Fractal approximation of 2-D object

Jacques Lévy Véhel, André Gagalowicz

INRIA

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France
Tél (1) 39 63 55 11

Rapports de Recherche

N° 1187

Programme 6
Robotique, Image et Vision

FRACTAL APPROXIMATION OF 2-D OBJECT

Jacques LEVY VEHEL
André GAGALOWICZ

Mars 1990

FRACTAL APPROXIMATION OF 2-D OBJECT

APPROXIMATION FRACTALE D'OBJETS 2-D

Jacques LEVY VEHEL and André GAGALOWICZ

INRIA
Domaine de Voluceau
Rocquencourt - B.P. 105
78153 LE CHESNAY CEDEX
FRANCE

ABSTRACT

We present some new techniques for shape approximation with fractals, using Iterated Function System, a powerful method which allows good control on the resulting fractal. The main point discussed here can be stated as follows : given a grey level image A, find a few number of functions and associated probabilities that approximately generate A. Two directions have been explored : the first uses a gradient method, thus it was necessary to define a *smooth* error function ; the second one is based upon the ideas of simulated annealing. We then generalize the methods to a broader class of functions, and present some results.

Nous présentons de nouvelles techniques pour l'approximation fractale de formes utilisant les Systèmes de Fonctions Itérées. Ces systèmes constituent une méthode puissante qui assure un bon contrôle de la fractale obtenue. Le problème que l'on se pose est le suivant : étant donné une image de niveaux de gris A, trouver un petit nombre de fonctions et de probabilités associées qui génèrent approximativement A. Deux directions ont été explorées : la première utilise une méthode de gradient, impliquant la définition d'une fonction d'erreur dérivable ; la deuxième est fondée sur les idées du recuit simulé. Nous généralisons les méthodes à une classe plus vaste de fonctions et présentons des résultats.

A - INTRODUCTION

Fractals were defined by Mandelbrot [M1] to be subsets of R^N which Hausdorff-Besicovitch dimension exceeds their topological dimension. Many features make these sets of great interest for image synthesis ; for instance, the property that as one views the sets at greater magnification more and more structure is revealed, or the fact that only a small number of parameters are needed for their specification, independently of the visual complexity.

Fractals also provide often a convenient way to synthesize some natural objects with significant data compression. Fractal landscapes, clouds or vegetation are quite common.

The main drawback is the control that most fractal techniques allow over the resulting object : generally, one or two parameters (cf. [M1] and [F1]) control the fractal dimension and the size ; the problem is that no precise prediction of the result can be drawn from the parameters. Moreover, for a given pattern, no general method is available to adjust the parameters in order to make the result match this pattern.

B - ITERATED FUNCTION SYSTEM

The method of Iterated Function System (IFS) provides a much better control and understanding of the phenomena than classic fractal methods.

We briefly recall now the main aspects of this theory developed at Georgia Institute of Technology by Barnsley and al.

Let K be a d-dimensionnal Euclidean space.

An IFS is a couple (W, P) where :

$W = \{w_1, \ldots, w_n\}$ is a finite set of mappings of K into itself. Each w_i is a strict contraction (all eigenvalues have a magnitude less than one).

$P = \{p_1, \ldots, p_n\}$ is a set of probabilities : $\forall\ i, 0 \leq p_i \leq 1, \sum p_i = 1$.

We then consider the following process : let z_0 be any point in K, we randomly choose a map w_i (with probability p_i), compute $z_1 = w_i(z_0)$. We repeat this process a great number of times. If all points are plotted after a sufficiently great number of iterations, they will distribute themselves approximately upon a compact set A, called the attractor of the IFS.

Thus with every IFS is associated a unique attractor A, and a special probability measure, the p-balanced measure, which is the stationnary distribution of the random walk ; the probabilities, through this measure, have an influence on the density of the points of A (the grey levels).

A characterization of A is given by the following theorem (cf [B1]) :

if (W, P) is an IFS and A its attractor, then :

$$A = \bigcup_{i=1}^{n} w_i(A) \qquad (1)$$

The important "Collage Theorem" holds as follows (cf [B3]) :

Let the contraction mappings $w_i \;\; K \rightarrow K$, $i = 1, \ldots, n$ be chosen such that :

$$h(L, \bigcup_{i=1}^{n} w_n(L)) < \varepsilon, \qquad \text{for some } \varepsilon > 0$$

then :

$$h(L, A) < \frac{\varepsilon}{1-s}$$

where : A is the attractor of the IFS

h is the Haussdorff metric :

$$h(B, C) = \max[\max_{x \in B}(\min_{y \in C}(d(x,y))), \max_{y \in C}(\min_{x \in B}(d(x,y)))] \qquad (2)$$

$$0 < s < 1 \;/\; d(w_i(x), w_i(y)) \leq s\, d(x,y), \qquad \forall\, x, y \in K$$

It follows that, to find a suitable set of maps to reconstruct A, we need only make an approximate covering of "lazy tiling" of L by continuously distorted smaller copies of itself.

C - THE INVERSE PROBLEM

The design of fractal objects with IFS is made relatively simple and intuitive by equality (1). In fact, this method leads us to the inverse problem, which can be stated as follows :

Given a set A, determine an IFS that will approximately generate A.

In [B2], Barnsley describes a method to solve the inverse problem :

Using equality (1) and the moment theory of p-balanced measures for IFS ($M(\mu,m) = \int_K z^m \, d\mu(z)$), he shows that for every m, $M(\mu,m)$ can be computed recursively and explicitly in terms of $\{M(\mu,n), n = 0, 1, \ldots, m - 1\}$, with coefficients explicitly expressed with the parameters of the contractions. Then he computes the approximation of these moments obtained from a digitized image of the attractor A (the twindragon in the example of Barnsley), and he obtains approximately the IFS associated with A.

But, as stated in [B2], this method only works under certain hypotheses ; the most important one is that μ should be uniform upon A, which, in the case of two contractions w_1 and w_2, is only true when $w_1(A) \cap w_2(A) = \emptyset$ (in fact, $w_1(A) \cap w_2(A)$ may be not empty, but its "measure" should be much "lower" than the measure of A. For a more detailed explanation see [H1]).

In [L1], we have presented a method which yields good results even when the $w_i(A)$ do overlap each other ; it is a gradient-based method which uses the Collage Theorem and minimizes the Haussdorff distance between the set L and the attractor A. Because the Haussdorff distance is not a smooth function, we had to use a special algorithm for the optimization (a bundle algorithm), and also to perform some filtering on the intermediate images.

As in every optimization method, this one strongly depends on initialization : if the starting point in the contraction space is ill-chosen, the algorithm will get stuck in a local minimum. Thus, several runs with different starting points will often be necessary to find a suitable minimum, i.e a set of contractions whose attractor L looks very much like A.

Here, we are concerned with the problem of finding not only the contractions but also the associated probabilities *so that we will be able to generate the shape A plus the*

grey levels on A. Of course, in the general case, there might be no exact solution, however, we are interested in finding the "best" approximation.

We have developed two different methods :

- Define a suitable error function and use a gradient algorithm.

- Design a method based upon the ideas of simulated annealing.

Prior to explaining those methods, we recall some important properties of the stationnary measure upon the attractor, and state the fundamental equality that will allow us to develop both the gradient and the annealing methods.

D - SOME PROPERTIES OF THE INVARIANT MEASURE

I Notations

$W = \{w_1, \ldots, w_n\}$ is a finite set of mappings of K (complete metric space) into itself. Each w_i is a strict contraction.

$P = \{p_1, \ldots, p_n\}$ is a set of probabilities : $\forall\ i,\ 0 \leq p_i \leq 1,\ \sum p_i = 1$.

The attractor of the IFS (W,P) is denoted by A.

M is the set of Borel regular measures on K having bounded support and finite mass.

Let f be a continuous function on K, and suppose that f is Lipschitz. We define :

$$f_{\#} : \quad M \to M, \quad \mu \to f_{\#}\mu$$

$f_{\#}\mu$ is defined as follows :

for E, $K \supset E$: $\quad f_{\#}\mu(E) = \mu(f^{-1}(E))$

the support of a measure μ is the closed set :

$$\text{spt}\ \mu = X \setminus \cup \{V : V \text{ open}, \mu(V) = 0\}$$

II Theorem :

We define :

$$W(\gamma) = \sum_{i=1}^{n} p_i \, w_i \, \# \ (\gamma) \qquad \text{for } \gamma \in M$$

Then there exists a unique measure $\mu \in M$ such that :

$$W(\mu) = \mu \qquad (3)$$

This theorem follows from the fact that W is a contraction map in M for the metric :

$L(\mu,\gamma) = \sup \{\mu(\phi) - \gamma(\phi), \phi : K \rightarrow \mathbf{R}, \phi \text{ Lipschitz}\}.$

for a detailed proof see [H1].

This measure is called the invariant measure with respect to (W,P) and is such that :

$$\text{spt}\,\mu = A$$

μ has other interesting properties, for example, it has been shown [H1], that for any function a :

$$a : \qquad N^* \rightarrow \{1, \ldots n\},$$

the set :

$$\bigcap_{\rho=1}^{\infty} W_{a(1)} \circ W_{a(2)} \ldots \circ W_{a(\rho)}(A)$$

is a singleton $x_a \in A$. This simply means that if you apply recursively at A a randomly chosen sequence of w_i, you congerge to one point in A.

if we denote by Π the function :

$$\begin{aligned} C(n) &\rightarrow K \\ a &\rightarrow x_a , \end{aligned}$$

where :

C(n) is the set of maps : a : $N^* \rightarrow \{1 \ldots n\}$

then we have :

$$\mu = \Pi_{\#}\ \sigma$$

where σ is the product measure on C(n) induced by the measure p_i on each factor $\{1,\ldots n\}$. This means that the probability to obtain each point x lying upon A is proportional to the product of the p_i associated with the sequence of W_{ip} used to obtain x.

III Application

How can we use this theory ?

Equality (3) means that, for every E, $K \supset E$:

$$W(\mu)(E) = \mu(E)$$

that is :

$$\sum_{1=1}^{n} p_i\ \mu(w_i^{-1}(E)) = \mu(E) \qquad (4)$$

Our input, both for the optimization and annealing methods, will be a grey level image A, i.e. a function :

$$[1, NX] \times [1, NY] \rightarrow [0,1]$$
$$(i,j) \rightarrow A(i,j)$$

We define also :

$$A_p = w_p\ (A).$$

As it is easy to see, this means that, for every (i,j) :

$$A_p\ (i,j) = A(w_p^{-1}(i,j)) \qquad (5)$$

Let us suppose that our inverse problem has a solution. This means that there exists an IFS (W,P) such that the attractor of (W,P) is the sets of point (i,j) with : $A(i,j) > 0$, and that the invariant measure μ of (W,P) is proportionnal to the function A.

$$\mu = \lambda A, \ \lambda \text{ constant} \qquad (6)$$

Notice that the shape of the attractor does not depend on P but only on W ; μ depends on both W and P.

Using (4) and (6), we draw :

$$A(i,j) = \sum_{k=1}^{n} p_k \, A(w_k^{-1}(i,j))$$

Using (5) :

$$A(i,j) = \sum_{k=1}^{n} p_k \, A_k \, (i,j) \qquad (7)$$

As this equality is very important, since it will allow us to develop our methods for solving the inverse problem, we have verified it numerically. To do so, we have chosen a test image, a fern (originally presented by Barnsley), and we have applied equality (7) to it.

However, some problems arise when one tries to verify formulas like (1) or (7), as was noticed in [L1] : errors occur, due to the sampling of the source image, the resolution of the computed image, the finite number of iterations made to obtain the attractor, and round-off errors. In order to separate these errors from the possible ones due to the use of formula (7), we made two simulations :

- We "verify" first formula (1) ; we use a binary image for A, obtained from the fern by thresholding all positive points to one (figure 1, top left). We compute each binary image $w_i(A)$ and the union A' of them (figure 1, top middle). We then compute the error between the two images, that is the absolute value of (A - A'). This error, E1, only attributed to numerical factors, is on the top right of figure 1.

- To verify formula (7), we take the grey level fern F, (figure 1, bottom left), compute each grey level image $w_i(A)$, and finally the sum A" weighted by the p_i (figure 1, bottom middle). The error, E2, is again the absolute value of (F - A").

To make use of this result, we should find a way to separate the purely numerical errors ; an approximation of that is to compare E1 and E2, keeping in mind that E1 is a sort of lower bound for the error E2. As it makes no sense to compare a grey level image (E2) and a binary one (E1), we have made a double threshold on error image E2 : all points with value lower than t are forced to 0, while all values greater than t are forced to 1. t was chosen equal to 0.1 (grey level values go from 0 to 1). Note that this is a rather severe estimation of the error E2 : Points in E2 which are only slightly greater than the threshold, yet which yield a hardly noticeable difference between F and A", will contribute to our filtered error as much as point belonging to only one of the two images. The thresholded image E2' is on bottom right of figure 1.

To compare E1 and E2', we can compare their means :

mean of E1 :	0.0164
mean of E2' :	0.0168

The very low difference between these values as well as the visual comparison of the images show that the use of (7) instead of (1) does not introduce noticeable errors. The same conclusions can be drawn from all others simulation we made.

E - A GRADIENT-BASED METHOD

I The algorithm

The method is a numerical one, using an optimization algorithm.

The inputs are :

- A digitized image of the set A to generate ;
- Initial values of the contractions w_i of the IFS ;
- Initial values of the probabilities p_i.

The program outputs n contractions and probabilities which generate a set L*, that minimizes the error function F = d(L*, A) for a certain distance d.

Of course :

- For some A, the program won't give a good approximation ;
- For a given A, we can lower d(L*, A), provided that we increase n enough.

We use the following notations :

AB is an image (matrix NX*NY) which contains a numerical representation of the attractor A : AB(i,j) > 0 means that the point (i,j) is inside A. The values are normalized (AB(i,j) ∈ [0,1]).

$w_1, \ldots, w_n$ are the n contractions to be found.

$$w_k(i,j) = \begin{bmatrix} p_1^k & p_2^k \\ p_3^k & p_4^k \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} p_5^k \\ p_6^k \end{bmatrix}$$

for each k, w_k is a strict contraction : $\forall u \in R^2, |w_k(u)| \leq s_k |u|, s_k < 1$.

AC is an image which contains a numerical representation of $L = \cup w_i(A)$.

AB_k contains a numerical representation of $w_k(A)$.

We denote by G the gradient of F.

F is a function of 6n + n variables : the terms of the contractions p_1^k, and the n probabilities p_i ; G is a 7n vector, and we have :

contraction : $G(1) = \frac{\partial F}{\partial p_1^1}, \ldots, G(6) = \frac{\partial F}{\partial p_6^1}, G(7) = \frac{\partial F}{\partial p_1^2}, \ldots, G(13) = \frac{\partial F}{\partial p_6^2}, \ldots$

probability : $G(6n+i) = \frac{\partial F}{\partial p_i}$

To measure the distance between A and L we consider, for each pixel, the difference of intensity between A and L, i.e. :

$$F = d(L,A) = \sum_{i,j} [\, AC(i,j) - AB(i,j) \,]^2 \qquad (8)$$

We have : $d(L,A) = 0 \leftrightarrow L = A$.

To compute AC, following equality (7), we use the formula :

$$AC = \sum_i p_i AB_i$$

We first compute, for each k, $AB_k = w_k(AB)$, which means :

For every point (i,j) in AB, compute $w_k(i,j) = (x,y)$, and set :

$AB_k(E(x + 0.5), E(y + 0.5)) = AB(i,j)$

where $E(x + 0.5)$ means that we take the nearest integer to x.

F is thus defined as follows :

$$F = \sum_{i,j} [\sum_k p_k AB_k(i,j) - AB(i,j)]^2 \qquad (9)$$

The advantage of this criterion is that the computation of the gradient G is easy. We must first introduce the w_k in the definition of F.

It is straightforward to see that :

$$AC(i,j) = \sum p_k AB_k(i,j) = \sum p_k AB(w_k^{-1}(i,j))$$

We have : $w_k^{-1}(i,j) = (x,y)$, and the problem is that AB is defined only in N^2. Thus, to compute $AB(w_k^{-1}(i,j))$ we shall have to make, for instance, a bilinear interpolation between AB(E(x), E(y)), AB(E(x) + 1, E(y)), AB(E(x), E(y) + 1), AB(E(x) + 1, E(y) + 1).

We can now compute G : for j not greater than 6n, each G(j) is a $\frac{\partial F}{\partial p_m^l}$

$$\frac{\partial F}{\partial p_m^l} = \frac{\partial}{\partial p_m^l} [\sum_{i,j} [\sum_k p_k AB(w_k^{-1}(i,j)) - AB(i,j)]^2]$$

$$\frac{\partial F}{\partial p_m^l} = \sum_{i,j} \frac{\partial}{\partial p_m^l} [\sum_k p_k AB(w_k^{-1}(i,j)) - AB(i,j)]^2$$

$$\frac{\partial F}{\partial p_m^l} = 2\sum_{i,j} \frac{\partial}{\partial p_m^l} p_l AB(w_l^{-1}(i,j))[\sum_k p_k AB(w_k^{-1}(i,j)) - AB(i,j)]$$

$$\frac{\partial F}{\partial p_m{}^l} = 2 \sum_{i,j} \frac{\partial}{\partial p_m{}^l} AB(w_l{}^{-1}(i,j)) \times p_l \, [\, AC(i,j) - AB(i,j)]$$

but $$w_k(i,j) = \begin{bmatrix} p_1{}^k & p_2{}^k \\ p_3{}^k & p_4{}^k \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} p_5{}^k \\ p_6{}^k \end{bmatrix}$$

and then :

$$w_k{}^{-1}(i,j) = D \times \begin{bmatrix} p_4{}^k(i - p_5{}^k) - p_2{}^k(j - p_6{}^k) \\ \\ p_3{}^k\,(i - p_5{}^k) - p_4{}^k\,(j - p_6{}^k) \end{bmatrix}$$

where $D = (p_1{}^k\, p_4{}^k - p_2{}^k\, p_3{}^k)^{-1}$

And so :

$$\frac{\partial}{\partial p_m{}^l} AB(w_l{}^{-1}(i,j)) = \frac{\partial AB}{\partial i}(w_l{}^{-1}(i,j)) \times \frac{\partial [D(p^l{}_4(i-p^l{}_5) - p^l{}_2(j-p^l{}_6))]}{\partial p_m{}^l}$$

$$+ \frac{\partial AB}{\partial j}(w_l{}^{-1}(i,j)) \times \frac{\partial [D(p^l{}_3(i-p^l{}_5) - p^l{}_1(j-p^l{}_6))]}{\partial p_m{}^l}$$

We notice that, to compute G, we need to know the derivatives in the image AB.

The first six elements of G are :

$$G(1) = \sum_{i,j} X \times C_1$$

$$G(2) = \sum_{i,j} Y \times C_1$$

$$G(3) = \sum_{i,j} X \times C_2$$

$$G(4) = \sum_{i,j} Y \times C_2$$

$$G(5) = \sum_{i,j} C_1$$

$$G(6) = \sum_{i,j} C_2$$

where :

$$X = D \times [p_4^1(i - p_5^1) - p_2^1(j - p_6^1)]$$

$$Y = D \times [-p_3^1(i - p_5^1) + p_1^1(j - p_6^1)]$$

$$C_1 = p_1 \times D \times [AC(i,j) - AB(i,j)] \times [-p_4^1 \frac{\partial AB}{\partial i}(X,Y) + p_3^1 \frac{\partial AB}{\partial j}(X,Y)]$$

$$C_2 = p_1 \times D \times [AC(i,j) - AB(i,j)] \times [p_2^1 \frac{\partial AB}{\partial i}(X,Y) - p_1^1 \frac{\partial AB}{\partial j}(X,Y)]$$

The derivatives of F according to the probabilities are easy to compute and are simply :

$$G(6n + k) = 2 \sum_{i,j} AB_k(i,j) \times [\, AC(i,j) - AB(i,j)]$$

The optimization algorithm itself uses a quasi-Newton method and a Rosen-Goldfarb active constraints strategy (the constraints are : w_i contractions, and sum of probabilities equal to one).

II Results

The algorithm has been tested on two types of images : "classical" fractals (the twindragon) and "natural" shapes (leaves).

When initial values for W and P were randomly chosen, several runs (between 10 and 20) were necessary to find a good minimum ; however, when these values were chosen such that the initial image more or less covers the attractor, less than 5 runs were generally sufficient.

The best result on the twindragon gave an error lower than one per cent between the reference image and the computed shape.

The artificial leaf presented in figure 2 is generated by 4 contractions with equal probabilities ; on figure 3, we have kept the same W, but P is now :

$$P = (0.35, 0.15, 0.35, 0.15).$$

In both cases, the best error found was less than 2 per cent, which is hardly noticeable in an image.

The last test was made on a natural leaf, picked from a maple tree on the grounds, at INRIA.This leaf is shown in figure 4, and the result of the algorithm is on figure 5. This time, it is easy to see a difference between the two images, but we must recall that the reference shape *is not* a fractal, and the only thing the algorithm can do is to find a good fractal approximation for it.

F - ANNEALING

I Theory

Simulated Annealing is a powerful technique for finding the global minimum of a function when a great number of parameters have to be taken into account. It is based upon an analogy with the annealing of solids, where a material is heated to a high temperature then very slowly cooled in order to let the the system reach its ground energy. The delicate point is to lower T not too rapidly, thus avoiding local minima. Application to other optimization problems is done by generalizing the states of the physical system to some defined states of the system being optimized, and generalizing the temperature to a control parameter for the optimization process ; most of the time, the Metropolis algorithm is used : at "temperature" T, the jump from a state of energy E to a state of energy E' is made with probaility one if E' is lower than E and with a probability proportionnal to (E - E')/T if not.

To simulate annealing, we define :

S = space of states = $\{X = (x_1,x_2,....x_{6*n+6})\}$ for a given n, such that :

w_1 is defined by the six parameters : (x_1, x_2, x_3, x_4) for the contraction matrix and (x_5, x_6) for the translation and $P = (x_{6*n+1}, ..., x_{6*n+6})$.
Each point in S verify the constraints :

(C1) : each w_i is a contraction.

(C2) : $x_{6*n+1} + x_{6*n+2} + ...+ x_{6*n+6} = 1$

μ = space of moves.

μ is a set of neighborhood relations over S. For a given state X, μX is the set of states that we can reach from X using a single iteration. Requirements on μ are obvious : for every (X,X') in S, if X' is in μX, then X should be in $\mu X'$; no point is in its own neighborhood ; and it must be possible to go from any point

X to any other point X' while staying in μ

We choose the moves to be displacements along one coordinate axis in R^{6*n+6}. Thus every element in μX differs from X by one and only one x_i.

ß = selection probability

ß is a set of probability functions $ß_X$ such that :

$$ß_X : \quad \mu X \rightarrow [0,1] \qquad X' \rightarrow ß_X(X')$$

ß gives the probability to select one move in μX. As said in [O1], the only requirements upon ß are :

$$\forall\, ß_X, \qquad \sum_{X' \varepsilon\, \mu X} ß_X(X') = 1$$

and :

$$\forall\, (X,X') \qquad ß_X(X') = ß_{X'}(X)$$

In particular, ß does not need to remain constant when the temperature decreases. We shall use this property to improve our algorithm : at high temperatures, we allow moves of any length (that is : modulus of X' - X can be arbitrarily great) in order to be sure of potentially reaching any point in S, while at low temperatures, when we think we are near the minimum, we shall only examin points in the near neighborhood of X.

π = acceptance probability

π is the probability of accepting one move, once it has been selected. Let F(X) be the energy of state X. In order to simulate annealing, we should have :

$\lim_{T\rightarrow\infty} \pi(F(X),F(X'),T) = 1$ (at high temperature, all states have equiprobability)

$\lim_{T\rightarrow 0} \pi(F(X),F(X'),T) = 0$ (at temperature 0, no moves are allowed)

We shall use :

$$\pi(F(X),F(X'),T) = 1 \qquad \text{if} \qquad F(X') \leq F(X)$$

$$\pi(F(X),F(X'),T) = \exp[(F(X) - F(X'))/T] \qquad \text{else.}$$

The algorithm will then generate a Markov chain according to the simple following process :

- choose in S an initial point X

- for T going down from T0 to Tf, according to the lowering schedule f(T) :
  select at random a point X' in S according to ß.
  compute $\pi(X,X',T)$.
  pick a random number a in [0,1].
  if $a \leq \pi(X,X',T)$, then X = X'.

The problem is then to choose T0,Tf, and the schedule function f(T).

Let F_M (resp. F_m) be the maximum (resp. the minimum) of F upon S, and N be the number of independant variables. S. and D. Geman have shown [G1] that if :

$N(F_M - F_m)/\log k \leq T(k)$ at any iteration k greater than 2,

then X converges with probability one to the minimum when $k \to \infty$.

In our case, typical values are : N=30, $F_M - F_m = 1000$. If we want to end the process when it is frozen (no more moves occur), we must reach a temperature lower than 10. The schedule given above would then lead us to an unwieldy number of iterations, approximately exp(3000).

II Implementation

Fortunately, the process happens to converge much faster in most cases we tested.

We can lower T faster (linear law) provided that, between each change of temperature, the system has the time to go back to quasi-equilibrium. This is achieved by making several iterations at each temperature (about 50). As for T0, we have not been able to improve the value given by the theory, and the typical starting T is 10,000. We end at Tf = 10, thus we still must perform 50,000 iterations. Keeping the notations of E - I, one iteration consists in :

a) computing the $p_k AB_k$ for the w_k or p_k corresponding to the x_i that changed.

b) update AC using the new value of AB_k

c) compute F and update X

Obviously, steps a) b) and c) are time-consuming ; this is why we have optimized these computations : the code for those routines have been written in assembler, and parallelized as much as possible (32 points of AB_k and AC are computed at a time). Finally, the desired result for X is obtained in a few hours (between 1 and 4).

Note : To find the minimum, we visit about 50,000 points in the case of 4 contractions ; this may seem like a lot, but we should keep in mind the huge size of S, which is the subset of $\mathbf{R}^{30}$ defined by the constraints (C1) and (C2).

III Results

We applied the algorithm to the test images previously presented. In all cases, and no matter how the starting values had been chosen, the annealing algorithm found a minimum (W,P), which, when used as an input for the synthetizing process described in **B**, produced images very near to the reference (less than 5 per cent of error). We show some new pictures in next section.

G - GENERAL FUNCTIONS

I Introduction

Until now, we have restricted the w_i to be affine functions : more exactly, our algorithms for the inverse problem can only find a set of affine functions for approximating a given shape. But nothing in the theory of IFS imposes such a limitation. The only constraint on the functions is that they must be contractions (in fact, it has been showed that only an average condition of contraction is needed, see [B4]).

Actually, it is easy to compute attractors for a set of non-affine contractive functions, and the generated images, which are sometimes surprising, have many more diverse shapes than before. Generally, they are not self-affine, which allows us to construct less "regular" patterns.

We have tried to use two classes of functions : homographic and sine polynomials. We have generated images by functions of this type, and applied our algorithms to them.

II Homographic functions

Instead of writing :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p1 & p2 \\ p3 & p4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p5 \\ p6 \end{bmatrix}$$

where A is a 2x2 matrix and T a 2x1 vector, we can use an homogeneous matrix :

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} p1 & p2 & p5 \\ p3 & p4 & p6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The idea is then to replace the two 0 of the last line by any number, and make the calculations as in projective geometry ; we shall have :

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} p1 & p2 & p5 \\ p3 & p4 & p6 \\ a & b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

To stay in the plane, we state :

$x''=x'/z'$ $\qquad$ $y''=y'/z'$

In the general case, to ensure that these functions are contractions will lead to very complicated equations. However, it is easy to verify it for given numbers. Besides, this problem does not really arise in the analysis process (i.e. when we run the algorithm for solving the inverse problem) : if at a certain step, a function does not

contract, the algorithm will eliminate it because its error will be large.

To test our algorithm, we designed the following images :

a) The "smoke" in figure 6 has been generated with the following functions :

w_1: $x'' = -0.85\,x$

$y'' = -0.85\,y - 0.5$

w_2 : $x'' = (-0.8\,x - 2.5)/(0.048\,y + 1)$

$y'' = -0.32\,y / (0.048\,y + 1)$

with equiprobabilities.

b) figure 7 was generated with :

w_1 : $x'' = 0.33\,x$

$y'' = 0.33\,y + 0.33$

w_2 : $x'' = (0.66\,x + 0.33\,y + 0.33)/(0.33\,y + 1)$

$y'' = (0.85\,y + 0.4)/(0.33\,y + 1)$

w_3 : $x'' = (0.66\,x + 0.33\,y + 0.33)/(0.33\,y + 1)$

$y'' = (0.85\,y - 0.12)/(0.33\,y + 1)$

with equiprobabilities.

b) figure 8 was generated with :

w_1 : $x'' = 0.8\,x/(3.33\,x + 1)$

$y'' = 0.8\,y/(3.33\,x + 1)$

w_2 : $x'' = (0.4\,y + 0.15)/(4.1\,y + 1)$

$y'' = (0.4\,x - 0.3)/(4.1\,y + 1)$

w_3 : $x'' = (0.4\,y + 0.15)/(4.1\,y + 1)$

$y'' = (-0.4\,x + 0.3)/(4.1\,y + 1)$

w_4 : $x'' = (0.8\,x - 0.12)/(3.33\,x + 1)$

$y'' = -0.8\,y/(3.33\,x + 1)$

w_5 : $x'' = 0.4\,y /(3.5\,y + 1)$

$y'' = (0.4\,x + 0.3)/(3.5\,y + 1)$

w_6 : $x'' = 0.4\,y /(3.5\,y + 1)$

$y'' = (-0.4\,x - 0.3)/(3.5\,y + 1)$

with $P = (0.2,0.2,0.2,0.2,0.1,0.1)$

We have tested the annealing algorithm on these images, with the obvious modifications

brought by the new class of function (there are 3 more parameters for each contraction). Once again, the results were satisfying, with errors lower than 5 per cent.

III Sine polynomials

For the second class of functions, we state :

$x' = a \cos x + b \sin y + e$
$y' = c \sin x + d \cos y + f$

We generated an image with the very simple following values :

w_1 : $a = d = 1, b = c = e = f = 0.$
w_2 : $a = d = e = f = 0, b = c = 1.$
w_3 : $a = b = 0.5, c = e = f = 0, d = -0.5.$
w_4 : $a = e = f = 0, b = c = d = -0.5.$
with equiprobabilities.

The result is shown in figure 9 ; as previously, the annealing algorithm was able to find (W,P) with an error lower than 5 per cent.

H - CONCLUSION

We have designed two algorithms for solving our inverse problem : the faster one, the gradient method, proves to be able to reconstruct grey levels images but is not very reliable, due the large number of local minima of the error function. The annealing method is more robust and has demonstrated good results for affine, homographic functions, as well as for sine polynomials.

Of course, the problem that remains is the one of *a priori* choosing the class of functions for a given shape ; with some practice, one can see if a pattern is self-similar, or self-affine ; for the set of shapes bound to be generated by homographic functions or polynomials, it seems harder, then for the general case... Presently, we do not have any solutions for that problem.

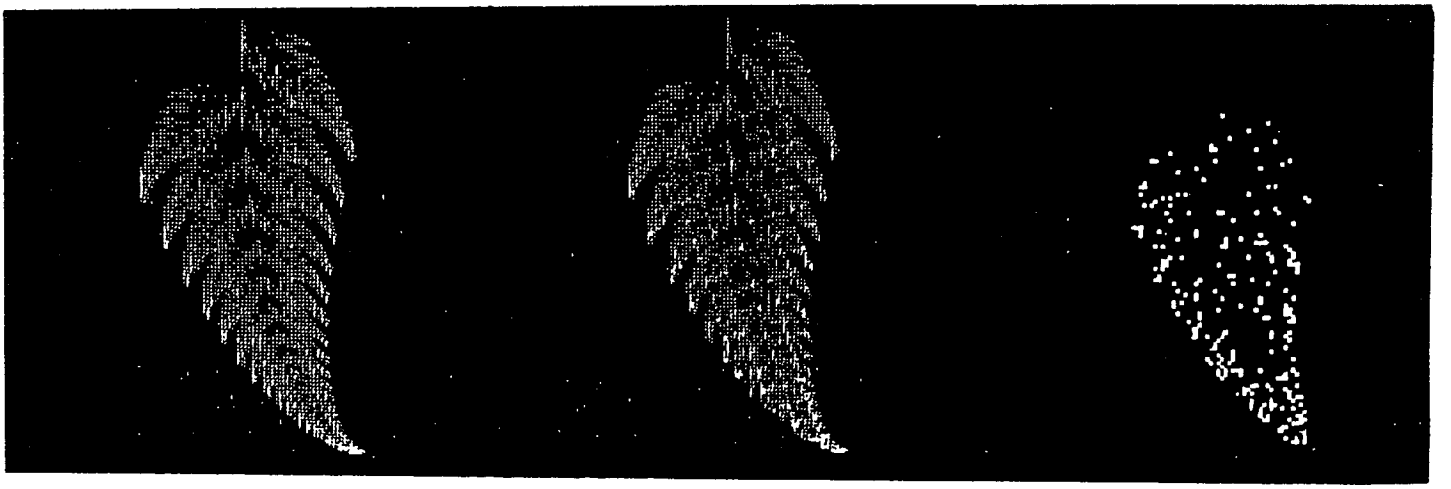
Another desirable extension is the resolution of the 3D case, which implies a

drastic reduction of the current computation time.

Acknowledgements :

Many thanks to Robert Ehrlich for great help in implementing the parallel version of the annealing alogrithm and for many interesting talks ; I also thank Christine Graffigne for introducing me to simulated annealing, Rachid Deriche and Chuck Hansen.

check of formula (1) ; binary images



check of formula (7) ; grey level images

figure 1

Figure 4 : natural leaf, reference image.

Figure 5 : natural leaf, computed image.

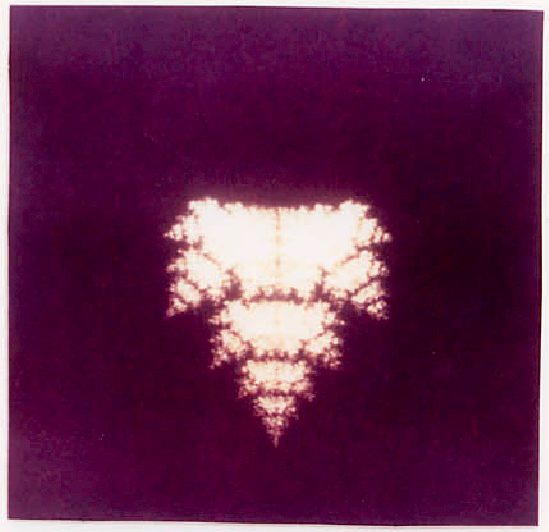


Figure 2 : artificial leaf, equiprobability.

Figure 3 : artificial leaf, P = (0.35, 0.15, 0.35, 0.15).

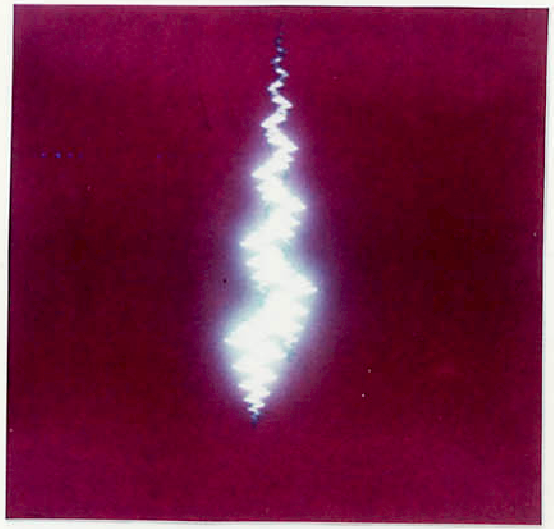


Figure 6 : "smoke".

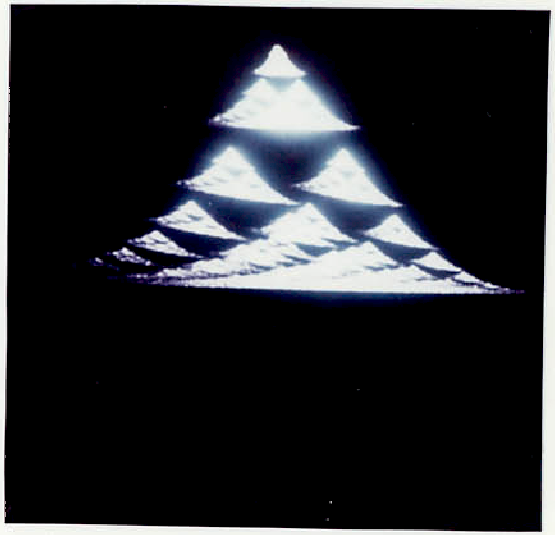


Figure 7.

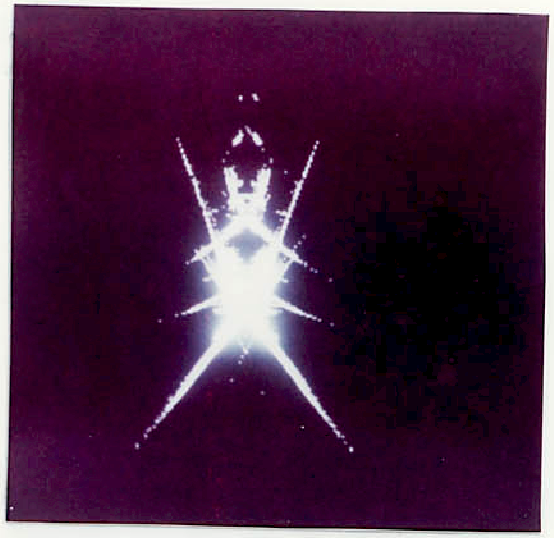


Figure 8 : humanoid.

Figure 9 : use of sine polynomials.

REFERENCES

[A1] E. Aarts and P. van Laarhoven, "Simulated annealing : a pedestrian review of the theory and some applications", NATO ASI Series, Vol. F30.

[B1] M. Barnsley, "Making chaotic dynamical systems to order".

[B2] M. Barnsley, S. Demko, "Iterated function system and the global construction of fractals", Proceedings of the Royal Society, A 399, 243-245, 1985.

[B3] M. Barnsley, V. Ervin, D. Hardin and J. Lancaster, "Solution of an inverse problem for fractals and other sets", Proc. Natl. Acad. Sci. USA, Vol 83, 1986.

[B4] M. Barnsley, S. Demko, J. Elton, J; Geronimo, "Invariant measures for Markov processes arising from Iterated function systems with place-dependent probabilities".

[D1] S. Demko, L. Hodges, B. Naylor, "Construction of fractal objects with IFS", Siggraph, Vol 19, Number 3, 1985.

[F1] Fourier, Alain, Don Fussel, Carpenter, "Computer rendering of stochastic models", Communication of the ACM, 25 (June 1982).

[G1] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesien restorarion of images", Trans. PAMI-6, No 6, November 1984, 721, 741.

[H1] J. Hutchinson, "Fractals and self-similarity", Indiana University Journal of Mathematics 30, 713-747, 1981.

[L1] J. Lévy Véhel and A. Gagalowicz, "Shape approximation by a fractal model", Eurographics'87.

[M1] B. Mandelbrot, "The fractal geometry of nature", W.H Freeman and Co., San Francisco, 1983.

[O1] R. Otten and L. van Ginneken, "Annealing : the algorithm", RC 10861, 12/3/84, Mathematics.