Stochastics and Dynamics, Vol. 2, No. 2 (2002) 161–173

A PROBABILISTIC POWER DOMAIN ALGORITHM FOR FRACTAL IMAGE DECODING

V. DRAKOPOULOS

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

A. KAKOS and N. NIKOLAOU

Division of Information Systems, Bank of Greece, 341 Mesogeion Ave., 152 32, Cholargos, Hellas

Received 21 January 2002
Revised 11 April 2002

A new algorithm, called herein the random power domain algorithm, is discussed; it generates the image corresponding to an iterated function system with probabilities, a technique used in fractal image decoding. A simple complexity analysis for the algorithm is also derived.

Keywords: Fractals; random algorithm; image comparison; iterated function systems; power domain.

AMS Subject Classification: 28A80, 94A08, 68Q10, 68Q55

1. Introduction

A number of algorithms have been proposed for rendering the invariant measure supported by the attractor of an (hyperbolic) *iterated function system*, or *IFS* for short, on the plane. In what follows, however, a competitive alternative will be described and implemented as a consequence of the introduction of domain theory in dynamical systems, measures and fractals [2]. It uses the probabilistic power domain as its computational model and generates the invariant measure of an IFS with probabilities.

The proposed algorithm, after comparing with the most commonly used probabilistic algorithms for rendering the invariant measure supported by the attractor, namely the Random Iteration Algorithm (RIA, see [1]) and the Random Adaptive Cut Algorithm (RACA, see [4]), shows to be, under all known circumstances, substantially faster than both the others and much more efficient than the RIA in terms of memory requirements. Since the RIA chooses the points of the attractor at random, it will unavoidably redraw some points more than once. Figure 1 shows a demography of the dendrite shown in Fig. 7(e). The dendrite is put on a

(a) (b) (c)

Fig. 1. Distribution of an orbit rendering a dendrite displayed in vertical bars using (a) the RIA, (b) the RACA and (c) the proposed algorithm.

square lattice on $\mathbb{R}^2$ with small mesh size; the number of points from the dendrite in each little box of that lattice is measured and represented by a vertical bar, thus visualizing the distribution. Apparently, the border of the dendrite is visited most frequently, while the other points seem to be avoided most often. This substantial waste of time is the main reason why the proposed algorithm, which produces no more than the necessary points, is that faster. Furthermore, many of the points of the attractor will never be plotted, due to the randomness of the RIA; to make matters worse, the attractors produced by two successive runnings of the algorithm will not be exactly the same.

The RIA theoretically suggests that its corresponding iterated procedure should be performed for an infinite number of times before the actual attractor is produced. In the words of Monro and Dudbridge in [5] "There is no definite stopping criterion for the RIA". On the contrary, the RACA uses a criterion to terminate the recursive descent of the *tree of transformations* and the idea behind it can be served as a basis for a magnification algorithm. The main advantage of our algorithm is that it encapsulates an economical and enhanced stopping criterion as opposed to the RACA; roughly speaking, it terminates (in finite time) automatically and quickly on any digitized screen without needing to fix a number of iterations in advance. Moreover, for a given discrimination capability of the computer screen, the proposed algorithm has a determined upper and lower bound for the number of computations required before the best possible attractor for the given resolution is constructed. The exact number of computations, however, cannot be specified analytically in a closed formula, but can be very easily computed with the aid of a computer.

An analytic description of the algorithm as well as a digest of the theoretical fundamentals, on which its model is based, follows. However, an extended abstract for the theory can be found in [3], where power domains are discussed along with IFS. A general reference oriented toward computational theory is [6].

2. The Probabilistic Power Domain

First of all a link between measure theory and domain theory is needed. The link is established in [2]; however, for a deep understanding of the algorithm and its implementation that follows, we shall make an account of the most fundamental conclusions that appear in that paper.

We are especially interested in Borel measures, which have the significant property that the measure of any Borel set is determined by the topologically important

open sets or, alternatively, the compact sets; such a Borel measure is called regular. More formally, a Borel measure μ on a locally compact Hausdorff space is called *regular*, if, for all Borel subsets B of X, we have

$$\mu(B) = \inf\{\mu(O) \mid B \subseteq O, O \text{ open}\} = \sup\{\mu(F) \mid B \supseteq F, F \text{ compact}\}.$$

It can be proved that, if X is a locally compact, second countable Hausdorff space as in our case, every open set will be *σ-compact* (it is a countable union of compact sets) and every Borel measure is regular.

The computational model of the new algorithm for the construction of the invariant measure of an IFS with probabilities is based on the probabilistic power domain. Hence the following definitions are essential for the understanding of the proposed algorithm. The lattice of open sets of a topological space X is denoted by $\Omega(X)$.

Definition 1. An *(e)valuation* on a topological space X is a map $\nu\colon \Omega(X) \to [0, \infty)$ which satisfies:

(i) $\nu(U) + \nu(V) = \nu(U \cup V) + \nu(U \cap V)$
(ii) $\nu(\emptyset) = 0$, and
(iii) $U \subseteq V \Rightarrow \nu(U) \leq \nu(V)$.

A *continuous (e)valuation* is an (e)valuation such that whenever $A \subseteq \Omega(X)$ is a directed set (with respect to $\subseteq$) of open sets of X, then

$$\nu\left(\bigcup_{O \in A} O\right) = \sup_{O \in A} \nu(O).$$

Definition 2. The *Probabilistic Power Domain* $\mathcal{P}X$ of a topological space X consists of the set of continuous (e)valuations ν on X with $\nu(X) \leq 1$ and is ordered as follows:

$$\mu \sqsubseteq \nu \text{ if and only if for all open sets } O \text{ of } X, \mu(O) \leq \nu(O).$$

Definition 3. For any $b \in X$, the *point (e)valuation* based at b is the (e)valuation $n_b\colon \Omega(X) \to [0, \infty)$ defined by

$$n_b(O) = \begin{cases} 1, & \text{if } b \in O, \\ 0, & \text{otherwise.} \end{cases}$$

Any directed set $(\mu_i)_{i \in I}$ of (e)valuations has a *least upper bound* (l.u.b.) given by $\bigsqcup_i \mu_i = \mu$, where for $O \in \Omega(X)$ we have $\mu(O) = \sup_{i \in I} \mu_i(O)$; hence $(\mathcal{P}X, \sqsubseteq)$ is a *directed complete partial order* (d.c.p.o.), with bottom element (which is the $\perp \in \mathcal{P}X : \perp(O) = 0, O \in \Omega(X)$) It is easy to prove that any finite linear combination $\sum_{i=1}^{n} r_i n_{b_i}$ of unit point (e)valuations n_{b_i} with constant coefficients $r_i \in [0, \infty)$, $i = 1, 2, \ldots, n$ is also an (e)valuation on X.

What is really interesting about (e)valuations is that, if X is an ω-continuous d.c.p.o., then $\mathcal{P}X$ is an ω-continuous d.c.p.o. as well; moreover any (e)valuation

on an ω-continuous d.c.p.o. is the directed l.u.b. of linear combinations of point (e)valuations and it extends uniquely to a measure. Hence (e)valuations and measures are in fact the same on ω-continuous d.c.p.o's.

The set of all positive Borel measures μ on a locally compact, second countable Hausdorff space X with $\mu(X) \leq 1$, denoted by $M(X)$, can be partially ordered by putting

$$\mu \sqsubseteq \nu \text{ if and only if for all Borel sets } B \subseteq X, \mu(B) \leq \nu(B).$$

With this ordering $M(X)$ becomes a d.c.p.o.

3. A Computational Model

For any Hausdorff metric space X the *upper space* $\mathcal{U}X$ consists of all nonempty compact subsets of X (see Fig.2), that is

$$\mathcal{U}X = \{\emptyset \neq C \subseteq X \mid C \text{ compact}\}.$$

This space has a topology, called the *upper topology*, whose base is the collection

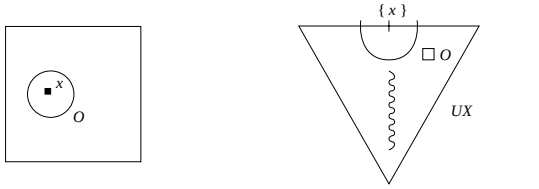
$$\Box_a = \{C \in \mathcal{U}X \mid C \subseteq a\},$$

where $a \in \Omega(X)$ is an open set of X. This means that, for any open subset a of X, the collection of all compact nonempty subsets of X that are included in a forms an element of the base for the upper topology. This topology is T_0; the specialisation ordering $\sqsubseteq_u$ of $\mathcal{U}X$ is the superset inclusion, i.e.

$$A \sqsubseteq_u B \Longleftrightarrow \forall a \in \Omega(X)[A \subseteq a \Rightarrow B \subseteq a] \Longleftrightarrow A \supseteq B.$$

Under this ordering ($\sqsubseteq_u$) the space $(\mathcal{U}X, \supseteq)$ becomes a d.c.p.o., which means that every directed set has a l.u.b. The l.u.b. of a directed set of compact subsets is their intersection; the elements of X are maximal elements of $\mathcal{U}X$. The singleton map $s\colon X \to \mathcal{U}X$ with $s(x) = \{x\}$ can be used for embedding X onto the set of maximal elements of $\mathcal{U}X$.

If $\mathcal{U}X$ is ω-continuous, then we can identify (e)valuations and measures on $\mathcal{U}X$, for a continuous (e)valuation μ on $\mathcal{U}X$ extends uniquely to a Borel measure on $\mathcal{U}X$ and a measure on $\mathcal{U}X$ is a continuous (e)valuation. The support of the invariant measure of an IFS has the very important property that it contains exactly the



$X = [0,1] \times [0,1]$ $X = [0,1] \times [0,1]$

Fig. 2. The upper space for the unit square.

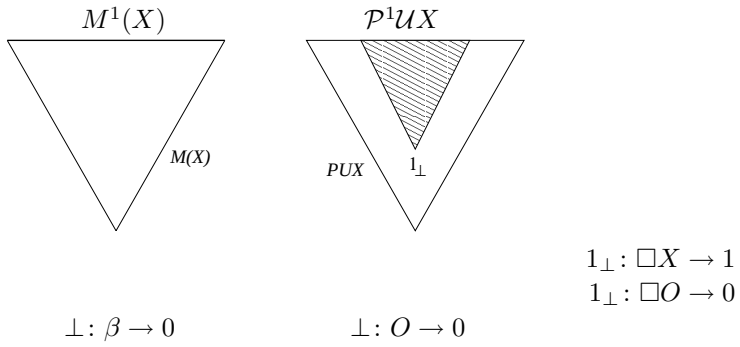


Fig. 3. The probabilistic power domain.

points of the attractor. We will now extend the notion of the support in the case of (e)valuations.

An (e)valuation $\mu \in \mathcal{PU}X$ is said to be supported in $s(X)$, if $\mu(\mathcal{U}X \setminus s(X)) = 0$; if μ is supported in $s(X)$, then the support of μ is the set of points $y \in s(X)$ such that $\mu(N) > 0$ for all neighbourhoods $N \subseteq \mathcal{U}X$ of y. We denote by $S(X)$ the set that contains all the supported (e)valuations in $s(X)$ ($S(X) \subseteq \mathcal{PU}X$).

Since the invariant measure of an IFS is a normalized Borel measure, we restrict our attention to normalized measures and normalized (e)valuations. We denote by $M^1(X), \mathcal{P}^1\mathcal{U}X, S^1(X)$ the sets which contain the normalized Borel measures, the normalized (e)valuations and the normalized supported (e)valuations, respectively (see Fig. 3). $\mathcal{PU}X$ and $\mathcal{P}^1\mathcal{U}X$ have exactly the same maximal elements and the elements of $S^1(X)$ are maximal elements of $\mathcal{PU}X$.

The maps $e\colon M^1(X) \to S^1(X)$ with $e(\mu)(O) \mapsto \mu(s^{-1}(O))$ for any $O \in \Omega(X)$ and $j\colon S^1(X) \to M^1(X)$ with $j(\nu)(B) \mapsto \nu(s(B))$ for any Borel subset B of X are well defined and continuous. Since j is the inverse of e, the restrictions of e and j give an isomorphism between $M^1(X)$ and $S^1(X)$.

Let us recall that $\mathcal{PU}X$ is ω-continuous whenever $\mathcal{U}X$ is ω-continuous and in this case any (e)valuation $\mu \in \mathcal{PU}X$ is the directed l.u.b. of linear combinations of point (e)valuations. If $B \subseteq \mathcal{U}X$ is a countable order basis of $\mathcal{U}X$, then

$$\left\{ \sum_{i=1}^{n} r_i n_{b_i} \mid b_i \in B, r_i \text{ rational and } \sum_{i=1}^{n} r_i \leq 1 \right\}$$

is a countable order basis of $\mathcal{PU}X$. This observation provides us with an effective way of obtaining measures on X as the directed l.u.b. of linear combinations of point (e)valuations on $\mathcal{U}X$.

If X is compact, then both $\mathcal{U}X$ and $\mathcal{P}^1\mathcal{U}X$ have bottom elements; the bottom element of $\mathcal{U}X$ is the space X itself, whereas the bottom element of $\mathcal{P}^1\mathcal{U}X$ is the unit point (e)valuation n_X based at $X \in \mathcal{U}X$ with

$$n_X(O) = \begin{cases} 1, & \text{if } O \in \mathcal{U}X, \\ 0, & \text{otherwise.} \end{cases}$$

The above theoretical framework can be used for the construction of the invariant measure of an IFS with probabilities as follows.

4. The Random Power Domain Algorithm

Let $\{X; f_1, f_2, \ldots, f_N; p_1, p_2, \ldots, p_N\}$ or, more briefly, $\{X; f_{1-N}; p_{1-N}\}$ be an hyperbolic IFS with probabilities; the operator $T\colon \mathcal{P}^1\mathcal{U}X \to \mathcal{P}^1\mathcal{U}X$ given by

$$T(\mu)(O) = \sum_{i=1}^{N} p_i \mu(f_i^{-1}(O))$$

is well defined and continuous, and therefore has a least fixed point; it can be proved (see [2]) that the least fixed point of T is supported in X and its support is the unique attractor of the IFS. Since an element of $S^1(X)$ is a maximal element of $\mathcal{P}^1\mathcal{U}X$, we immediately deduce that an hyperbolic IFS with probabilities has a unique fixed point. If μ is the unique fixed point of T, then $j(\mu) \in M^1(X)$ is the unique invariant measure of the IFS.

We correspond the (e)valuation $n_{f_{i_1} f_{i_2} \cdots f_{i_n} X}$ to node $f_{i_1} f_{i_2} \cdots f_{i_n} X$ of the non-probabilistic tree shown in Fig. 4. The nodes of the tree are point (e)valuations ordered as usual so that the construction of the invariant measure can be obtained by using the tree shown in Fig. 5. Then, the tree is traversed until the outcome of any node $f_{i_1} f_{i_2} \cdots f_{i_n} X$ becomes smaller than a pixel on the computer screen. Then a pruning occurs and the corresponding branch will produce no more children. This tree, however, cannot be employed in any efficient implementation. The solution will be the use of an equivalent tree (see Fig. 6) that can be traversed efficiently. We prove that this alternative tree is computationally equivalent to the original one.

The two trees have exactly the same first two levels. From the third level onwards, however, the differences start. Even if the second level is identical in both trees, a different third level is produced. It is our intention to prove that this does not affect the theoretical basis and that the third level is computationally equivalent. If at some node of the second level $s_{i_1} s_{i_2} \sqrt{2} < \varepsilon$ then holds, due to

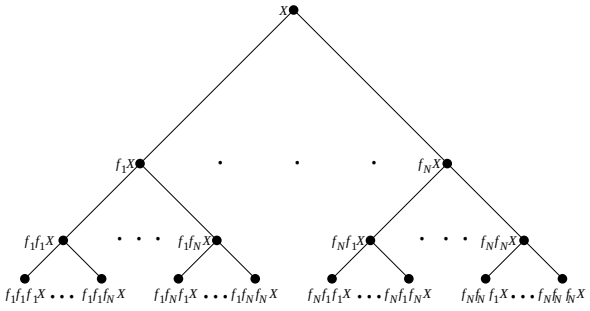


Fig. 4. The IFS tree.

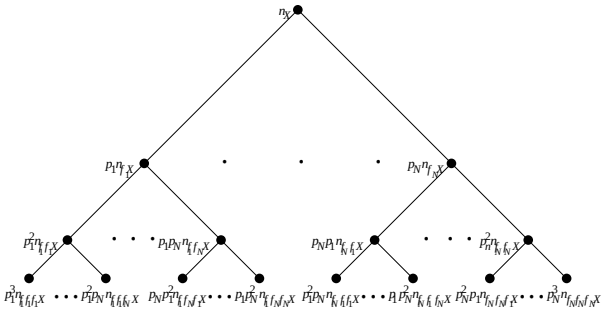


Fig. 5. The IFS tree with probabilities.

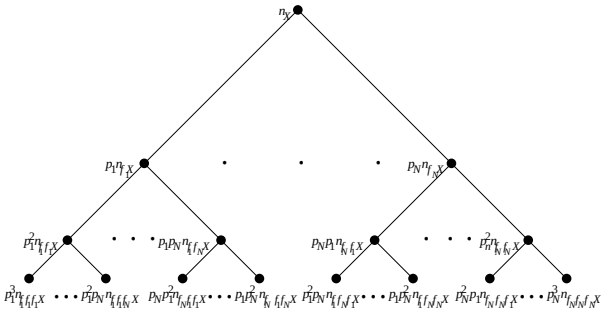


Fig. 6. The action tree with probabilities.

the commutative property of the real numbers, $s_{i_2} s_{i_1} \sqrt{2} < \varepsilon$ also holds. Hence, in the case of the first tree, $s_{i_1} s_{i_2} \sqrt{2} < \varepsilon$ means that the children of $f_{i_1} f_{i_2} X$ $(f_{i_1} f_{i_2} f_1 X, f_{i_1} f_{i_2} f_2 X, \ldots, f_{i_1} f_{i_2} f_N X)$ will not be produced and the pixel $f_{i_1} f_{i_2} X$ will be plotted. For this same tree, $s_{i_2} s_{i_1} \sqrt{2} < \varepsilon$ means that the children of $f_{i_2} f_{i_1} X$ $(f_{i_2} f_{i_1} f_1 X, f_{i_2} f_{i_1} f_2 X, \ldots, f_{i_2} f_{i_1} f_N X)$ will not be produced and the pixel $f_{i_2} f_{i_1} X$ will be plotted. On the other hand, for the second tree, the condition $s_{i_1} s_{i_2} \sqrt{2} < \varepsilon$ means that the children of $f_{i_1} f_{i_2} X$ $(f_{i_2} f_{i_1} f_1 X, f_{i_2} f_{i_1} f_2 X, \ldots, f_{i_2} f_{i_1} f_N X)$ will not be produced and the pixel $f_{i_1} f_{i_2} X$ will be plotted. Moreover, the condition $s_{i_2} s_{i_1} \sqrt{2} < \varepsilon$, for the second tree, means that the children of $f_{i_2} f_{i_1} X$ $(f_{i_1} f_{i_2} f_1 X, f_{i_1} f_{i_2} f_2 X, \ldots, f_{i_1} f_{i_2} f_N X)$ will not be produced and the pixel $f_{i_2} f_{i_1} X$ will be plotted. It is now more than obvious that the third level of the two trees are computationally equivalent. Using induction we can prove that the two trees are equivalent at all levels. Thus the two trees are, for our purposes, computationally equivalent.

Our contribution, i.e. the discovery of the equivalent tree, has beneficially effected the implementation of the algorithm. Not a single redundant computation is performed. On the other hand, the memory requirements have significantly dropped due to the introduction of this second tree.

If a branch is pruned, for example at node $f_{i_1} f_{i_2} \cdots f_{i_n} X$, then the corresponding atomic measure for the pixel of the attractor will be $p_{i_1} p_{i_2} \cdots p_{i_n}$, which is assigned by the corresponding point (e)valuation $p_{i_1} p_{i_2} \cdots p_{i_n} n_{f_{i_1} f_{i_2} \cdots f_{i_n} X}$. However, two pruned branches might correspond to the same pixel. In this case, the "measure" contribution of all branches that end up at the same pixel should be summed up before any rendering of the pixel takes places. In this way, a record of which branches correspond to the same pixel must be kept so that one is able to add up all individual contributions to determine the actual atomic measure of every screen pixel of the attractor. In this way a graphic representation of the invariant measure is achieved.

Although the theory suggests that at depth n, $T^{(n)}\mu_0$ is an (e)valuation, μ_0 being any normalized (e)valuation, the unbalanced pruning of the tree might raise some objections regarding the soundness of our implementation. However, if d_{max} is the level at which the last pruning takes place, then we can safely assume that our approximation to the invariant measure μ is $T^{(d_{max})}\mu_0$, μ_0 being the uniform measure on the screen. This means that we can make the equivalent theoretic assumption that all branches of the point (e)valuation tree of Figs. 5 and 6 are pruned at level d_{max}. The following argument justifies this assumption.

For any branch of the tree which is pruned at an earlier stage d than d_{max} we can "insert" its children and grandchildren – effectively pruned – in the tree diagram until the complete level d_{max} is reached. Then our approximation to the invariant measure is still a sound one for, if at level d we had decided that the parent node represented less than a pixel on the screen, then its children would still be included in that pixel (due to the ordering of nodes); then their total measure contribution to the pixel would be equal to that of their parent at level d. This means that the sum of the atomic measures that all children have in total would be equal to what the node at level d had. Thus, this balanced pruning – which we just proved to be equivalent to the unbalanced one we perform – justifies our argument that what we actual produce is a measure, in fact $T^{(d_{max})}\mu_0$.

5. The Algorithm and Its Complexity

Hence the algorithm can be described roughly as follows; we start from the fixed point which corresponds to the transformation with the greatest probability and traverse the action tree until a parallelogram $f_{i_1} f_{i_2} \cdots f_{i_N} X$ becomes small enough so that it is actually a pixel on the computer screen. This happens when $s_{i_1} s_{i_2} \cdots s_{i_N} \sqrt{2} < \varepsilon$, $\varepsilon = 1/M$, where M is the resolution of the screen, and in that case this branch has contributed $p_{i_1} p_{i_2} \cdots p_{i_N}$ to the measure of the pixel $f_{i_1} f_{i_2} \cdots f_{i_N} X$. After having sum up all measure contributions for each pixel, we color the attractor pixels accordingly. In that way, a graphic representation of the invariant measure is produced.

The actual algorithm in a form of pseudo-code, which also provides a definition for the equivalent tree of Fig. 6, has as follows:

0. Start.
1. Compute all contractivity factors s_i.

2a. Assign measure 1 on the whole screen, i.e. $\mu([0,1]^2) = 1$.

2b. All atomic measures $\mu(x)$ of pixels $x \in [0,1]^2$ are initially set to zero.

3a. Call *procedure measure* $([x_0, x_0, \ldots, x_0], 0, [\sqrt{2}, \sqrt{2}, \ldots, \sqrt{2}], [1, 1, \ldots, 1])$, for $x_0 \in [0,1]^2$.

3b. Plot all pixels $x \in [0,1]^2$ with a color proportional to their atomic measure $\mu(x)$.

4. End.

where

```
procedure measure([x_1, x_2, ..., x_N], q, [d_1, d_2, ..., d_N], [m_1, m_2, ..., m_N]) {
  for i = 1, ..., N do {
    x'_i = f_i(x_q)
    d'_i = s_i * d_q
    m'_i = p_i * m_q
  }
  for i = 1, ..., N do {
    if (d'_i > 1/M) then do
      call measure([x'_1, x'_2, ..., x'_N], i, [d'_1, d'_2, ..., d'_N], [m'_1, m'_2, ..., m'_N])
    else do
         μ(x'_i) = μ(x'_i) + m'_i
  }
}
```

Having described the algorithm, we shall try to identify the number of computations needed for the construction of the attractor. Since the contractivity factors of the affine transformations are known, we can find an estimation for the depth of the tree. If s_{max} and s_{min} are the largest and the smallest contractivity factors, respectively, then the depth of the tree will lie between

$$d_{max} = \left[\frac{-\ln M}{\ln s_{max}}\right] + 1 \quad \text{and} \quad d_{min} = \left[\frac{-\ln M}{\ln s_{min}}\right] + 1$$

that are the greatest and the smallest depth, respectively. This means that in the worst case we need $10(N + N^2 + \cdots + N^{d_{max}}) = 10[(N^{d_{max}+1} - 1)/(N-1) - 1]$, thus giving $O(N^{d_{max}})$ while in the best case we need $10(N + N^2 + \cdots + N^{d_{min}}) = 10[(N^{d_{min}+1} - 1)/(N-1) - 1]$ computations. This number is explained by the fact that nine computations are needed for the calculation of the new point and only one computation is needed for the calculation of the quantity $s_{i_1}(s_{i_2} \cdots s_{i_n}\sqrt{2})$ since at each step the quantity $s_{i_2} \cdots s_{i_n}\sqrt{2}$ has already been computed in previous nodes.

The exact number of computations performed before the construction of the attractor is $C(\sqrt{2})$, where

$$C(x) = \begin{cases} 1, & \text{if } x < \varepsilon \\ \sum_{i=1}^{N} C(s_i x) + 10N, & \text{otherwise,} \end{cases}$$

where N is the number of affine transformations and $\varepsilon = 1/M$, M being the resolution of the screen. Although, it is extremely difficult to find a closed formula which gives the exact number of computations, the above mentioned recursive formula can be used to obtain this number with the aid of a computer.

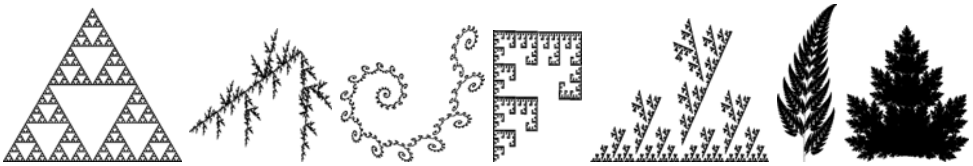


Fig. 7. The Sierpiński triangle, a dendrite, a spiral, a meander, a second dendrite, a fern leaf and a maple leaf.

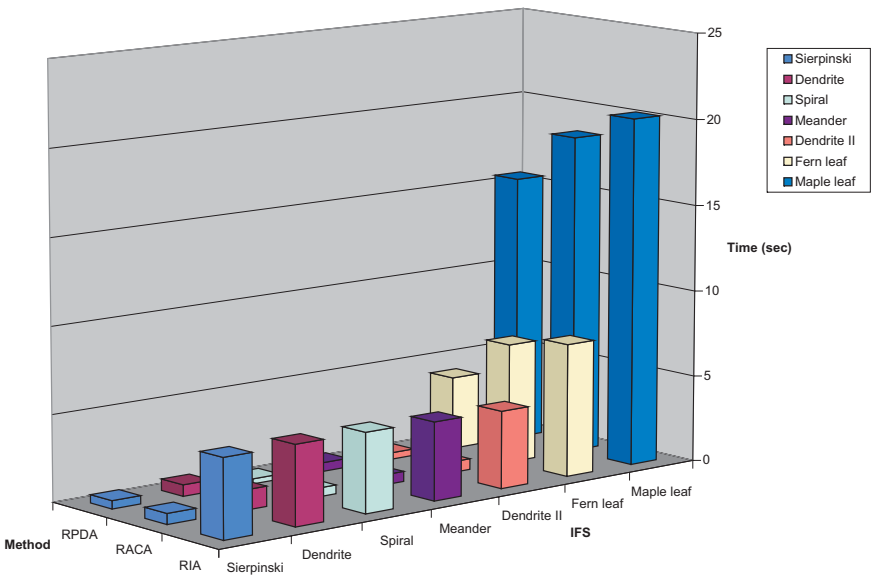


Fig. 8. The time results of the systematic comparison.

6. Comparative Results

The current implementation of our algorithm is written in Microsoft Visual Basic 6.0. It is capable of drawing fractal images using the Random PDA, the RIA and the RACA, and displaying the depth of the action tree, the number of points used for rendering and the total runtime. The fractal images ($M = 100$) used for the comparisons of the various algorithms are illustrated in Fig. 7. The Random PDA was finally tested and rated by comparing the various fractal attractors produced by it versus the attractors produced using the RIA and the RACA. Time results are given in CPU seconds on a Pentium III PC with a 400 MHz CPU clock running Windows 98.

As we have already stated, the proposed algorithm is more efficient than the RIA and the RACA; the systematic comparison we performed gave the results shown in Fig. 8 and justifies our argument.

7. Conclusions

Our algorithm is extremely efficient for the decoding of pictures which have been coded using some method based on IFS. This is explained by the fact that natural images – such as the face of a person – lack self-similarity and hence a lot of affine transformations are employed for the coding; each of them, however, will have a very small contractivity factor and hence our tree will have small depth (10 for the maple leaf, 31 for the fern leaf, 45 for the spiral and 8 for the other figures); so, the construction of the attractor will terminate very quickly, since only a few number of points are necessary (see Table 1).

Table 1. The number of pixels used for the systematic comparison.

Pixels drawn	RPDA	RACA	RIA
Sierpiński	6561	6561	9900
Dendrite	8617	12349	9900
Spiral	3464	4838	9900
Meander	6561	6561	9900
Dendrite II	5413	6561	9900
Fern leaf	57031	71776	49900
Maple leaf	206080	206080	200000

One should also mention that without the use of the equivalent trees, it is doubtful if this performance of the Random PDA could have ever been achieved. Any implementation that avoids their use is bound either to make use of huge amount of memory or unavoidably to resort to recomputations. In both cases, such an implementation becomes highly inefficient and the performance of the algorithm would be substantially decreased.

Appendix

A transformation f is *affine*, if it may be represented by a matrix A and translation $\mathbf{t}$ as $f(\mathbf{x}) = A\mathbf{x} + \mathbf{t}$, or (if $\mathbf{x} \in \mathbb{R}^2$)

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & s \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d \\ e \end{pmatrix}.$$

The *code* of f is the 6-tuple (a, b, c, s, d, e), and the *code of an IFS* is a table whose rows are the codes of $f_1, f_2, \ldots, f_N$. If we add one extra column with the corresponding probabilities $p_1, p_2, \ldots, p_N$, then we are talking about the *code of an IFS with probabilities*.

We list the IFS codes (see Tables 2–8) for the examples discussed in the main text.

Table 2. The IFS code for the Sierpiński triangle.

f	a	b	c	s	d	e	p
1	0.5	0	0	0.5	0	0	0.33
2	0.5	0	0	0.5	0.5	0	0.34
3	0.5	0	0	0.5	0.25	0.5	0.33

Table 3. The IFS code for a dendrite.

f	a	b	c	s	d	e	p
1	0.5	0	0	0.5	0.0625	0.15	0.25
2	0.21	−0.20625	0.528	0.21	0.789	0	0.25
3	0.5	0	0	0.5	0.375	0.375	0.25
4	−0.2	0.1125	−0.288	−0.2	0.609	0.975	0.25

Table 4. The IFS code for a spiral.

f	a	b	c	s	d	e	p
1	−0.18	0.126	−0.2571	−0.18	0.815	0.8485	0.05
2	−0.8	0.4	−0.4	0.8	−0.088	0.2514	0.95

Table 5. The IFS code for a second dendrite.

f	a	b	c	s	d	e	p
1	0.5	0	0	0.5	0.3125	0	0.33
2	0.5	0	0	0.5	0.496094	0	0.33
3	0.28	−0.25	0.64	0.28	0.523437	0.05	0.34

Table 6. The IFS code for a meander.

f	a	b	c	s	d	e	p
1	0	0.33	−0.768	0	0.4609	0.9375	0.33
2	0.5	0	0	0.22	0.5	0.9375	0.33
3	0.5	0	0	0.24	0.5	0.9375	0.34

Table 7. The IFS code for a fern leaf.

f	a	b	c	s	d	e	p
1	0	0	0	0.16	0.5	0.07	0.001
2	0.2	−0.195	0.3066667	0.22	0.41625	0.045	0.070
3	−0.15	0.21	0.3466667	0.24	0.5575	−0.07333	0.070
4	0.85	0.03	−0.5333	0.85	0.07249999	0.1725	0.859

Table 8. The IFS code for a maple leaf.

f	a	b	c	s	d	e	p
1	0.6	0	0	0.6	0.18	0.36	0.295
2	0.6	0	0	0.6	0.18	0.12	0.295
3	0.4	0.3	−0.3	0.4	0.27	0.36	0.25
4	0.4	−0.3	0.3	0.4	0.27	0.09	0.16

Acknowledgements

The authors wish to thank the anonymous reviewer for all the helpful suggestions.

References

1. M. F. Barnsley, *Fractals everywhere* (Academic Press, 1993), 2nd edition.
2. A. Edalat, *Dynamical systems, measures and fractals via domain theory*, *Inform. Comput.* **120** (1995) 32–48.
3. A. Edalat, *Power domains and iterated function systems*, *Inform. Comput.* **124** (1996) 182–197.
4. D. Hepting, P. Prusinkiewicz and D. Saupe, *Rendering methods for iterated function systems*, in *Fractals in the Fundamental and Applied Sciences*, eds. H.–O. Peitgen, J. M. Henriques and L. F. Penedo (North-Holland, 1991), pp. 183–224.
5. D. M. Monro and F. Dudbridge, *Rendering algorithms for deterministic fractals*, *IEEE Computer Graphics Appl.* **15** (1995) 32–41.
6. G. D. Plotkin, *A note on inductive generalization*, in *Machine Intelligence*, eds. B. Meltzer and D. Michie (North-Holland, 1970), Vol. 5, 153–163.