Image compression using recurrent bivariate fractal interpolation surfaces

P. Bouboulis* L. Dalla ‡ V. Drakopoulos†

March 1, 2005

Abstract

A new method for constructing recurrent bivariate fractal interpolation surfaces through points sampled on rectangular lattices is proposed. This offers the advantage of a more flexible fractal modelling compared to previous fractal techniques that used affine transformations. The compression ratio for the above mentioned fractal scheme is higher than other fractal methods or JPEG, though not as high as JPEG2000. Theory, implementation and analytical study are also presented.

keywords: Fractal interpolation, IFS, fractal image coding, fractals, bivariate fractal interpolation surfaces, fractal image compression

1 Introduction

The theory of image coding using an *iterated function system*, or *IFS* for short, was first proposed by Barnsley [1993]. Barnsley modeled real-life images by means of fractal objects, i.e., by the *attractors*, or by the invariant measure supported by the attractors, evolved through iterations of a set of contractive affine transformations. With the help of IFS's along with a collage theorem, he laid the foundation of the fractal-based image compression. A set of contractive affine transformations can approximate a real image and so, instead of storing the whole image, it is enough to store the relevant parameters of the transformations reducing memory requirements and achieving high compression ratios.

The effectiveness of fractal image compression has been demonstrated by Jacquin [1992], Barnsley & Hurd [1993] and Fisher et al. [1995]. They have shown that a well-designed fractal compressor yields comparable compression ratios and image quality to the JPEG algorithm. However, fractal compression is a heavily unbalanced technique; the computational requirements of the compression algorithm are orders of magnitude greater than those of the decompressor. An overview of the variety of schemes that have been investigated can be found in [Wohlberg, 1999]. The book by Lu [1997] combines introductory material with an in-depth discussion of many aspects of fractal coding.

Our intention is to extend the 2-dimensional fractal interpolation models in order to construct (continuous) *fractal interpolation surfaces* (or FIS's for short) of a non-affine character and to explore their potential use in the context of image interpolation. In this respect, we consider points of an image to be samples (on a uniform rectangular grid) of a continuous surface. The use of bivariate FIS's following the recurrent IFS formalism permits the representation of static images with a fractal model that is more general, flexible and computationally efficient compared with other fractal techniques which use FIS's like in [Price, 1998] and [Drakopoulos, 2004].

*Department of Informatics and Telecommunications, Telecommunications and Signal Processing, University of Athens, Panepistimioupolis 157 84, Athens, Hellas (macaddic@otenet.gr).

†Department of Informatics and Telecommunications, Theoretical Informatics, University of Athens, Panepistimioupolis 157 84, Athens, Hellas (vasilios@di.uoa.gr).

‡Department of Mathematics, Mathematical Analysis, University of Athens, Panepistimioupolis 157 84, Athens, Hellas (ldalla@math.uoa.gr).

2 The 2D Recurrent Bivariate Model

An *hyperbolic* IFS is defined as a pair consisting of a complete metric space (X, ρ) and a finite set of contractive transformations $w_i: X \to X,\ i = 1, 2, \ldots, M$. It is often convenient to write an IFS formally as $\{X; w_1, w_2, \ldots, w_M\}$ or, somewhat more briefly, as $\{X; w_{1-M}\}$. The *attractor* of a hyperbolic IFS is the unique set A_∞ for which $A_\infty = \lim_{k\to\infty} W^k(A_0)$ for every starting set A_0, where

$$W(A) = \bigcup_{i=1}^{M} w_i(A) \text{ for } A \in \mathcal{H}(X),$$

and $\mathcal{H}(X)$ is the metric space of all nonempty compact subsets of X with respect to the Hausdorff metric.

The Collage Theorem (see [Fisher, 1995]) provides a measure of the goodness of fit of an attractor associated with an IFS and a given nonempty compact set. An attractor that is close to a given set is one with an associated IFS such that the union of all the maps applied to the given set is close to the given set. The closer the union is to the given set, the closer the attractor of the IFS will be to the given set. Therefore, in order to test the closeness of an attractor to a given set, one need not compute the attractor itself. The collage theorem, however, is not constructive, it does not indicate how to find a set of proper maps, but rather, it provides a way to test an IFS without need for computation of the attractor. In this work we use IFS on $\mathbb{R}^3$ to model images for data compression. Given an image $z = f(x, y)$ that gives the grey level at each point (x, y), we can use the attractor of such an IFS to approximate it.

Let $X = [0, 1] \times [0, p] \times \mathbb{R}$. Let $S = \{(x_i, y_j, z_{ij}) : i = 0, 1, \ldots, N;\ j = 0, 1, \ldots, M\}$ be an interpolating set with $(N + 1)(M + 1)$ *interpolation points* such that $0 = x_0 < x_1 < \cdots < x_N = 1$ and $0 = y_0 < y_1 < \cdots < y_M = p$. Using the interpolation points, we partition $[0, 1] \times [0, p]$ into NM rectangles $I_{ij} = [x_{i-1}, x_i] \times [y_{j-1}, y_j],\ i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$ which we call *sections* or *regions*. Furthermore, let $Q = \{(\hat{x}_k, \hat{y}_l, \hat{z}_{kl}) : k = 0, 1, \ldots, K;\ l = 0, 1, \ldots, L\}$ be a set with $(K+1)(L+1)$ points with $Q \subset S$ $(Q \neq S)$, such that $0 = \hat{x}_0 < \hat{x}_1 < \cdots < \hat{x}_K = 1$ and $0 = \hat{y}_0 < \hat{y}_1 < \ldots < \hat{y}_L = p$. Using the points of Q we partition $[0, 1] \times [0, p]$ into KL rectangles $J_{kl} = [\hat{x}_{k-1}, \hat{x}_k] \times [\hat{y}_{l-1}, \hat{y}_l], k = 1, 2, \ldots, K$ and $l = 1, 2, \ldots, L$ which we simply call *intervals* or *domains*. It is evident that for every domain there are some regions lying inside.

Let $\mathbb{J}$ be a labelling map such that $\mathbb{J}: \{1, 2, \ldots, N\} \times \{1, 2, \ldots, M\} \to \{1, 2, \ldots, K\} \times \{1, 2, \ldots, L\}$ with $\mathbb{J}(i, j) = (k, l)$. Define mappings $w_{ij}: X \to \mathbb{R}^3$,

$$w_{ij}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_{ij}x + k_{ij} \\ d_{ij}y + l_{ij} \\ e_{ij}x + g_{ij}y + h_{ij}xy + s_{ij}z + m_{ij} \end{pmatrix}, \tag{1}$$

such that

$$\begin{gathered} w_{ij}\begin{pmatrix} \hat{x}_{k-1} \\ \hat{y}_{l-1} \\ \hat{z}_{k-1,l-1} \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ y_{j-1} \\ z_{i-1,j-1} \end{pmatrix}, w_{ij}\begin{pmatrix} \hat{x}_{k} \\ \hat{y}_{l-1} \\ \hat{z}_{k,l-1} \end{pmatrix} = \begin{pmatrix} x_{i} \\ y_{j-1} \\ z_{i,j-1} \end{pmatrix}, \\ w_{ij}\begin{pmatrix} \hat{x}_{k-1} \\ \hat{y}_{l} \\ \hat{z}_{k-1,l} \end{pmatrix} = \begin{pmatrix} x_{i-1} \\ y_{j} \\ z_{i-1,j} \end{pmatrix} \quad \text{and} \quad w_{ij}\begin{pmatrix} \hat{x}_{l} \\ \hat{y}_{l} \\ \hat{z}_{kl} \end{pmatrix} = \begin{pmatrix} x_{i} \\ y_{j} \\ z_{i,j} \end{pmatrix} \end{gathered} \tag{2}$$

for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. We refer to s_{ij} as the *contactivity* or *vertical scaling factors*. The w_{ij} map the vertices of the domain $J_{kl} = J_{\mathbb{J}(i,j)}$ to the vertices of the region I_{ij} (see Fig. 1). Define the function u_{ij} by

$$u_{ij}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{ij} & 0 \\ 0 & d_{ij} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} k_{ij} \\ l_{ij} \end{pmatrix} = \begin{pmatrix} \phi_{ij}(x) \\ \psi_{ij}(y) \end{pmatrix},$$

so that $w_{ij} = (u_{ij}, F_{ij})$. Then

$$u_{ij}(J_{\mathbb{J}(i,j)}) = I_{ij}, \quad i = 1, 2, \ldots, N \text{ and } j = 1, 2, \ldots, M.$$

Bivariate maps of this form have the property that vertical lines are mapped to vertical lines contracted by the factor s_{ij}. From the above constraints a system of 8 linear equations arises; thus, can always be solved for the parameters a_{ij}, k_{ij}, d_{ij}, l_{ij}, e_{ij}, g_{ij}, h_{ij}, m_{ij} in terms of the coordinates of the interpolation points and the *vertical scaling factors* (see [Dalla, 2002], [Xie, 1997]). Finally, let Φ: $\{1,2,\ldots,N\}\times\{1,2,\ldots,M\}\to\{1,2,\ldots,NM\}$ be a 1-1 function (i.e. an enumeration of the set $\{(i,j): i=1,2,\ldots,N; j=1,2,\ldots,M\}$).

A *recurrent IFS* (or RIFS for short) *associated* with the set of data S consists of the IFS $\{X;\ w_{1-N,1-M}\}$ and a row-stochastic matrix $P=(p_{nm}\in[0,1]:\ n,m\in\{\Phi(i,j),\ i=1,2,\ldots N; j=1,2,\ldots,M\})$, such that

$$\sum_{n=1}^{NM} p_{mn}=1,\quad m=1,2,\ldots,NM. \tag{3}$$

The recurrent structure is given by the *connection matrix* $C=(C_{nm})$ which is defined by

$$C_{nm}=\begin{cases}1, & \text{if } p_{mn}>0\\ 0, & \text{if } p_{mn}=0,\end{cases}$$

where $n,m=1,2,\ldots,NM$. The transition probability for a certain discrete time Markov process is p_{nm}, which gives the probability of transfer into state m given that the process is in state n. Condition (3) says that whichever state the system is in (say n), a set of probabilities is available that sum to 1, and they describe the possible states to which the system transits at the next step.

We assume that $pN,\ pK\in\mathbb{N}$, the regions (defined by the interpolation points) are squares of side $\delta=1/N$, while the domains are squares of side $\Delta=1/K$ (thus $M=pN$ and $L=pK$) and the number $a=\Delta/\delta=N/K$ is an integer greater than one. The number a^2 expresses the number of regions contained in any domain.

If we define the enumeration $\Phi(i,j)=(i-1)M+j,\ i=1,\ldots,N$ and $j=1,\ldots,M$, then $\Phi^{-1}(n)=((n-1)\operatorname{div}M+1,\ (n-1)\bmod M+1),\ n=1,\ldots,NM$ and the $NM\times NM$ stochastic matrix P is defined by

$$p_{nm}=\begin{cases}1/a^2, & \text{if } I_{(n-1)\operatorname{div}M+1,(n-1)\bmod M+1}\subseteq J_{\mathbb{J}((m-1)\operatorname{div}M+1,(m-1)\bmod M+1)}\\ 0, & \text{otherwise}\end{cases}$$

The basic concept behind fractal image compression using FIS's is simple. Suppose we are given a digitised image f that we wish to encode. Peak some image pixels and construct the interpolating set. Usually these pixels have a constant distance δ along the horizontal and vertical direction. Using the interpolation points we form a rectangular grid on $\mathbb{R}^2$ and call the emerging rectangles (of side length δ) regions. Select then, some of the interpolation points and construct the set Q. Usually these pixels have a constant distance Δ along the horizontal and vertical direction, where Δ is a multiple of δ. Using the points of Q form another rectangular grid and call the emerging rectangles (of side length Δ) domains. The goal is to find for each region I_{ij} a map w_{ij} that maps the pixels of a domain J_{kl} (where $(k,l)=\mathbb{J}(i,j)$) to the pixels of the specific region I_{ij} as close to the original pixels as possible. Storing the interpolation points and the parameters which describe each w_{ij} we can generate the attractor of the formed RIFS via the Deterministic Iteration Algorithm (see [Barnsley, 1993a]).

If the vertical scaling factors obey $0<|s_{ij}|<1$, then there is a metric d on X, equivalent to the Euclidean metric, such that the RIFS $\{X;w_{1-N,1-M},P\}$ is hyperbolic with respect to d and has A_∞ as its attractor. The following proposition gives the conditions needed in order to construct interpolating fractal functions (see also [Bouboulis et al., 2004]).

Proposition 1 *Let the RIFS be defined above and consider the domain* $J_{kl},\ k=1,2,\ldots,K,\ l=1,2,\ldots,L$. *Suppose that*

$LEFT_{kl}[\nu],\quad \nu=1,\ldots,a-1$, *denote the vertical distance of each interpolation point* $\{(x_{(k-1)a},y_{(l-1)a+\nu},z_{(k-1)a,\,(l-1)a+\nu}),\ \nu=1,\ldots,a-1\}$ *from the line defined by the points*

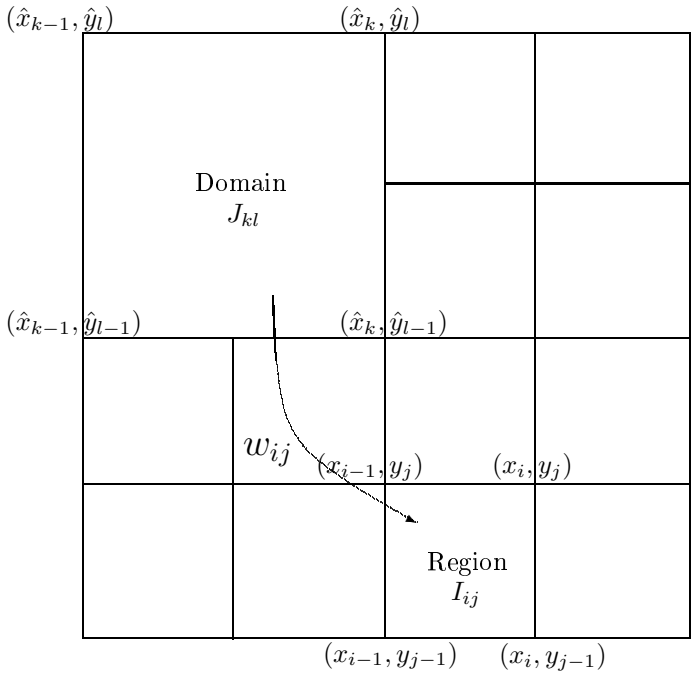


Figure 1: Mapping domains of an image to regions of an image.

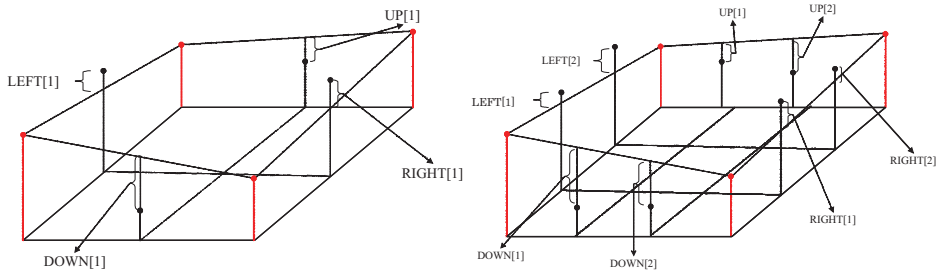
$RIGHT_{kl}[\nu]$, $UP_{kl}[\nu]$, $DOWN_{kl}[\nu]$, $\nu = 1, \ldots, a-1$, *denote the vertical distances of each interpolation point* $\{(x_{ka}, y_{(l-1)a+\nu}, z_{ka,\,(l-1)a+\nu}),\ \nu = 1, \ldots, a-1\}$ *from the line defined by the points* $(x_{(k-1)a}, y_{(l-1)a}, z_{(k-1)a,\,(l-1)a})$ *and* $(x_{ka}, y_{la}, z_{ka,\,la})$.

Each one of these vertical distances is taken positive, if the corresponding interpolation point is above the corresponding line; otherwise is taken negative (see Fig. 2). Let the vertical scaling factors obey

$$\begin{aligned} s_{i,j} \cdot RIGHT_{\mathbb{J}(i,j)}[\nu] &= s_{i+1,j} \cdot LEFT_{\mathbb{J}(i+1,j)}[\nu] \\ s_{i,j} \cdot UP_{\mathbb{J}(i,j)}[\nu] &= s_{i,j+1} \cdot DOWN_{\mathbb{J}(i,j+1)}[\nu] \\ \text{for } i,\ j,\ \nu \in \mathbb{N}: \quad i = 1, \ldots, N-1, &\qquad j = 1, \ldots, M-1,\ \ \nu = 1, \ldots, a-1. \end{aligned}$$

Then A_∞ will be the graph of a continuous function $f: [0,1] \times [0,p] \to \mathbb{R}$ that interpolates the data $S = \{(x_i, y_j, z_{ij}) : i = 1, 2, \ldots, N; j = 1, 2, \ldots, M\}$.

Definition 1 *The function f whose graph is the attractor of the RIFS described in Prop. 1 is called the* fractal interpolation surface *corresponding to the data $\{(x_i, y_j, z_{ij}) : i = 1, 2, \ldots, N; j = 1, 2, \ldots, M\}$*



(a) (b)

Figure 2: The condition of continuity. (a) $a = 2$, (b) $a = 3$

3 The Proposed Algorithms

Let $I_{ij} = [x_{i-1}, x_i] \times [y_{j-1}, y_j]$ be one of the regions. Let, also, $J_{kl} = [\hat{x}_{k-1}, \hat{x}_k] \times [\hat{y}_{j-1}, \hat{y}_j]$ be one of the corresponding domains. It is obvious, that if the contractivity factors were known for the region I_{ij}, the other parameters could be easily determined.

3.1 Geometric calculation of the contractivity factors

From what was mentioned previously, we see that the vertices $(\hat{x}_{k-1}, \hat{y}_{l-1})$, $(\hat{x}_{k-1}, \hat{y}_l)$, $(\hat{x}_k, \hat{y}_{l-1})$, $(\hat{x}_k, \hat{y}_l)$ of the domain are mapped to the vertices of the region (x_{i-1}, y_{j-1}), (x_{i-1}, y_j), (x_i, y_{j-1}), (x_i, y_j) and the part of the function (image) that is "contained" in the domain J_{kl} is contracted vertically by the unknown contractivity factor s_{ij}. Choose $y = \hat{y}_{l-1}$. Let $|\mu_0^y|$ be the mean absolute vertical distance between any function (image) value $f(x, y)$ (with $x = \hat{x}_{k-1}, \hat{x}_{k-1} + 1, \ldots, \hat{x}_{k-1} + \delta = \hat{x}_k$) and the straight line between the vertices $(\hat{x}_{k-1}, \hat{y}_{l-1})$ and $(\hat{x}_k, \hat{y}_{l-1})$. The value of the vertical distance is taken as positive, if $f(x, y)$ is above the straight line and negative otherwise. Now choose $y = \hat{y}_{l-1} + 1$ and compute $|\mu_1^y|$ similarly. In general, choose $y = \hat{y}_{l-1} + r$ (for $r = 0, 1, \ldots, \delta$) and let $|\mu_r^y|$ be the mean absolute vertical distance between any value $f(x, y)$ (with $x = \hat{x}_{k-1}, \hat{x}_{k-1} + 1, \ldots, \hat{x}_{k-1} + \delta = \hat{x}_k$) and the straight line between the vertices $(\hat{x}_{k-1}, y)$ and $(\hat{x}_k, y)$. Similarly choose $x = \hat{x}_{k-1} + r$ (for $r = 0, 1, \ldots, \delta$) and let $|\mu_r^x|$ be the mean absolute vertical distance between any function (image) value $f(x, y)$ (with $y = \hat{y}_{l-1}, \hat{y}_{l-1} + 1, \ldots, \hat{y}_{l-1} + \delta = \hat{y}_l$) and the straight line between the endpoints $(x, \hat{y}_{l-1})$ and $(x, \hat{y}_l)$. The sign of μ_r^x is taken as described above. Then $\mu = mean\{\mu_r^x, \mu_r^y, r = 0, 1, \ldots, \delta\}$. We compute ν similarly by using the values of the function lying inside region I_{ij}. Thus the contractivity factor is computed by $s_{ij} = \nu/\mu$.

3.2 The inverse algorithm

We now present an iterative algorithm for finding the *model parameters*, i.e. the interpolation points $(x_i, y_i, z_{i,j})$, the contractivity factors and the addresses (i.e., the best-matched domains) associated with each region of the function. Suppose we are dealing with a $N_1 \times N_2$ pixel image in which each pixel can be one of 256 levels of grey. We choose δ and Δ *a priori* and form the set of interpolation points S and the set of address points Q. The number of interpolation regions, NM, is greater than the number of distinct address domains, say M_1. According to the existing theory, we need to store the model parameters. For each emerging region we seek for the "best-mapped" domain with respect to a metric h. If a "good" match is found we store the respective contractivity factor and its address (i.e. the number of the "best-mapped" domain), otherwise we partition the region into a^2 smaller regions of side-length δ/a, store the new interpolation points (the vertices of the smaller regions) and repeat the procedure for each new region. The following algorithm describes the procedure in details.

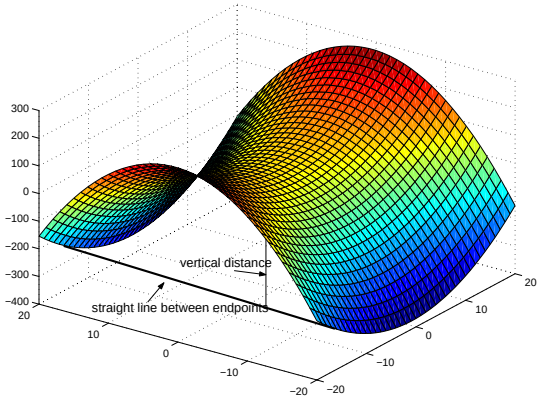


Figure 3: Geometric calculation of the contractivity factors.

1) Choose values for δ and Δ, such that $\Delta = a\delta$. Choose, also, error tolerance ε and a maximum depth $d_{\max}$.

2) Create two queues, one named *squeue* and put all the regions inside as well as a queue named *iqueue* and put all the initial interpolation points inside. In addition create two empty queues named *cqueue* and *aqueue* (we store the contractivity factors in the first and the addresses in the latter). Set the depth $d = 1$ and create an empty queue named *squeue2*.

3) **While** *squeue* is not empty **do**

   (a) Get one region from *squeue*.

   (b) **For** domain $j = 1 : M_1$ **do**

      (i) Compute the contractivity factor for the map associated with the j-th domain and the region.

      (ii) **If** $|s| \geq 1$, **then** go to (v). Otherwise check the condition of continuity. If it doesn't hold, goto (v).

      (iii) Compute the other parameters and map the points of the j-th domain (say D_j) through w according to the *mapping algorithm*. Say $w(D_j)$ the emerging set.

      (iv) Compute (with a proper distance measure) the distance h_{ij} between $w(D_j)$ and the points of region i.

      (v) **Next** j

   (c) Find the j for which h_{ij} is a minimum.

   (d) **If** $h_{ij} > \varepsilon$ and $d < d_{\max}$
   then create four new regions, add them to *squeue2*, add the vertices of each new region to *iqueue* (as additional interpolation points) and add 0 to *aqueue*
   Else
   store j with the minimum distance inside *aqueue* and s inside *cqueue*.

4) **If** *squeue2* is not empty, then set $squeue = squeue2$, $d = d + 1$ and go to (3).

5) Store $d_{\max}$, δ, Δ, *cqueue*, *iqueue* and *aqueue*.

6) End.

The mapping algorithm We map the domain D_j to the region R.

1. Put the endpoints of the region R to the new set $w(D_j)$.
2. Compute the other parameters. Let $a = \delta/\Delta$, where δ, Δ are the side of the region and the corresponding domain, respectively.
3. Map the points of the first and last row as follows. The first and the last point of these rows of D_j are interpolation points and they have been already mapped. Map the $(a+1)$-th point of the domain to the 2nd point of the $w(D_j)$. Continue by mapping the $(2a+1), (3a+1), \ldots$ point of the domain to the 3rd, 4th, ... point of the $w(D_j)$.
4. For all other rows: Map the $(a+1), (2a+1), \ldots$ point of the domain to the 1st, 2nd, ... point of the $w(D_j)$.

3.3 A decompression algorithm

To reconstruct the original image, we may use the Deterministic Iteration Algorithm (DIA) as in [Barnsley, 1993]. The DIA, though, has a minor drawback. When mapping the points of a domain to the corresponding region, it is possible that some points of the region will not be mapped to a specific point of the region. For example, the pixel $(100, 150)$ may be mapped to the point $(34.33, 12.46)$, which is not a specific pixel. It is possible that most of the pixels will be mapped in this way. So, in order to reconstruct the original pixels, we must use those points occurring from the mapping of each region that are close to the corresponding pixel. In the previous example, the point $(34.33, 12.46)$ (among others) must be used for the reconstruction of the pixels $(34, 12), (35, 12), (34, 13)$ and $(35, 13)$. Of course, this will increase the amount of calculations and it is likely that if the number of iterations is not adequate, points near a specific pixel won't be produced, leading to more and more iterations.

In order to confront these problems we introduce a variation of the DIA for the case at which $\delta = a^r$ and $\Delta = a^{r+1}$. In this case, the algorithm needs exactly $r+1$ iterations to reconstruct the original image without computing unnecessary points. This means that the algorithm computes the points with integer coordinates (that are actual pixels), thus not facing the same problems. Figs. 4 and 5 show how one image is divided into regions and domains. We assume that the image has dimensions 9×9, $\delta = 4$ and $\delta = 8$ resulting in one domain and four regions. The number in each pixel states the iteration in which every pixel is computed. We have chosen $\delta = 8 = 2^3$ and $\Delta = 16 = 2^4$, thus $2 + 1 = 3$ iterations for the complete reconstruction are needed.

The decompression algorithm

1. Create the queues $AD[i], CON[i], IP[i]$, $i = 1, 2, \ldots, d_{\max}$, which contain the adresses, contractivity factors and interpolation points of the regions of side $\delta/a^{(i-1)}$. All these numbers were stored in *aqueue*, *cqueue* and *iqueue*, respectively.
2. Put the interpolation points inside the picture.
3. Compute $steps = trunc\left(\frac{\log(\delta)}{\log(a)}\right)$
4. **for** $t = 1 :$ *steps* **do**

   for $i = 1 : min\{steps, d_{\max}\}$

   (a) For every region of side $\delta/a^{(i-1)}$ **do**

   (i) Find the corresponding domain (j) (from $AD[i]$).

   (ii) **If** $j \neq 0$ **then**

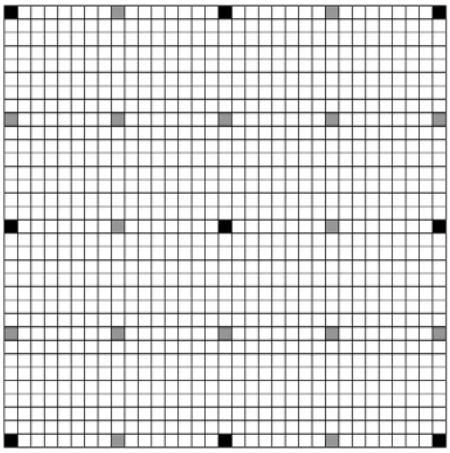


Figure 4: The grid of the pixels and not their colors is shown by choosing $\delta = 8$ and $\Delta = 16$, thus emerging 16 regions and 4 domains. The endpoints of the domains are painted black. Every colored pixel is an interpolation point.

A. Find the corresponding contractivity factor (from $CON[i]$) and compute the rest of the map parameters.

B. Do the following **until** the last row of the domain is reached ($a^{(t-1)}$ times). *Remember that the domain (as a part of the picture) is initially empty and gradually fills iteration after iteration.* At this stage only pixels of distance (horizontal or vertical) $\delta/a^{(t-1)}$ are already mapped and, therefore, capable of been mapped again. So, to go from one line to the "next" you must skip $\delta/a^{(t-1)}$ lines.

- For the first row do: Skip the first point (it has been mapped already). Map the "next" $a-1$ points of the domain, skip the "next" point map the "next" $a-1$ points etc. Do this **until** the right domain endpoint is reached ($a^{(t-1)}$ times). *Remember that these $a-1$ points are not consecutive! To go from one point to the "next" you must skip $\delta/a^{(t-1)}$ points.*
- Map the next $a-1$ lines as follows: At each line map a^t+1 points. *Remember that these points are not consecutive! To go from one point to the "next" you must skip $\delta/a^{(t-1)}$ points.*

(iii) For the last line do: Skip the first point (interpolation point). Map the "next" $a-1$ points of the domain, skip the "next" point map the "next" $a-1$ points etc. Do this until the right domain endpoint is reached ($a^{(t-1)}$ times). Remember that these $a-1$ points are not consecutive! To go from one point to the "next" you must skip $\delta/a^{(t-1)}$ points.

4 Comparative Results and Conclusions

As mentioned above, after the application of the algorithm on an image we need to store some interpolation points (integers), some addresses (also integers) and some contractivity factors (floating points). In order to store the latter we must first quantise them. We use the uniform quantiser. In addition, we may apply some sort of lossless compression to the interpolation points and the addresses to increase the compression ratio even further. In the next examples we used the entropy coding compression and 6 bits for the quantisation of the contractivity factors (smaller values decrease signficantly the quality of the reconstructed image).

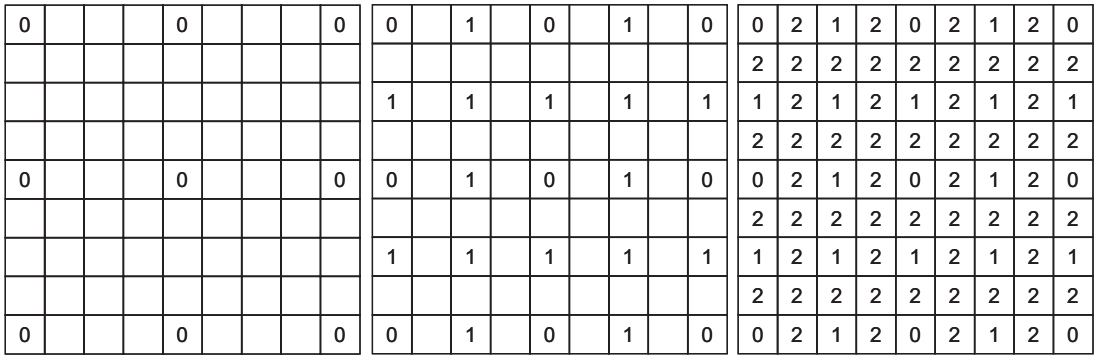


Figure 5: The decompression (reconstruction) of an image region is shown. (a) Initially we put the interpolation points. (b) At the first iteration we map the interpolation points of the domain to each region. Thus, each region has 5 new points. (c) At the second iteration we map the existing points (i.e, the interpolation points and the points that emerged from iteration 1) of the domain to each region.

In order to compute the distances described in the algorithms, we used the ρ_2 (Euclidean) distance measure. The original image used as our reference point in the experiments presented here is the $2049 \times 2049 \times 8$ bpp image of Lena shown in Fig. 4(a). Finally, for the calculation of the contractivity factors we used the geometric approach described in Subsection 3.1.

As we described previously, the method we introduced here partitions one image into regions and domains, so it is necessary that the dimensions of the image must be multiples of Δ. Thus, one must realise that we can only model images that fulfill the previous criteria. For the rest, we must make some minor modifications to the algorithms (i.e., add some "dummy pixels"). We applied the two methods to the image of Lena. Fig. 4 shows PSNR versus compression results for 2049×2049 Lena using (a) some fractal-based and (b) some fractal-based and some non-fractal-based methods. "PSA2D" is the 2D piecewise self-affine model, "Bivariate" is the 2D piecewise bivariate model we introduce, while "Barnsley" stands for the method devoloped by Iterated Systems and described in length by Ning Lu in [Lu, 1997]. To measure the time each method needed to compress this image we used a Pentium 4 PC with a 2.66 GHz CPU clock running Windows XP. The method was implemented by using C++ and we created an executable that can compress any image. The results are shown in Table 1 and Figs. 6-8.

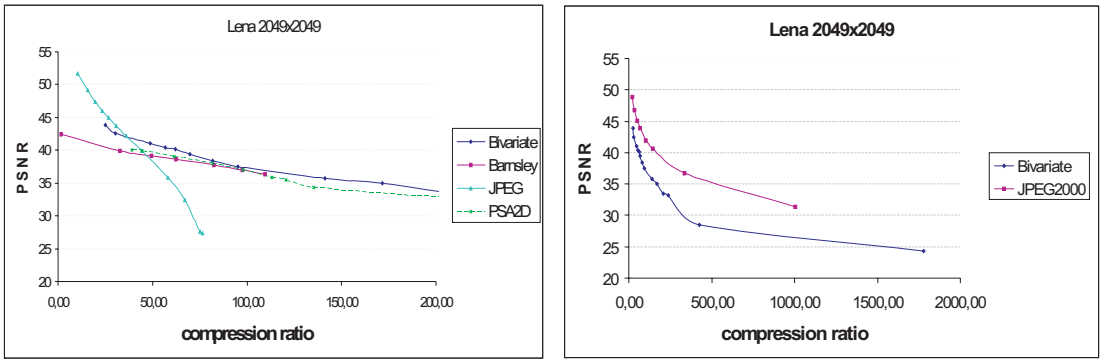
The 2-dimensional fractal interpolation method described here achieve compression ratios comparable to that of Barnsley's and to JPEG. Its major drawback is that it is less efective at the edges of the image as one can observe by closely looking at the above images. Choosing a smaller value for δ eliminates this drawback, the PSNR value approaches the PSNR value of the JPEG format, but the compression ratio is decreased significantly. In general the bivariate model acts better than JPEG, but not as good as JPEG2000 or SPIHT (another wavelet-based approach in image compression). It is interesting, though, to see that the model we propose outperforms the "best" previous fractal method developed by Iterated Systems Inc. This happens because the latter uses affine maps instead of bivariate ones.

(a) (b)

Figure 6: (a) The original image of Lena. (b) The test image after compression with the bivariate model for $\delta = 64$, $\Delta = 128$, $\varepsilon = 4$, $d_{max} = 4$.

(a) (b)

Figure 7: The test image compressed (a) by a factor of 56.60 with 40.37dB PSNR and $\delta = 32$, $\Delta = 64$, $\varepsilon = 2.4$, $d_{max} = 3$ (b) by a factor of 141 with 45.74dB PSNR and $\delta = 64$, $\Delta = 128$, $\varepsilon = 2.6$, $d_{max} = 3$ using the bivariate model.



(a) (b)

Figure 8: Comparison of results for 2049×2049 image of Lena.

Parameters $(\delta, \Delta, d_{\max}, \varepsilon)$	Enc. time (sec)	PSNR (dB)	comp. ratio
$32, 64, 3, 0.7$	$\approx$ 2 min	43.22	25:1
$64, 128, 4, 0.6$	$\approx$ 2 min	42.52	30.23:1
$64, 128, 4, 1.6$	$\approx$ 1 min	41.03	48.85:1
$32, 64, 3, 2.4$	$\approx$ 1 min	40.37	56.6:1
$64, 128, 4, 2.2$	$\approx$ 25 secs	40.01	62:1
$64, 128, 4, 2.6$	$\approx$ 25 secs	39.44	69.72:1
$32, 64, 3, 3.2$	$\approx$ 20 secs	38.38	81.91:1
$64, 128, 4, 4$	$\approx$ 20 secs	37.46	95.08:1
$64, 128, 3, 2.6$	$\approx$ 18 secs	35.74	141.1:1
$64, 128, 3, 4$	$\approx$ 15 secs	35.01	171.76:1
$64, 128, 2, 1.7$	$\approx$ 18 secs	33.39	210.31:1
$64, 128, 2, 3.4$	$\approx$ 18 secs	33.18	238.61:1
$32, 64, 1, 4$	$\approx$ 25 secs	28.53	426:1
$64, 128, 1, 4$	$\approx$ 25 secs	24.37	1776:1

Table 1: Comparison of the application of the **Bivariate** model on the image of Lena (2049×2049) using various parameters.

References

[1] Barnsley, M. F. [1993] *Fractals everywhere* 2nd ed. (Academic Press Professional, San Diego CA).

[2] Barnsley, M. F. & Hurd, L. P. [1993] *Fractal image compression* (AK Peters, Wellesley MA).

[3] Bouboulis, P. & Dalla, Leoni & Drakopoulos, V. [2004] "On the box counting dimension of the recurrent bivariate fractal interpolation surfaces," (submitted).

[4] Dalla, Leoni [2002] "Bivariate Fractal interpolation functions on grids," *Fractals* **10**(1), 53–58.

[5] Drakopoulos, V. & Bouboulis, P. & Theodoridis S. [2004] "Image compression using affine fractal surfaces on rectangular lattices," *IEE Proc. Vision, Image & Signal Processing*, submitted.

[6] Fisher, Y. [1995] *Fractal image compression: Theory and application* (Springer-Verlag, New York).

[7] Jacquin, A. E. [1992] "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Processing* **1**(1), 18–30.

[8] Lu , N. [1997] *Fractal imaging* (Academic Press, San Diego CA).

[9] Price, J. R. & Hayes III, M. H. [1998] "Fractal interpolation of images and volumes," *Proc. of the 32nd Asilomar Conference on Signals, Systems, and Computers* **2**, 1698–1702.

[10] Wohlberg, B. & de Jager, G. [1999] "A review of the fractal image coding literature," *IEEE Trans. Image Processing* **8**(12), 1716–1729.

[11] Xie, H. & Sun, H. [1997] "The study of bivariate fractal interpolation functions and creation of fractal interpolation surfaces," *Fractals* **5**(4), 625–634.