though instead of choosing the index i at random from one to N, the index j is chosen such that the edge $\langle i, j\rangle$ is in the control digraph G, where i is the index of the previously applied map.[7]

The simple chaos game works on a strongly connected RIFS because there is a directed path of edges from every vertex to every other vertex. This is not so for an RIFS with a weakly connected or disconnected digraph, where a point may be trapped in an isolated portion of the digraph. In this case, the single iterated point is replaced by N points, one in each $\mathbf{R}^n$ of $(\mathbf{R}^n)^N$. Hence we have a sequence of point-vectors $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \dots, x_N^{(k)})$ whose components are defined

$$x_j^{(k+1)} = w_j(x_i^{(k)}) \tag{12}$$

where i is randomly chosen such that edge $\langle i, j\rangle$ is in the control digraph.[9] Figure 2 shows the RIFS chaos game algorithm.

```
Let $\{w_j\}_{j=1}^N$ be an IFS.
Initialize $x = (0,0)$.
Do forever ...
    Pick a number $i$ randomly from 1 to $N$.
    Let $x = w_i(x)$.
    Plot $x$.
End do.
```

1 **The simple chaos game algorithm.**

```
Let $<\{w\}_{i=1}^N, G>$ be an RIFS.
For $i$ = 1 to $N$ ...
    Initialize $x_i = (0, 0)$.
End for.
Do forever ...
    For $j$ = 1 to $N$ ...
        Pick a number $i$ such that $\langle i, j\rangle \in G$
        Let $y_j = w_j(x_i)$.
        Plot $y_j$.
    End for.
    For $i$ = 1 to $N$ ...
        Let $x_i = y_i$.
    End for.
End do.
```

2 **The RIFS chaos game algorithm.**

Fractal image compression

Fractal image compression is only one of many attempts at solving the inverse problem (see the sidebar, next page). Of all the other inverse-problem solutions, none have attained the robustness and generality of fractal image compression. Fractal image compression rivals other, more thoroughly researched compression algorithms and is the best choice for certain kinds of images.

However, fractal image compression ignores the natural morphological self-similarity of a shape; instead, it finds coincidental self-similarities between arbitrarily chosen square image segments. Although fractal image compression has been exceptionally successful as a solution to the inverse problem, its resulting fractal model is nearly meaningless. Fractal image compression is only an incidental step toward the development of general techniques for the fractal-based analysis and representation of arbitrary objects.

The following summary of fractal image compression reviews only the fundamental points of the method necessary for comparison with recurrent iterated function systems. Many enhancements and variations exist,[10] but they fall beyond the scope of this discussion.

Whereas RIFSs operate on a continuous metric space, fractal image compression operates on the discrete metric space of images. While some researchers treat images mathematically as measures and others treat them as functions, the following discussion treats an image as a *height field*, a 3D object defined by a function evaluated over a 2D plane, that is, $z = f(x, y)$.

The technique partitions an image (height field) into both a fine collection of nonoverlapping *range blocks* $\{R_j\}_{j=1}^N$ and a coarser, possibly overlapping collection of *domain blocks* $\{D_i\}_{i=1}^M$. For example, a 256×256 image would partition into $N = 64^2 = 4{,}096$ nonoverlapping 4×4-pixel range blocks and produce a collection of $M = 63^2 = 3{,}969$ overlapping 8×8 domain blocks spaced at four-pixel intervals.

Let $\{w_\alpha\}$ be a collection of affine transformations, where $\alpha = (a, b, c, d, e, f, g, h)$ is a vector of its parameters. Each of these 3D affine maps is composed of a 2D geometric part that reduces the size of a domain block to the size of a range block, accompanied by rotations and reflections, and a 1D gray-level (height) part that reduces the block's contrast and adjusts its brightness. Specifically, $(x, y, z) \xmapsto{w_\alpha} (x', y', z')$ can be expressed using the standard homogeneous 4×4 transformation matrix notation of computer graphics as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & 0 & e \\ c & d & 0 & f \\ 0 & 0 & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/s & 0 & 0 & 0 \\ 0 & 1/s & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{13}$$

where $1/s$ denotes the factor that scales a domain block to the size of a range block (domain blocks are s times larger than range blocks), $a, b, c, d \in \{-1, 0, 1\}$ form an isomorphism, e, f translate the scaled domain block to the range block, $g < 1$ reduces contrast, and h adjusts brightness.

The encoding process is now reduced to finding, for each range block R_j, the transformed domain block $w_{\alpha^*}(D_{i^*})$ that best matches it, such that

$$\left\| R_j - w_{\alpha^*}(D_{i^*}) \right\|^2 = \min_{\forall \alpha, i} \left\| R_j - w_\alpha(D_i) \right\|^2 \tag{14}$$

The location i^* of the best domain block and the parameters α^* of the best transformation are then stored in place of the range block R_j and will be referenced during decompression as $i(j)$ and $\alpha(j)$.

Reduction of block coding to RIFS

The range blocks are decoded by iterating the blockwise transformations on a domain partition of any initial image. Large features in an early image of the decoding sequence become fine features in later images. The decompression iteration constructs each range block R_j from the image, under $w_{\alpha(j)}$ of a domain block $D_{i(j)}$ from the previous image. The "fractal code" of the image is contained in the functions $\alpha(j)$ and $i(j)$, which return the corresponding transformation parameters and domain block location, respectively, for each range block R_j.

The Inverse Problem

Shortly after the introduction of iterated function systems, an *inverse problem* arose: given an object, find an iterated function system that represents that object within a given degree of accuracy. General inverse problem techniques differ from fractal image compression in that they are more concerned with model building and shape representation.

The collage theorem provided the first stepping stone toward solving the inverse problem. It states that a lazy tiling of an object out of smaller self-replicas yields a significantly more accurate IFS representation.

M.F. Barnsley et al., "Solution of an Inverse Problem for Fractals and Other Sets," *Proc. Nat'l Academy of Science*, Vol. 83, Apr. 1986, pp. 1,975-1,977.

Although the collage theorem relaxed the accuracy of self-similar tiling needed for reasonable fractal modeling, the harder problem of forming an object out of smaller self-replicas remained unsolved.

The automatic fractal modeling of shapes is based on heuristic minimization techniques. They have yielded promising results, but are not yet practical. The following is a selection of the many attempts at automatic solutions to the inverse problem.

Analytic solutions for very simple, highly constrained attractors were found using the method of moments:

M.F. Barnsley and S.G. Demko, "Iterated Function Schemes and the Global Construction of Fractals," *Proc. Royal Society A*, Vol. 399, 1985, pp. 243-275.

C.R. Handy and G. Mantica, "Inverse Problems in Fractal Construction: Moment Method Solutions," *Physica D*, Vol. 43, 1990, pp. 17-36.

A gradient descent search minimized the difference in moments between the IFS attractor and a given shape:

E.R. Vrscay and C.J. Roehrig, "Iterated Function Systems and the Inverse Problem of Fractal Construction using Moments," *Computers and Mathematics*, E. Kaltofen and S.M. Watt, eds., Springer-Verlag, New York, 1989, pp. 250-259.

Newton's method found the parameters that minimized the distance between a set and an IFS attractor:

W.D. Withers, "Newton's Method for Fractal Approximation," *Constructive Approximation*, Vol. 5, 1989, pp. 151-170.

Genetic algorithms overcome the problem of l ocal minima (though at the expense of numerous "generations" and slow convergence):

E.R. Vrscay, "Moment and Collage Methods for the Inverse Problem of Fractal Construction with Iterated Function Systems," in *Fractals and the Fundamental and Applied Sciences*, H-O. Peitgen, J. Henriques, and L. Penada, eds., North-Holland, New York, 1991, pp. 443-461.

A wavelet technique found the parameters of an IFS containing only scales and translations:

R. Rinaldo and A. Zakhor, "Inverse and Approximation Problem for Two-Dimensional Fractal Sets," *IEEE Trans. on Image Processing*, Vol. 3, No. 6, Nov. 1994, pp. 802-820.

A recent NATO Advanced Study Institute on Fractal Image Encoding and Analysis explored numerous techniques for using fractals to understand and compress images. Many of these papers are expected to appear in an upcoming special issue of the journal *Fractals*. Of these papers, only the following three presented morphological solutions to the inverse problem.

The computationally-intensive Kantorovich (Hutchinson) metric appeared to behave better for gradient-descent minimization:

N. Wadstromer, "An Approach to the Inverse IFS Problem using the Kantorovich Metric."

The following presentation described a technique for finding the best similarity transformations of an IFS given its fixed points:

E. Hocevar and W.G. Kropatsch, "Inventing the Formula of the Trees: A Solution of the Inverse Problem of the Representation of Self-Similar Images."

The following presentation adapted a model-based computer vision technique (used to identify known models in a scene) to detect self-recurrence of feature points in an image:

J.C. Hart, W.O. Cochran, and P.J. Flynn, "Similarity Hashing: A Model-Based Vision Solution to the Inverse Problem of Recurrent Iterated Function Systems."

The block-coding structure of fractal image compression is a representation of an RIFS. This is most easily seen when the domain blocks do not overlap and their boundaries align with the boundaries of range blocks. However, this result generalizes to include cases of overlapping and nonaligned domain blocks.

Theorem: The block-coding structure for fractal image compression, with domain blocks formed by the union of range blocks, reduces to an RIFS.

Proof: Fractal image compression produces a partitioned or local IFS attractor where each range block is the image of some domain block

$$R_j = w_{\alpha(j)}(D_{i(j)}) \tag{15}$$

The function $\alpha(j)$ returns the parameters of the transformation, and the function $i(j)$ returns the index of the domain block that the transformation maps to range block R_j. For simplicity, we abbreviate $w_{\alpha(j)}$ as w_j.

Every domain block D is the union of K range blocks, where $K = M^2/N^2$ for images

$$D_i = \bigcup_{k=1}^{K} R_{j(i,k)} \tag{16}$$

The function $j(i, k)$ returns the indices of the K range block components of the domain block i.

Substituting Equation 16 into Equation 15 shows that each range block is the image of the union of range blocks

$$R_j = w_j\left(\bigcup_{k=1}^{K}(R_{j(i(j),k)})\right) \tag{17}$$

Since the image of a union is equivalent to the union of images, Equation 17 becomes

$$R_j = \bigcup_{k=1}^{K} w_j(R_{j(i(j),k)}) \tag{18}$$

as illustrated in Figure 3.

This matches the form of an RIFS attractor (Equation 6) partitioned into range blocks $\boldsymbol{A} = (R_1, R_2, \ldots, R_N)$ with a digraph where each vertex j has an in-count of K edges $\langle j(i(j), k), j\rangle$ for $1 \le k \le K$. ■

Remarks:

1. Substituting Equation 15 into 16 shows that each domain block is the union of the images of domain blocks

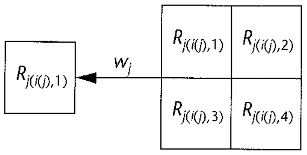
$$D_i = \bigcup_{k=1}^{m} w_{j(i,k)}(D_{i(j(i,k))})$$

as illustrated in Figure 4. This almost matches the form of the definition of the RIFS attractor in Equation 6, except that we have the union of images under different maps.
2. The domain blocks may overlap if their boundary falls on the range-block boundaries. The domain window may slide across the image in steps equal to the range-block edge length.
3. The RIFS has the open set property. Hence, the resulting RIFS is structured to ensure efficient operation of linear fractal algorithms based on area subdivision.
4. Yuval Fisher concurrently derived a result identical to the theorem to analyze fractal image compression using the tools developed for the RIFS.[4] In this context, the theorem broadens the scope of RIFS research using tools developed for fractal image compression.
5. The RIFS form permits image decoding through point sampling and the chaos game (as explained in greater detail in the next section).

Corollary: The block-coding structure for fractal image compression with sliding-window domain blocks can be represented by a 3D RIFS.

Proof: Let d be a positive real number such that the minimum amount domain blocks can be offset is od, and the range-block edge length is ld, where o and l are positive integers. Subdivide the range block partition $\{R_j\}_{j=1}^{N}$ into a finer partitioning $\{\hat R_{\hat j}\}_{\hat j=1}^{\hat N}$ with edge length d such that each original range block is the union of l finer range blocks. Define $\hat D_{\hat i(\hat j)}$ as the set of points in $D_{i(j)}$ such that

$$\hat R_{\hat j} = w_{j(\hat j)}(\hat D_{\hat i(\hat j)}) \tag{20}$$



3 Each range block is the union of images of range blocks. The union of four range blocks on the right forms the domain block $D_{i(j)}$.

where $j(\hat j)$ returns the index j such that $R_{\hat j} \subset R_j$. If $\hat R_{\hat j} \subset R_j$ and $w_j(D_{i(j)}) = R_j$ then $\hat D_{\hat i(\hat j)} \subset D_{i(j)}$. Equation 20 resembles Equation 15.

Since the finer domain-block edge length is an integer multiple of the finer range-block edge length d, each finer domain block is the union of K finer range blocks

$$\hat D_{\hat i} = \bigcup_{\hat k=1}^{\hat K} \hat R_{\hat j(\hat i,\hat k)} \tag{21}$$

which resembles Equation 16.

Substituting Equation 21 into 20 yields

$$\hat R_{\hat j} = \bigcup_{\hat k=1}^{\hat K} w_{j(\hat j)}(\hat R_{\hat j(\hat i(\hat j),\hat k)}) \tag{22}$$

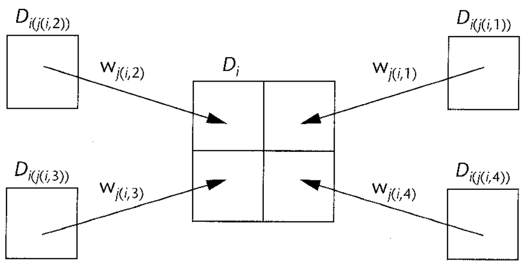
and, as before, we have an RIFS, with attractor given by the set vector $\boldsymbol{A} = (\hat R_1, \hat R_2, \ldots, \hat R_{\hat N})$. ■

Consider any assortment of range blocks and domain blocks such that each range block is the affine image of a domain block. Unlike the previous cases, however, the domain blocks are not unions of range blocks.

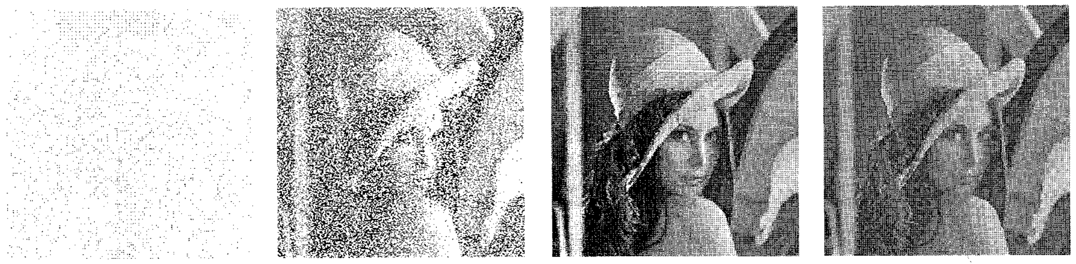
Conjecture: When the affine maps have rational coefficients, there exists a subpartitioning of range blocks such that each range sub-block is the image of the union of other range sub-blocks under the appropriate affine transformation. Hence, fractal image compression using arbitrary range and domain block shapes yields a data structure equivalent to a RIFS.

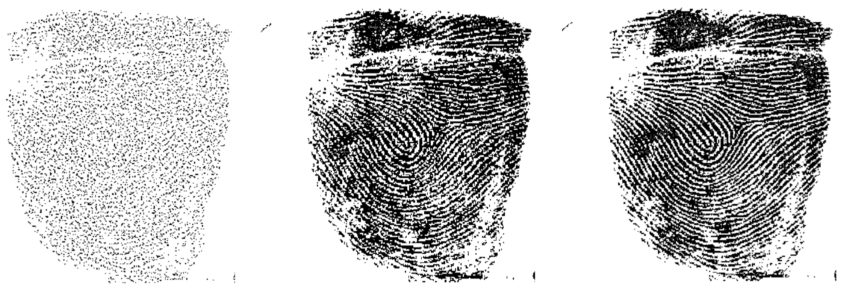
A proof of this conjecture would be lengthy and beyond the scope of this discussion. Such a proof would also find little practical application, since even simple conditions would require the subdivision of range blocks into very small elements. The resulting RIFS would contain so many maps as to be both useless as a fractal model and worthless as a compressed encoding.

Nonetheless, the following sketch would likely yield a proof of the conjecture. Define a sequence of range partitions $(\boldsymbol{R}^{(0)}, \boldsymbol{R}^{(1)}, \ldots)$ where $\boldsymbol{R}^{(0)}$ is the initial range partitioning resulting from the fractal image compression. For



4 Each domain block is the union of images of domain blocks under different maps.


5 Chaos game decompression of an RIFS representation of a compressed image. From left to right: one, 16, 256, and 512 iterations.


6 First few iterations (left) and final attractor (center) of a fingerprint encoded into 1,030 RIFS maps from the original (right).

every range partitioning $\boldsymbol{R}^{(l)}$ in the sequence, its accompanying collection of domain blocks $\boldsymbol{D}^{(l)}$ consists of sets $D = w_i^{-1}(R)$ for each range block $R \in \boldsymbol{R}^{(l)}$ and appropriate w_i. Given both $\boldsymbol{R}^{(l)}$ and $\boldsymbol{D}^{(l)}$, the next range partitioning $\boldsymbol{R}^{(l+1)}$ consists of elements that are each the nonempty intersection $R \cap D$ of a range block from $\boldsymbol{R}^{(l)}$ and a domain block from $\boldsymbol{D}^{(l)}$. The sequence converges when and if $\boldsymbol{R}^{(\ell)} = \boldsymbol{R}^{(\ell+1)}$ at some level ℓ. Convergence is expected after d iterations, when the affine maps have rational coefficients (that is, rational rotation angles) and d is the least common denominator of these coefficients.

Results

The conversion algorithm was applied to both grey-level images and to bitmaps. For a given shape, the algorithm can produce an edge image and an "escape time" coloring of the complement.

The RIFS representation of an image

One method for decompressing a fractal coded image consists of applying the domain-range transformations repeatedly to an arbitrary initial image, thus creating a sequence of images that converges to a limit that approximates the original image. The RIFS representation similarly allows the chaos game to decode the image by plotting the orbits of N points, as demonstrated in Figure 5.

While the chaos game reconstructs a recognizable version of the image, it does not converge. After 256 iterations, the (mean-square) signal-to-noise ratio with respect to the original image is a miserable 21 dB and never improves. Even though the image fidelity remains constant, the image itself changes. The signal-to-noise ratio between images decoded after 256 and 512 chaos game iterations is 21 dB. After only seven iterations, a typical fractal image decompression algorithm[11] converges and yields a signal-to-noise ratio of 1.6 dB with respect to the original image.

The disappointing performance of the RIFS chaos game is due to inherent differences between the RIFS's continuous space of reals $\mathbf{R}^3$ and fractal image compression's discrete space of integers $\mathbf{Z}^3$. A contraction on a discrete space must eliminate some information that cannot later be reproduced. Hence discrete space lacks the necessary information to properly decode the image from the RIFS representation.

In $\mathbf{R}^3$, the corresponding value of a pixel may be considered either an average value (z component) over the pixel's area (x, y components), or a single sample (z) from a specific point (x,y) in the pixel's area. For reconstruction of a fractal compressed image, these values typically form a fractal function over the pixel's area. Sampling a single point in this area returns a chaotic result, and integration of a fractal function is problematic. This sampling problem prevents the chaos game on $\mathbf{R}^3$ from accurately decoding the image.

The RIFS representation of a bitmap

Applying fractal image compression to bilevel images (bitmaps) provides a more successful application of the chaos game in the decoding process. The pixels in such images are either on or off, and the image can be considered a discrete approximation of a continuous set. In this form, only the geometric part of the transformation is used; the grey-level transformation is always the identity. Range blocks containing only "off" pixels are not encoded. The resulting RIFS form of the fractal encoding yields the "on" pixels of the image as its attractor.

For example, the process encoded a $256 \times 256 \times 1$ fingerprint, shown in Figure 6, at a range block resolution of 4×4 pixels, using only 1,030 of the possible 4,096 RIFS maps. Although the encoded fingerprint appears much noisier than the original, with a signal-to-noise ratio of only 5.9 dB, its features remain and would nevertheless serve identification purposes.

The process encoded an artist's etching of a dragon[12] in Figure 7, with 1,048 RIFS maps. Although most of the dragon's features, such as its bumps and scales, are reproduced, encoding has approximated the finely hashed shading patterns with fractal texture equivalents. The signal-to-noise ratio of this representation is 11.7 dB.

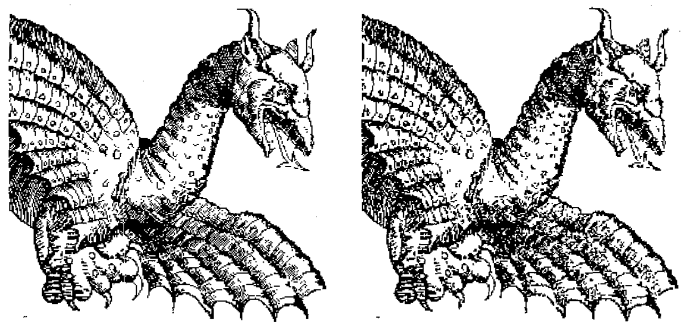
Figure 8 shows the encoding of a snowflake[13] with 827 RIFS maps. Though not as severely as in the fingerprint, the encoding has introduced some noise, with a signal-to-noise ratio of 13.4 dB, but the image certainly remains adequate for identification purposes. The fine block partitioning overlooks the morphological hexagonal and scalar self-similarity of the snowflake.

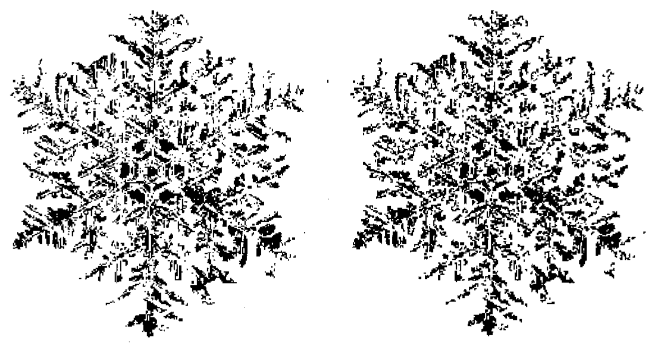
Each of the above examples was encoded from a 256 × 256 resolution original, which requires 8,192 bytes to store. The compression performance is listed in Table 1.

Each RIFS map consisted of one of eight isometries (3 bits), a modulo-four translation of ±256 (7 bits), and an index to the corner of the domain block, from which the appropriate RIFS-controlling digraph can be derived (12 bits), for a total encoding of 22 bits per map. Further encoding of these parameters through Huffman tables or arithmetic coding would yield higher compression ratios.

If the entire image contained information, then all 4,096 4 × 4 range blocks would require coding, producing 11,264 bytes (38 percent larger than the unencoded size). This increase happens when standard fractal image compression techniques are applied to bitmaps. The break-even point for RIFS encoding occurs at 2,979 maps, which occurs when detail covers 73 percent of the image.

In cases where much of the image is a solid color, run-length encoding often compresses adequately. However, for these examples, any savings due to the solid background is more than lost by the cost of the image detail. In each case, the run-length encoding (RLE) is larger than the unencoded image.


7 Original dragon (left) and approximation (right) by an RIFS of 1,048 maps.



8 Bentley's photograph of a real snowflake[13] (left) and approximation (right) represented by an RIFS of 827 maps.

Table 1. Performance of RIFS encoding of bitmaps. Sizes are in bytes and compression is relative to the original file size of 8,192 bytes.

Image	Maps	Size	Compression	SNR (ms)	RLE Size
Fingerprint	1,030	2,833	35%	5.9 dB	10,455
Dragon	1,048	2,882	35%	11.7 dB	8,749
Snowflake	827	2,275	28%	13.4 dB	13,234

Edge detection of a bitmap

Bitmap range blocks fall into three categories: all white, all black, and mixed. The RIFS representation ignores the all-white range blocks; each all-black range block is constructed of smaller copies of four all-black range blocks. Removing the all-black range blocks from the RIFS representation yields an RIFS whose attractor approximates the boundary of the bitmap, as demonstrated in Figure 9 (next page).

In two dimensions, the RIFS encoding perfoms as a fractal edge detector. For 3D volume information, the edge detector yields a fractal encoding of an isosurface.

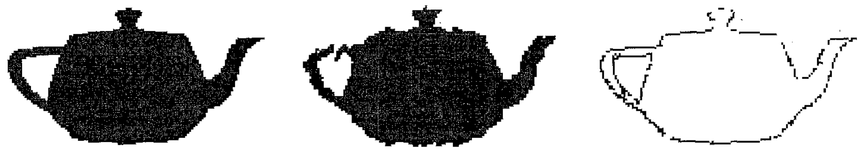
The escape time classification of a bitmap

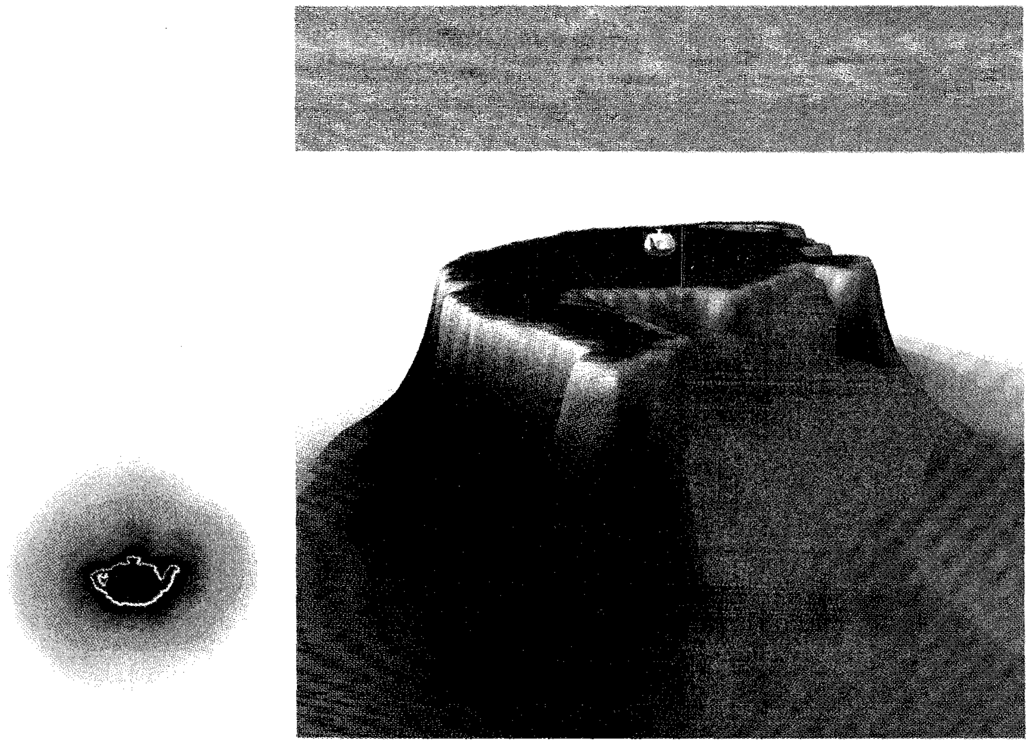
Escape time is a visualization tool used to understand the dynamics of fractal representations. Escape time indicates the number of RIFS transformation applications required to determine that a point does not belong to the fractal shape. The escape time is computed for each point in $\mathbf{R}^2$ by finding the longest possible sequence of inverted RIFS transformations such that the orbit of the initial point always remains within a given region surrounding the attractor.[14]

The theorem allows escape time to classify the complement of any arbitrary shape by first representing it as an RIFS, then computing the escape time of the RIFS. Figure 10 demonstrates two rendering styles of the escape-time classification for the teapot bitmap, computed using the escape buffer algorithm[15] on the 91-map RIFS representation.

Conclusion

Under certain conditions, fractal image compression yields a representation equivalent to an RIFS. While the RIFS representation does not directly improve perfor-


9 Original bitmap of the teapot (left), 91-map (8 × 8) RIFS solid representation (center), and 103-map (4 × 4) RIFS boundary representation (right).


10 Escape time visualization of the RIFS representation of the teapot bitmap, rendered using a color map (left) and as a height field (right).

mance on gray-level images, it does represent bitmaps more concisely than standard fractal image compression techniques.

The equivalence of fractal image compression to RIFSs is a step toward mapping the wealth of recent results in fractal image compression[10] to the broader task of solving the inverse problem of recurrent iterated function systems.

Block coding imposes an artificial grid onto an image. The fractal representation resulting from matching range blocks to domain blocks does not in general provide any clue to the actual fractal structure of the shape. For example, Sierpinski's gasket or von Koch's snowflake would require hundreds of RIFS maps to encode by this process even though they have easily identifiable self-similarities.

More recent methods that adaptively resize the blocks based on the underlying structure hide the artifacts of the compression in the high frequency components of the image,[4] but still overlook any self-similarity involving rotations or overlapping construction. Instead, the blocking structure itself should be abandoned in exchange for the invariant morphological properties of pattern recognition and computer vision, which hold great promise for producing RIFS models that not only compress data, but reveal its structural self-similarity. ■

Acknowledgments

This research is part of the Recurrent Modeling Project, which is supported in part by a gift from Intel Corp. The author is also supported in part by the NSF Research Initiation Award #CCR-9309210. The research was performed using the facilities of the Imaging Research Laboratory, which is supported in part under grants #CDA-9121675 and #CDA-9422044.

Conversations at the 1995 NATO Advanced Study Institute on Fractal Image Encoding and Analysis were very helpful in the revision of this article, and the author would like to thank its organizers: Yuval Fisher, Geir Øien, Claude Tricot, and Jacques Levy-Vehel for their invitation. Individual communication with Michael Barnsley and John Hughes was also helpful.

References

1. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, New York, 1990.
2. P. Prusinkiewicz and M. Hammel, "Language Restricted Iterated Function Systems, Koch Constructions and L-Systems," in *New Directions for Fractal Modeling in Computer Graphics*, J.C. Hart, ed., Siggraph Course Notes, ACM, New

York, July 1994, pp. 4-1–4-14.

3. J.C. Hart, "The Object Instancing Paradigm for Linear Fractal Modeling," in *Proc. of Graphics Interface*, Morgan Kaufmann, Palo Alto, 1992, pp. 224-231.
4. Y. Fisher, ed., *Fractal Image Compression: Theory and Applications to Digital Images*, Springer-Verlag, New York, 1994.
5. M.F. Barnsley and L.P. Hurd, *Fractal Image Compression*, AK Peters, Wellesley, Mass., 1993.
6. J. Hutchinson, "Fractals and Self-Similarity," *Indiana Univ. Math. J.*, Vol. 30, No. 5, 1981, pp. 713-747.
7. M.F. Barnsley, J.H. Elton, and D.P. Hardin, "Recurrent Iterated Function Systems," *Constructive Approximation*, Vol. 5, 1989, pp. 3-31.
8. M.F. Barnsley, *Fractals Everywhere*, Academic Press, New York, 1988.
9. H.-O. Peitgen, H. Jurgens, and D. Saupe, *Chaos and Fractals*, Springer-Verlag, New York, 1994.
10. D. Saupe and R. Hamzaoui, "A Review of the Fractal Image Compression Literature," *Computer Graphics*, Vol. 28, No. 4, Nov. 1994, pp. 268-276.
11. A.E. Jacquin, "Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations," *IEEE Trans. Image Processing*, Vol. 1, No. 1, Jan. 1992, pp. 18-30.
12. J. Evans, ed., *The Natural Science Picture Sourcebook: 500 Copyright-Free Illustrations of Unusual Animals and Plants for Copying and Reference*, Van Nostrand Reinhold, 1984.
13. W.A. Bentley and W.J. Humphreys, *Snow Crystals*, Dover, 1962.
14. P. Prusinkiewicz and M.S. Hammel, "Escape-Time Visualization Method for Language-Restricted Iterated Function Systems," in *Proc. of Graphics Interface*, Morgan Kaufmann, Palo Alto, 1992, pp. 213-223.
15. D.H. Hepting and J.C. Hart, "The Escape Buffer: Efficient Computation of Escape Time for Linear Fractals," in *Proc. of Graphics Interface 95*, May 1995, pp. 204-214.

John C. Hart *is an assistant professor in the School of Electrical Engineering and Computer Graphics at Washington State University, and a faculty member of the Imaging Research Laboratory. His research focuses on the efficient representation of geometric detail and includes fractal geometry, implicit surfaces, natural modeling and scientific visualization. Hart is a member of the ACM, Siggraph, and the IEEE Computer Society, and is currently a Director-at-Large on the Siggraph Executive Committee.*

Readers may contact Hart at the School of EECS, Washington State University, Pullman, WA 99164-2752, e-mail hart@eecs.wsu.edu.