ANNALS OF THE FACULTY OF ENGINEERING HUNEDOARA
2006, Tome IV, Fascicole 1, (ISSN 1584 – 2665)
FACULTY OF ENGINEERING HUNEDOARA,
5, REVOLUTIEI, 331128, HUNEDOARA

THE APPLICATION OF STOCHASTIC GRAMMARS FOR GENERATION OF FRACTALS

IORDAN Anca Elena, PĂNOIU Manuela

UNIVERSITY POLITECHNICA OF TIMIŞOARA,
ENGINEERING FACULTY OF HUNEDOARA

ABSTRACT:
Encoding fractal sets as quad-trees is considered. Every node of the tree is compared to a dictionary which is recursively built. Comparison is performed by looking for self-similarity within images. Geometrical operations (rotation and rescaling) are used to match statistical moments of the images to compare. Numerical results are used to determine the probabilities of assigning to the rules of a context-free grammar inferred from the set of trees.

KEYWORDS:
stochastic fractals, stochastic grammars, probability

1. INTRODUCTION

Syntactic techniques provide a pleasant and almost natural way for generating fractal sets. One of the reasons for their popularity is that the objects actually being processed are symbols related to geometrical primitives (frequently vectors) rather than obscure numerical coefficients.

These symbols are gathered into words, and sequences of words are iteratively built, giving better and better approximations of the fractal set. It is confounding but promising to see how the same principle is able to generate rather simple curves as well as much more complicated sets such as natural plants and forests. Thus, study of fractal sets, regardless of the dimension of the initial space (1, 2 or 3), is transposed into the study of infinite words.

Little attention has been given to the thorny inverse problem. In a few words, it consists in computing the grammar that can generate a given fractal set. Obviously, the interest of considering *fractal words* instead of the sets themselves would fade if it were not possible to find a good inference algorithm. At this point, results gained in syntactic pattern recognition help, in particular to give answers to what follows. Before designing any inference method for fractal sets, one must consider the following points thoroughly:

- Which geometrical primitives must the symbols encode?
- Since the set being analysed is only a discretisation of the actual fractal set (up to the resolution of the camera), how will the structures appear in the encodings?

- What is the standard deviation allowed between the language of the inferred system and the initial set?
- Must this system be minimal (according to some criterion) and how does this modify the properties of the language?

2. FRACTALS

A fractal is a rough or fragment geometric shape that can be subdivided in parts, each o, which is (at least approximately) a reduced-size copy of the whole shape. Fractals are generally self-similar and independent of scale.

There are many mathematical structures that are fractals: Koch snowflake, Peano curve, Mandelbrot set and Lorenz attractor. Fractals also describe many real-world objects, such as clouds, mountains, turbulence, coastlines, roots, branches of trees, blood vessels etc, that do not correspond to simple geometric shapes (lines, arcs, circles, etc.).

Benoit B. Mandelbrot gives a mathematical definition of a fractal: Fractal is set which the Hausdorff-Besicovich dimension strictly exceeds the topological dimension.

Hausdorff dimension can be calculated by taking the limit of the quotient of the *log* change in object size and the *log* change in measurement scale, as the measurement scale approaches zero. The differences come in what is exactly meant by „object size" and what is meant by „measurement scale" and how to get an average number out of many different parts of a geometrical object. Hausdorff dimension quantify the static geometry of an object.

Equation for Hausdorff dimension: $N \times s^D = 1$, where

N – number of parts of an object;

s – scale;

D – Hausdorff dimension.

Topological dimension is the „normal" idea of dimension:

- a point has topological dimension 0;
- a line has topological dimension 1;
- a surfece has topological dimension 2;
- a cube has topological dimension 3, etc.

All fractal types:

- *Deterministic fractals*: fractal is identical for the same initial conditions.
- *Stochastic fractals*: we must use series of random numbers for computing this fractal type.

3. STOCHASTIC GRAMMARS

A stochastic context-free grammar is a probabilistic extension of a context-free grammar. The extension is implemented by adding a probability measure to every production rule: $A \rightarrow \lambda$ [p]. The rule probability p is usually written as P ($A \rightarrow \lambda$). This probability is a conditional probability of the production being chosen, given that non-terminal A is up for expansion (in generative terms). Saying that stochastic grammar is context-free essentially means that the rules are conditionally independent and, therefore, the probability of the complete derivation of a string is just a product of the probability of participating in the derivation.

A stochastic context-free grammar is a context-free grammar such that for each rule $A^i \rightarrow \lambda^j$:

$$\sum_{\{j\}} P(A^i \rightarrow \lambda^j) = 1.$$

The probability of partial derivation $\nu \Rightarrow \mu \Rightarrow ... \Rightarrow \lambda$ is defined in inductive manner as:

a) $P(\nu)=1$;
b) $P(\nu \Rightarrow \mu \Rightarrow ... \Rightarrow \lambda)=P(A \rightarrow w)P(\mu \Rightarrow ... \Rightarrow \lambda)$, where production $A \rightarrow w$ is a production of G , μ is derived from ν by replacing one occurrence of A with w, and $\nu, \mu, ..., \lambda \in V_G^*$.

The string probability $P(S \overset{*}{\Rightarrow} \lambda)$ is the sum of all left-most derivations $A \Rightarrow ... \Rightarrow \lambda$. The prefix probability $P(S \Rightarrow_L^* \lambda)$ is the sum of the strings havings λ as a prefix,

$$P(S \Rightarrow_L^* \lambda) = \sum_{\omega \in V_G^*} P(A \Rightarrow^* \lambda\omega) .$$

4. STOCHASTIC FRACTALS

A fractal grammar consists of an alphabet of symbols that can be used in the strings, a set of production rules which translate each symbol into a non-empty string of symbols and an axiom from wich to begin construction. The fractal is built by recursively feeding the axiom through the production rules. Each caracter of the input string is checked against the rule list to determine which character or string to replace it with in the output string. We specify more than one production rule for a symbol, giving each a probability of occurring.

a) Let be the grammar:
The axiom: S
The alphabet of symbols: L, A, B.
The set of production rules:
S → LLL | LLLL | LLLLLL
L → LABL | LBAL
A → LABL | LBAL
B → LABL | LBAL
The probability of each production rule is:
P(S → LLL)=0.33
P(S → LLLL)=0.33
P(S → LLLLLL)=0.34
P(L → LABL)=1/2
P(L → LBAL)=1/2
P(A → LABL)=1/2
P(A → LBAL)=1/2
P(B → LABL)=1/2
P(B → LBAL)=1/2

Whenever a „S" is encountered during string rewriting, there would be a 33% chance it would behave as LLL, a 33% chance it would behave as LLLL and a 34% chance it would behave as LLLLLL. Whenever a „L" is encountered during string rewriting, there would be a 50% chance it would behave as LABL and a 50% chance it would behave as LBAL.

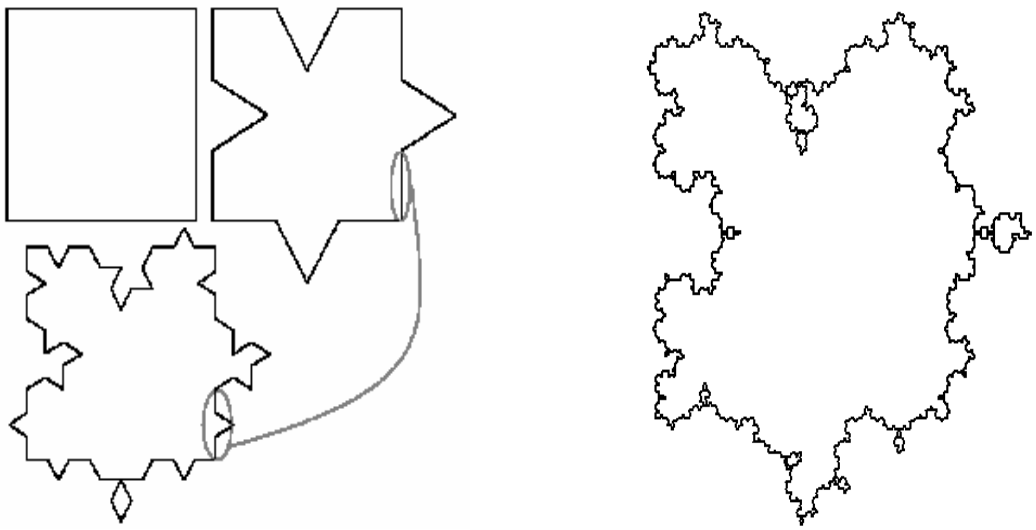

Fig.1. The first 3 iterations and the 5th recursion

The output string is drawn as an image by using a turtle graphical interpretation, where each symbol is assigned a graphical operation for the turtle to perform. For example, the turtle may be given the following instructions:

L: draw a line segment;
A: turn left 60 degrees and draw a line segment;
B: turn right 60 degrees and draw a line segment.
1st recursion : LLLL
2nd recursion: LBALLBALLABLLABL
3rd recursion: LBALLABLLABLLBALLBALLBALLABLLABLLBALLABL
LBALLABLLABLLBALLABLLBAL

b) Let be the grammar:
The axiom: S
The alphabet of symbols: L, C, D.
The set of production rules:
S → LLL | LLLL | LLLLLL
L → LCLDL | LDLCL
C → LCLDL | LDLCL
D → LCLDL | LDLCL
The probability of each production rule is:
P(S → LLL)=0.33
P(S → LLLL)=0.33
P(S → LLLLLL)=0.34
P(L → LCLDL)=1/2
P(L → LDLCL)=1/2
P(C → LCLDL)=1/2
P(C → LDLCL)=1/2
P(D → LCLDL)=1/2
P(D → LDLCL)=1/2

Whenever a „S" is encountered during string rewriting, there would be a 33% chance it would behave as LLL, a 33% chance it would behave as LLLL and a 34% chance it would behave as LLLLLL.Whenever a „L" is encountered during string rewriting, there would be a 50% chance it would behave as LCLDL and a 50% chance it would behave as LDLCL.

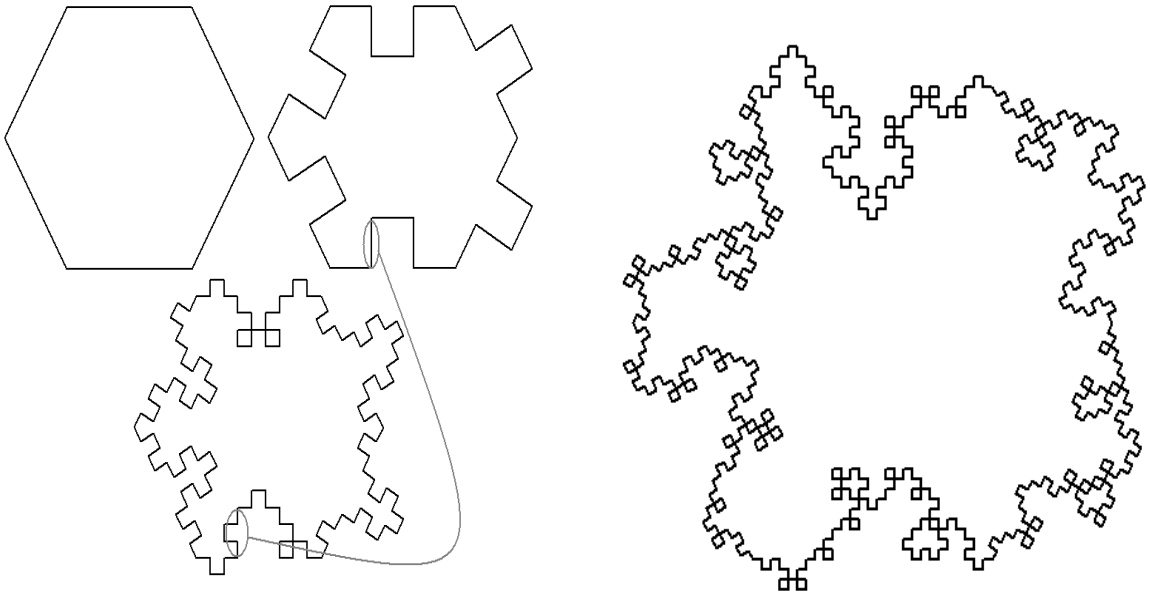

Fig. 2 The first 3 iterations and the 4th recursion

The output string is drawn as an image by using a turtle graphical interpretation, where each symbol is assigned a graphical operation for the turtle to perform. For example, the turtle may be given the following instructions:

L: draw a line segment;
C: turn left 90 degrees and draw a line segment;
D: turn right 90 degrees and draw a line segment.
1st recursion: LLLLLL
2nd recursion: LCLDLLDLCLLDLCLLCLDLLCLDLLCLDL
3rd recursion: LDLCLLCLDLLCLDLLCLDLLCLDLLCLDLLCLDLLDLCL
LCLDLLCLDLLCLDLLDLCLLDLCLLCLDLLCLDLLDLCL
LDLCLLDLCLLDLCLLDLCLLDLCLLDLCLLCLDLLDLCL
LCLDLLCLDLLDLCLLCLDLLDLCLLCLDL

c) Let be the grammar:
The axiom: S
The alphabet of symbols: L, A, B, C, D.
The set of production rules:
S → LLL | LLLL | LLLLLL
L → LABL | LBAL | LCLDL | LDLCL
A → LABL | LBAL
B → LABL | LBAL
C → LCLDL | LDLCL
D → LCLDL | LDLCL
The probability of each production rule is:
P(S → LLL)=0.33
P(S → LLLL)=0.33
P(S → LLLLLL)=0.34
P(L → LABL)=1/4
P(L → LBAL)=1/4
P(L → LCLDL)=1/4
P(L → LDLCL)=1/4

P(A → LABL)=1/2
P(A → LBAL)=1/2
P(B → LABL)=1/2
P(B → LBAL)=1/2
P(C → LCLDL)=1/2
P(C → LDLCL)=1/2
P(D → LCLDL)=1/2
P(D → LDLCL)=1/2

Whenever a „S" is encountered during string rewriting, there would be a 33% chance it would behave as LLL, a 33% chance it would behave as LLLL and a 34% chance it would behave as LLLLLL.

Whenever a „L" is encountered during string rewriting, there would be a 25% chance it would behave as LABL, a 25% chance it would behave as LBAL, a 25% chance it would behave as LCLDL and a 25% chance it would behave as LDLCL.

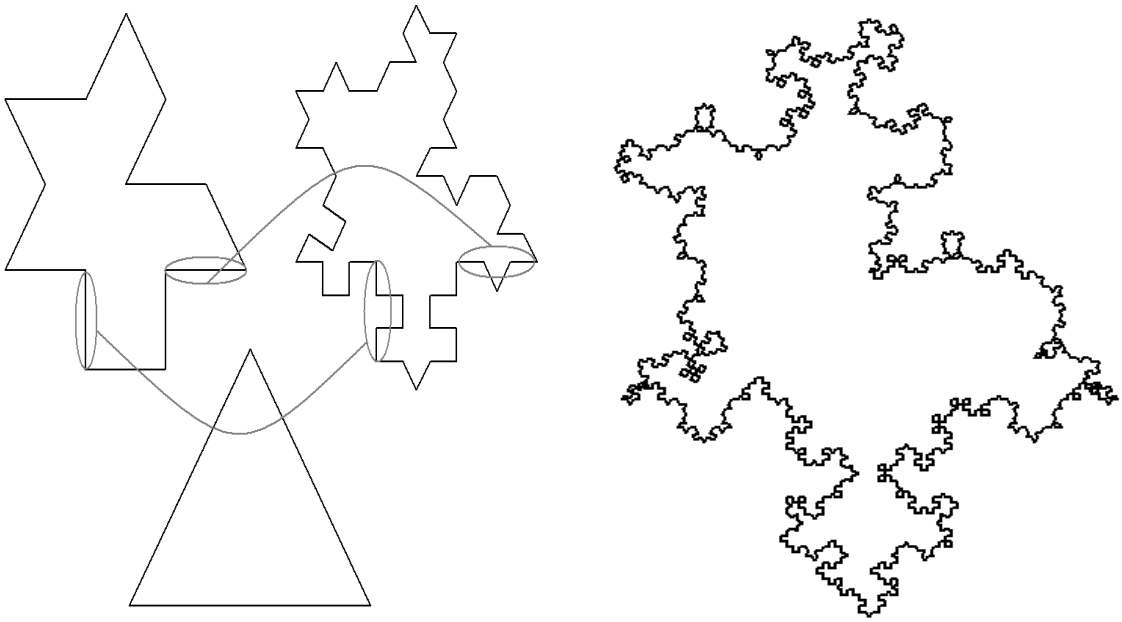

Fig. 3. The first 3 iterations and the 5th recursion

The output string is drawn as an image by using a turtle graphical interpretation, where each symbol is assigned a graphical operation for the turtle to perform. For example, the turtle may be given the following instructions:

L: draw a line segment;
A: turn left 60 degrees and draw a line segment;
B: turn right 60 degrees and draw a line segment;
C: turn left 90 degrees and draw a line segment;
D: turn right 90 degrees and draw a line segment.
1st recursion: LLL
2nd recursion: LDLCLLABLLBAL
3rd recursion: LDLCLLCLDLLBALLCLDLLBALLABLLABLLBALLBAL
LABLLBALLBALLCLDL

The application is implemented in Borland Delphi 6.0, under Microsoft Windows operating system. In terms of implementation, a fractal grammar is a type of a list of nodes. To implement the output string it was defined an object type which include the following elements: the node structure and all the procedures and the functions for creating the output string and for traversing the output string.

```
type segment=record
          v:real;
          m:real;
        end;
  neterminal=record
          c:char;
          s:segment;
         end;
  productie=record
          m1:neterminal;
          m2:array[1..10] of neterminal;
          nr:byte;
          p:real;
         end;
  gramatica=record
          axioma:neterminal;
          neterm:array[1..10] of neterminal;
          p:array[1..20] of productie;
          nr,nn:byte;
         end;
pnod=^nod;
  nod=record
          s:neterminal;
          leg:pnod;
         end;
  lista=object
          public
            procedure init;
            procedure adaug(n:neterminal);
            procedure sterg(var n:neterminal);
            function vida:boolean;
            function prim:pnod;
            function urmator(p:pnod):pnod;
          private
            f,s:pnod;
        end;
```

5. CONCLUSIONS

In this work it presents the way of generation of a fractal when the regularity of placement for primitive elements is effectuated by local perturbation. This kind of fractals is described by stochastic grammars.

The fractal it means as a compose structure of primitive elements, which repeat more or lest regulated. The number and the type of primitive elements who compose it can describe it and by the rules which makes the spatial distribution of primitive elements. The fractals can be deterministic or stochastic.

In case of stochastic fractals the way of repletion of primitive elements respects some statistic rules.

REFERENCES / BIBLIOGRAPHY

[1.] Pengjian Shang, Santi Kamae, *Fractal nature of time series in the sediment transport phenomenon*, Chaos, Solitons, & Fractals, Volume 26, Issue 3 , November 2005, Pages 997-1007

[2.] Knuth D. E., *Fundamental Algorithms*, Third Edition, Massachusetts: Addison-Wesley, 1997

[3.] Robert L. Devaney, Linda Keen, editors, *Chaos and fractals: the mathematics behind the computer graphics*, American Mathematical Society, Providence, RI, 1989

[4.] Fournier, A., Fussell, D., Carpenter, L., *Computer Rendering of Stochastic Models*, Communications of the ACM, June 1982

[5.] B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman, San Fzancisco, 1982.

[6.] R. Szeliski, D. Terzopoulos, *Constrained fractals using stochastic relaxation*, Submitted to A CM Transactions on Graphics, 1989.

[7.] H.-O. Peitgen, P. H. Richter, *The Beauty of Fractals*, Springer, Berlin, New York, Tokyo, 1986