Plane-and Space-Filling Trees by Means of Chain Coding

E. Bribiesca*[1], N. Espinosa-Dominguez[2]

[1,2] Department of Computer Science
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
Apdo. Postal 20-726, México, D.F., 01000. Fax: (5255)5622-3620
*E-mail: ernesto@leibniz.iimas.unam.mx

ABSTRACT
An algorithm for constructing fractal trees is presented. Fractal trees are represented by means of the notation called the unique tree descriptor [E. Bribiesca, A method for representing 3D tree objects using chain coding, J. Vis. Commun. Image R. 19 (2008) 184-198]. In this manner, we only have a one-dimensional representation by each fractal tree via a chain of base-five digit strings suitably combined by means of parentheses. The unique tree-descriptor notation is invariant under rotation and translation. Furthermore, using this descriptor it is possible to obtain the mirror image of any fractal tree with ease. In this paper, we focus on fractal plane-filling trees and space-filling trees.

Keywords: Fractal trees, tree descriptor, plane-filling trees, space-filling trees, chain coding.

RESUMEN
Se presenta un algoritmo para la construcción de árboles fractales. Un árbol fractal es representado por medio de la notación llamada descriptor único de árboles [E. Bribiesca, A method for representing 3D tree objects using chain coding, J. Vis. Commun. Image R. 19 (2008) 184-198]. De esta manera, se tiene solamente una representación unidimensional para cada árbol fractal por medio de una cadena de digitos de base cinco adecuadamente combinados por medio de paréntesis. La notación del descriptor único de árboles es invariante bajo rotación y traslación. Además, usando este descriptor es posible obtener la imagen especular de cualquier árbol fractal con facilidad. El contenido de este artículo se enfoca en el estudio de los árboles fractales que cubren el plano y el espacio.

1. Introduction

Fractal trees are often used in different fields. Fractal trees were primarily studied by Mandelbrot [2]. Mandelbrot and Frame [3] analyzed plane-filling trees. Gonzalez [4] presented a tutorial and recipe for moving fractal trees. Frongillo et al. [5] gave an interesting method for constructing fractal trees in 3D (three dimensions). This paper deals with plane- and space-filling trees by means of tree descriptors. Thus, a fractal tree is represented by means of a chain of base-five digit strings suitably combined by means of parentheses. In order to describe fractal trees, we use the tree descriptor which was presented in [1]. Tree structures cover a wide variety of applications [6].

In graph theory there is no unique way of drawing a graph; the relative positions of points representing vertices and lines representing edges have no significance [6]. However, in the proposed method for representing fractal trees via tree descriptors, the topology and geometry of 3D tree structures should be preserved. The tree-descriptor notation is invariant under *rotation* and *translation*. Furthermore, using this notation, it is possible to obtain the *mirror image* of any fractal tree with ease. The tree-descriptor notation preserves the shape of fractal trees and allows us to know their topological and geometrical properties. The proposed method for representing the topology and geometry of fractal trees using tree descriptors preserves information and allows a considerable data reduction, which is an important advantage in computer vision, image representation, and pattern recognition.

This paper is organized as follows. In Section 2, we present a set of concepts and definitions, which are important for introducing the proposed method of

fractal trees. Section 3 gives the method for fractal-tree representation by means of chain coding. Section 4 gives some examples of plane- and space-filling trees. Finally, in Section 5, we present some conclusions.

2. Concepts and definitions

In order to introduce our proposed algorithm for constructing fractal trees, a number of concepts and definitions are presented below:

• *Shape* is referred to as shape of object, and an object is considered to be a geometric entity.

• The term *sphere* means the enclosing surface together with its interior.

• *Branches* of trees are represented as *pipelines*; this improves the understanding of fractal trees.

• The tree *descriptor* of a fractal tree is defined by the computation of the chain elements of all its branches using the parenthesis notation.

2.1 The tree descriptor

In order to have a self-contained paper, we summarize the main concepts of the tree descriptor which was presented in [1]. The tree descriptor is a useful tool to represent plane- and space-filling trees. A *graph* is composed of a set of points called *vertices v*, joined by e*dges* (branches) e. A graph is *connected* if there is a path between any two vertices of the graph. A tree is a connected graph which contains no cycles [6]. In a tree, any two vertices are connected by a unique path and the number of edges is equal to the number of vertices minus one, i.e.

$$e = v - 1. \qquad (1)$$

The *degree* of a vertex in a tree is the number of edges incident to it. The tree descriptor describes trees of maximum degree six in three dimensions. This is due to the fact that only orthogonal straight-line segments are used.

Definition 1. *A chain a is an ordered sequence of m elements and is represented by*

$$a = a_1 a_2 ... a_m = \{a_i : 1 \le i \le m\}. \qquad (2)$$

Definition 2. *Branches of trees are composed of constant straight-line segments (length l of each straight-line segment is considered equal to one). Two contiguous straight-line segments of a branch define a direction change and two direction changes define a chain element.*

Definition 3. *An element a_i of the set* {0, 1, 2, 3, 4} *of a chain indicates the orthogonal direction changes of the contiguous straight-line segments of the 3D branch in that element position.*

Each element of the chain labels a vertex of the branch and indicates the orthogonal direction changes of the polygonal path in such a vertex. Figures 1(a)-(e) illustrate the only five possible chain elements [7, 8]. In order to improve the understanding of the chain elements, we have colored the straight-line segments which are defined by their corresponding chain elements. Thus, the straight-line segment defined by the chain element “0” in green, “1” in cyan, “2” in yellow, “3” in magenta, and “4” in red, respectively. Formally an element a_i of a chain, taken from the set {0, 1, 2, 3, 4}, labels a vertex of the branch and indicates the orthogonal direction change of the polygonal path in such a vertex. Figures 1(a)-(e) summarize the rules for labelling the vertices: to a straight-angle vertex, a “0” is attached; to a right-angle vertex, one of the other labels corresponds, depending on the position of such an angle with respect to the preceding right angle in the path. If the consecutive sides of the reference angle have respective directions *b* and *c* (see Figure 1(a)), and the side from the vertex to be labelled has direction d (from here on, by direction, we understand a vector of length 1), then the chain element *ch* or label is given by the following function, where × denotes the vector product in $\Re^3$.

$$ch(b,c,d) = \begin{cases} 0, \text{ if } d = c, \text{ in green;} \\ 1, \text{ if } d = b \times c, \text{ in cyan;} \\ 2, \text{ if } d = b, \text{ in yellow;} \\ 3, \text{ if } d = -(b \times c), \text{ in magenta;} \\ 4, \text{ if } d = -b, \text{ in red;} \end{cases} \qquad (3)$$

Thus, the procedure to find the tree descriptor is as follows:

(i) S*elect* an arbitrary end vertex of the tree as the origin. Figure 1(f) illustrates the selected origin which is represented by a sphere.

(ii) *Compute* the chain elements of the tree. Figure 1(f) shows the first element of the chain which corresponds to chain element "3". Note that the first direction change (which is composed of two contiguous straight-line segments in white) is used only for reference. The second element corresponds to chain element "3" too, this is shown in Figure 1(f). The third element corresponds to chain element "3" again. The fourth element corresponds to chain element "0". The fifth element corresponds to chain element "0". The sixth element corresponds to chain element "0", too. Thus, in this stage, our chain is as follows: 333000.

(iii) The simplest output of a tree is the well-known parenthesis notation [9]. Using this notation, there is a correspondence between trees and nested parentheses. In order to define the next chain element of Figure 1(f), we have touched a vertex which is a junction. In what direction to go? Generally speaking, there are only five possible ways represented by the previous defined chain elements. In the case of the tree shown in Figure 1(f), there are only three possible ways represented by chain elements "1", "2", and "3". Note that when we are travelling around a branch, in order to obtain its chain elements and find zero elements, we need to know what nonzero element was the last one in order to define the next element. In this manner, orientation is not lost. We always select the directions in *numerical order*. Thus, first we select the direction represented by chain element "1" and we obtain 333000(12). The nested parenthesis describe the branch whose chain is (12). Then, we come back for the junction node and compute the chain of the next branch, which is (220), in this stage the tree descriptor is equal to 333000(12)(220). Finally, we come back for the junction node and compute the chain of the last branch, which is (311). So, the tree descriptor of the tree shown in Figure 1(f) is as follows:

333000(12)(220)(311)

A complete review of the tree-descriptor notation can be found in [1]

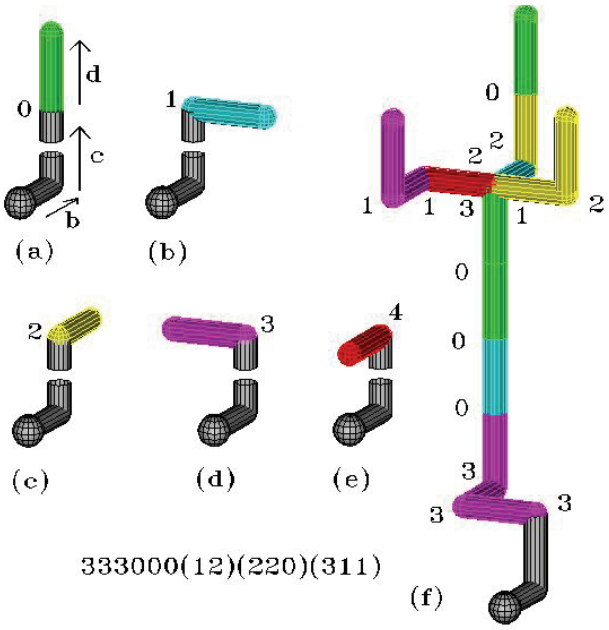

Figure 1. An example of a tree on five vertices: (a) chain element "0" in green; (b) chain element "1" in cyan; (c) chain element "2" in yellow; (d) chain element "3" in magenta; (e) chain element "4" in red; (f) chain elements of an example of a tree and its tree descriptor.

3. Fractal-tree representation by means of chain coding

Fractal trees are represented by means of tree descriptors. The tree-descriptor notation is described by chains of base-five digit strings suitably combined by means of parentheses. The *tree descriptor* of a fractal tree is defined by the computation of the chain elements of all its branches using the parenthesis notation. In this section, we describe plane- and space-filling trees.

3.1 Plane-filling trees

A plane-filling tree is a fractal tree that tends to occupy a defined plane as the number of iterations increases. In order to generate the chain, we propose the next recursive algorithm: first, an initial chain *(2(2)(4))(4(2)(4))* was defined. Then, we replace (*2)(4)* by *(2(2)(4))(4(2)(4))*. This replacement may be repeated n times, where n is the number of iterations. As n tends to infinity, the tree will fill the plane. This algorithm may be represented by

/* s0 is the initial string */
s0="(2(2)(4))(4(2)(4))";
for *i*=1 **to** *n* **do**
Search every substring "(2)(4)" in s0 and replace by "(2(2)(4))(4(2)(4))"
end

The initial four iterations are as follows:

n=1, (2(2)(4))(4(2)(4))

n=2, (2(2(2)(4))(4(2)(4)))(4(2(2)(4))(4(2)(4)))

n=3,(2(2(2(2)(4))(4(2)(4)))(4(2(2)(4))(4(2)(4))))
(4(2(2(2)(4))(4(2)(4)))(4(2(2)(4))-
(4(2)(4))))

n=4,
(2(2(2(2(2)(4))(4(2)(4)))(4(2(2)(4))(4(2)(4))))(4(2(2(2
)(4))(4(2)(4)))(4(2(2)(4))-
(4(2)(4)))))(4(2(2(2(2)(4))(4(2)(4)))(4(2(2)(4))(4(2)(4)
)))(4(2(2(2)(4))(4(2)(4)))(4(2(2)(4))- (4(2)(4)))))

3.2 Space-filling trees

A space-filling tree is a fractal tree that fills a defined space in $\mathbf{R}^3$. The algorithm for constructing space-filling trees is exactly the same as the first one used to generate plane-filling trees. To obtain the space-filling tree, string (1)(3) must be replaced by (1(1)(3))(3(1)(3)). The initial chain must be (1(1)(3))(3(1)(3)). The replacement can be done n times, where n is the number of iterations. As n tends to infinity, the tree will fill the space. This algorithm may be represented by

/* s0 is the initial string
s0="(1(1)(3))(3(1)(3))";
for *i=1* **to** *n* **do**
Search every substring "(1)(3)" in s0 and replace by "(1(1)(3))(3(1)(3))"
end

The initial four iterations are shown below:

n=1, (1(1)(3))(3(1)(3))

n=2, (1(1(1)(3))(3(1)(3)))(3(1(1)(3))(3(1)(3)))

n=3,(1(1(1(1)(3))(3(1)(3)))(3(1(1)(3))(3(1)(3))))
(3(1(1(1)(3))(3(1)(3)))(3(1(1)(3))-
(3(1)(3))))

n=4,
(1(1(1(1(1)(3))(3(1)(3)))(3(1(1)(3))(3(1)(3))))
(3(1(1(1)(3))(3(1)(3)))(3(1(1)(3))-
(3(1)(3)))))(3(1(1(1(1)(3))(3(1)(3)))(3(1(1)(3))(3(1)
(3))))(3(1(1(1)(3))(3(1)(3)))(3(1(1)(3))- (3(1)(3)))))

3.3 The tree construction algorithm

This algorithm consists in the following steps:
1. Define two auxiliary variables: *level* and *scale*. The first one should be an integer and the second one, a double or float.

2. Initialize the variables. Variable level should be always equal to zero and *scale* should be within the interval (0,1].

3. Get an input string *Str* which contains the Chain Code.

4. For *i=1* to length of the *string do*:

• If *Str[i]* is equal to "(" then level++.

•Get the substrings and make the cases *0, 1, 2, 3, or 4* to obtain the *segments* or the *Chain Code Directions*.

• Else if *Str[i]* is equal to ")" then *level*–.

5. Scale the segments in sc*ale*^{level} .

This is represented by the shown algorithm.

3.4 The scale factor for plane-filling trees

If the scale factor is equal to $1/\sqrt{2}$, then the tree will fill the plane. According to [4], there is a straightforward way to demonstrate this:
Let be K_1, K_2 , . . . K_n, . . . scale factors for every segment length; where K_1 corresponds to the first segment, K_2 to the second ones, and so forth. Product K_1 K_2 should be equal to 1/2 to make the Hausdorff Besikovitch dimension equal to 2 and the tree will fill the plane. To get this K_1=r, K_2=r,... K_n=r,...; where r is a fixed scale factor. Then $rr=r^2$ =1/2 and thus $r=1/\sqrt{2}$.

Figure 2 illustrates the four initial stages of plane-filling trees from $n = 1$ to $n = 4$. In order to have a better representation of plane-filling trees, the two initial reference straight- line segments of the fourth stage of the fractal tree of Figure 2 were removed.

3.5 The scale factor for space-filling trees

In order to get a space-filling tree, the scale factor should be $1/\sqrt[3]{2}$. Similar to the plane-filling tree, there is a straightforward demonstration as well.

According to [2], let K_1, K_2 , . . . K_n, . . . be scale factors for every tree branch; where K_1 corresponds to the first branch, K_2 to the second ones, etc. To make the Hausdorff Besikovitch dimension equal to 3 and get a filled space, the following should be accomplished: $K_1K_2K_3 = ½$. To perform this condition, The following is necessary: $K_1=r$, $K_2=r$,... $K_n=r$, ...; where r is a fixed scale factor. Then $rrr = r^3 = 1/2$ and thus $r = 1/\sqrt[3]{2}$.

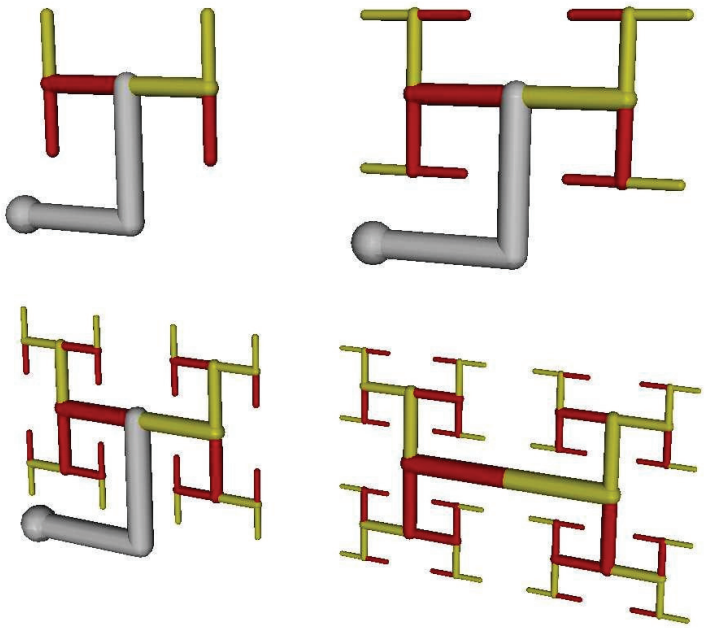

Figure 2. The four initial stages of plane-filling trees.

```
/* "file" contains the "Chain Code String" */
Input: file.txt
/* 0 < scale ≤ 1  */
Input: scale
level=0; Str=file.txt;
Lstr=Length of the Str;
for i=1 to Lstr do
        if Str[i]="(" then
                level++;
                Sstr=Substring of Str ;
        Lsstr=Length of the Sstr;
        for j=1 to Lsstr do
                switch Sstr[j] do
                case.."0"
                end
                case.."1"
                end

                case.."4"
                end
        end
    end
    Scale the Chain Code Directions in scalelevel
    else
        level–;
    end
end
```

Another way to demonstrate this value is in [5].

Figure 3 presents the four initial stages of space-filling trees from n = 1 to n = 4. In order to have a better representation of space-filling trees, the two initial reference straight-line segments of the fourth stage of the fractal tree of Figure 3 were removed.

3.6 Mirror images of space-filling trees

The mirror image of any space-filling tree is obtained with ease.

Definition 4. *Let a be a tree descriptor of a space-filling tree, mirror a is the descriptor obtained by replacing in a each occurrence of 1 by 3 and vice versa.*

Theorem 5. *Let a be a tree descriptor, then mirror (mirror a) = a.*

Considering Definition 4 and Theorem 5 for tree descriptors: the descriptor of the mirror image of a space-filling tree is another descriptor (termed mirroring descriptor) whose elements "1" are replaced by elements "3" and vice versa, preserving its nested parentheses.

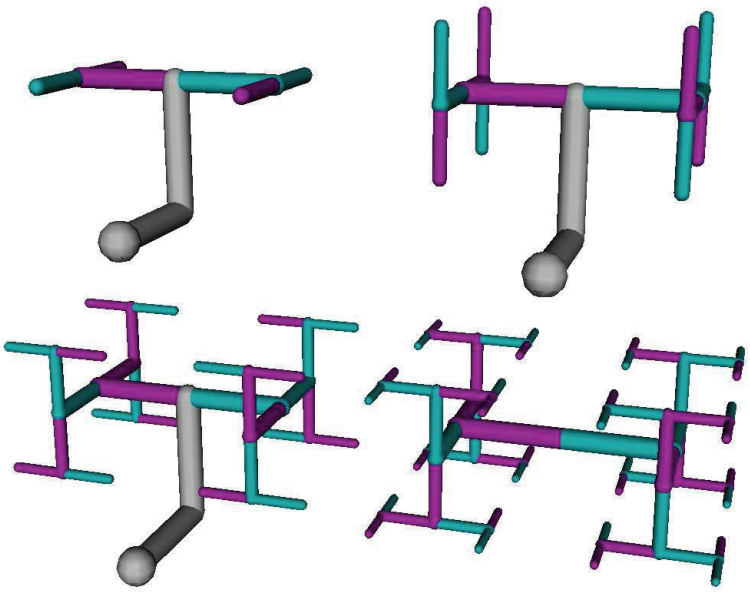

Figure 3. The four initial stages of space-filling trees.

Figure 4 shows the mirror images of the space-filling tree for n = 3 shown in Figure 3. In Figure 4(a) the mirroring plane is aligned with standard plane "XY". In Figure 4(b) the mirroring plane is aligned with plane "XZ" and in (c) with plane "YZ", respectively. Thus, the tree descriptor of the space-filling tree for n = 3 presented in Figure 3 is as follows:

(1(1(1(1)(3))(3(1)(3)))(3(1(1)(3))(3(1)(3))))(3(1(1(1)(3))(3(1)(3)))(3(1(1)(3))(3(1)(3))))

And its mirroring descriptor is equal to (3(3(3(3)(1))(1(3)(1)))(1(3(3)(1))(1(3)(1))))(1(3(3(3)(1))(1(3)(1)))(1(3(3)(1))(1(3)(1)))) Notice that elements "1" and "3" of the mirroring descriptor were changed; thus, this change does not depend on the orthogonal mirroring plane used, it is valid for all orthogonal planes. In order to enhance the mirroring property of the descriptors of space-filling trees, we have not sorted the parentheses (in the mirroring descriptor) in numerical order as it was mentioned in Subsection 2.1.

4. Some examples of plane- and space-filling trees

In order to prove our method of construction of plane- and space-filling trees using chain coding, we present some examples of plane- and space-filling trees. Figure 5 shows the seventh stage of a plane-filling tree. Figure 6 illustrates the ninth stage of a plane filling tree. Figure 7 presents the ninth stage of a space-filling tree. Finally, Figure 8 shows the eleventh stage of a space-filling tree.

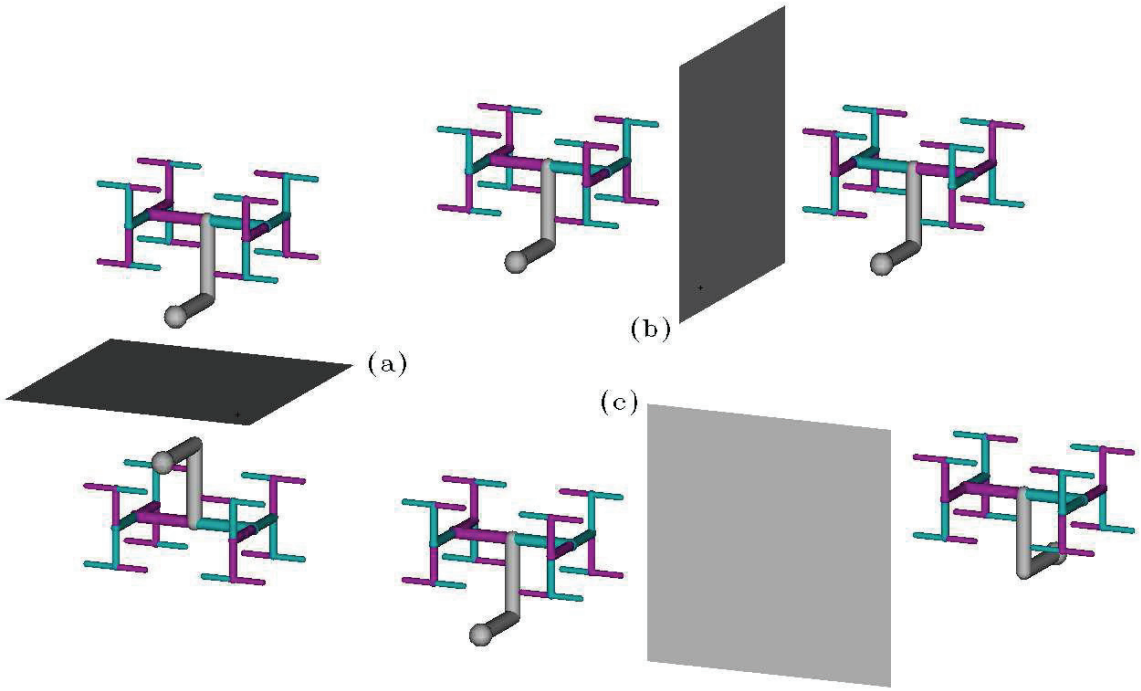

Figure 4. Mirror images of space-filling trees: (a) the mirroring plane is aligned with standard plane "XY", (b) the mirroring plane is aligned with plane "XZ", and (c) with plane "YZ", respectively.

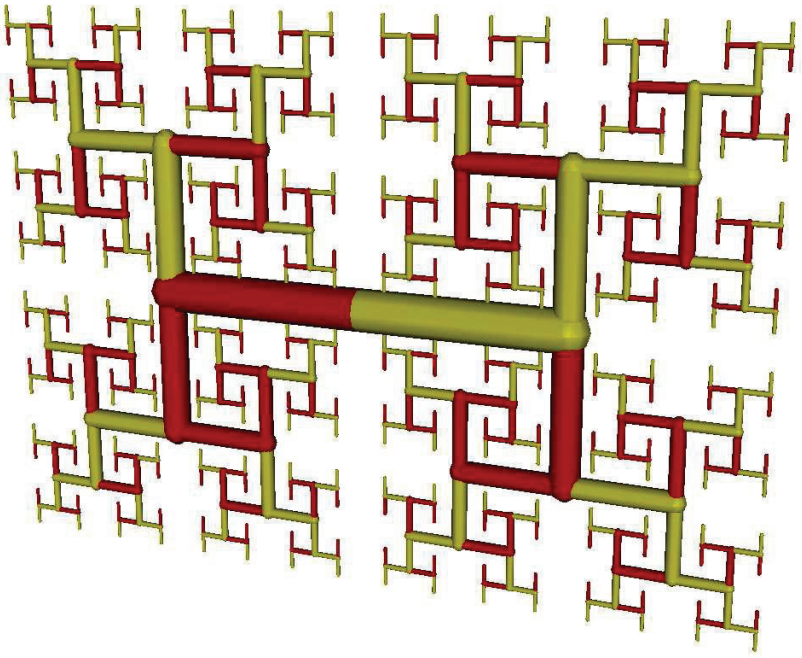

Figure 5. The seventh stage of a plane-filling tree.

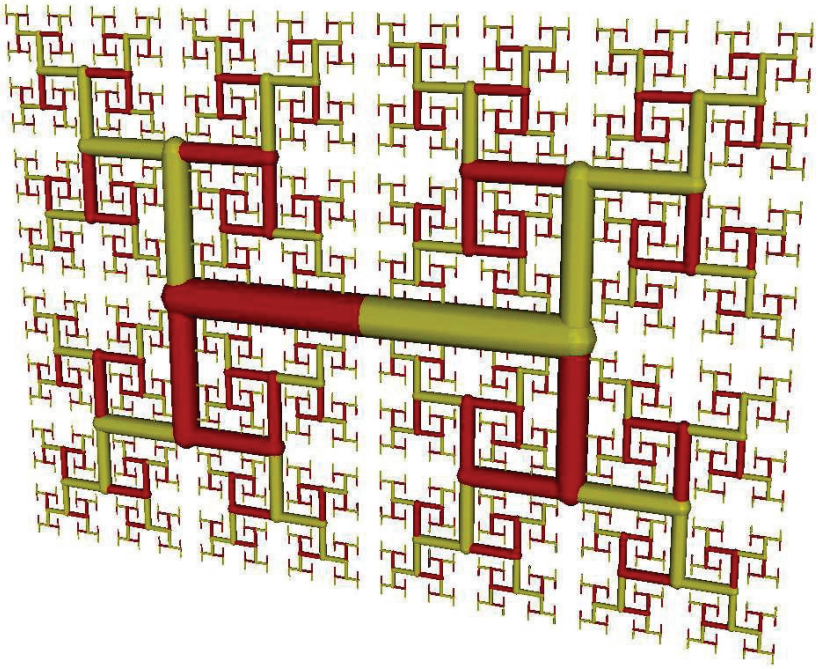

Figure 6. The ninth stage of a plane-filling tree.

5. Conclusions

We have described an algorithm for constructing plane- and space-filling trees by means of the tree-descriptor notation which is represented by chains of base-five digit strings suitably combined by means of parentheses. The tree-descriptor notation preserves the *shape* of plane- and space-filling trees and allows us to know their topological and geometrical properties. The tree-descriptor notation is invariant under *rotation* and *translation*. Furthermore, using this notation, it is possible to obtain the *mirror image* of any space-filling tree with ease.

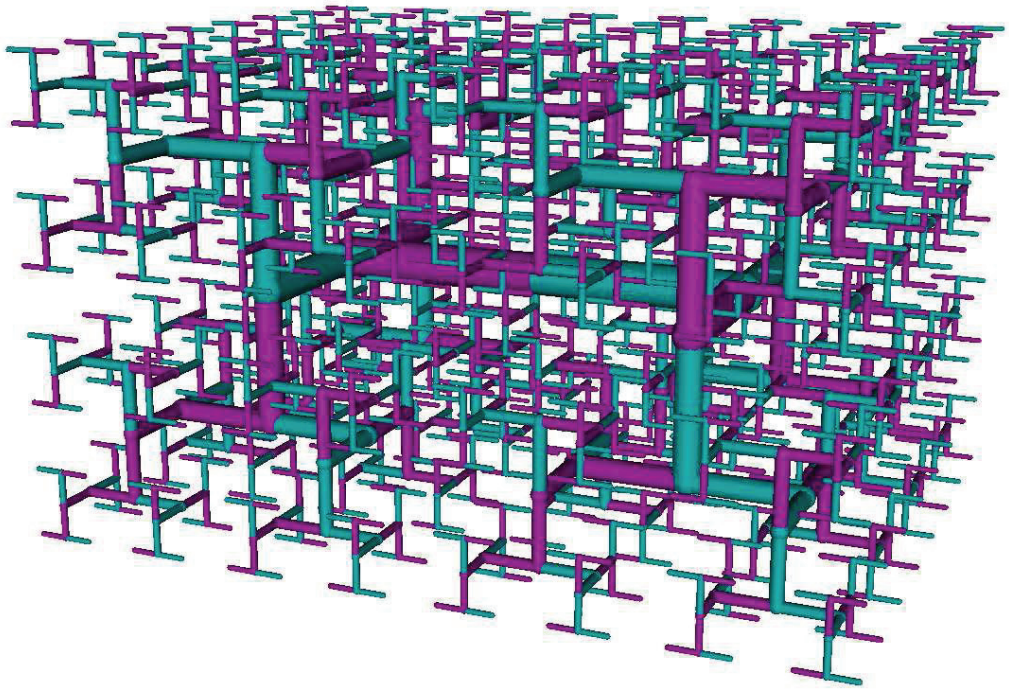

Figure 7. The ninth stage of a space-filling tree.

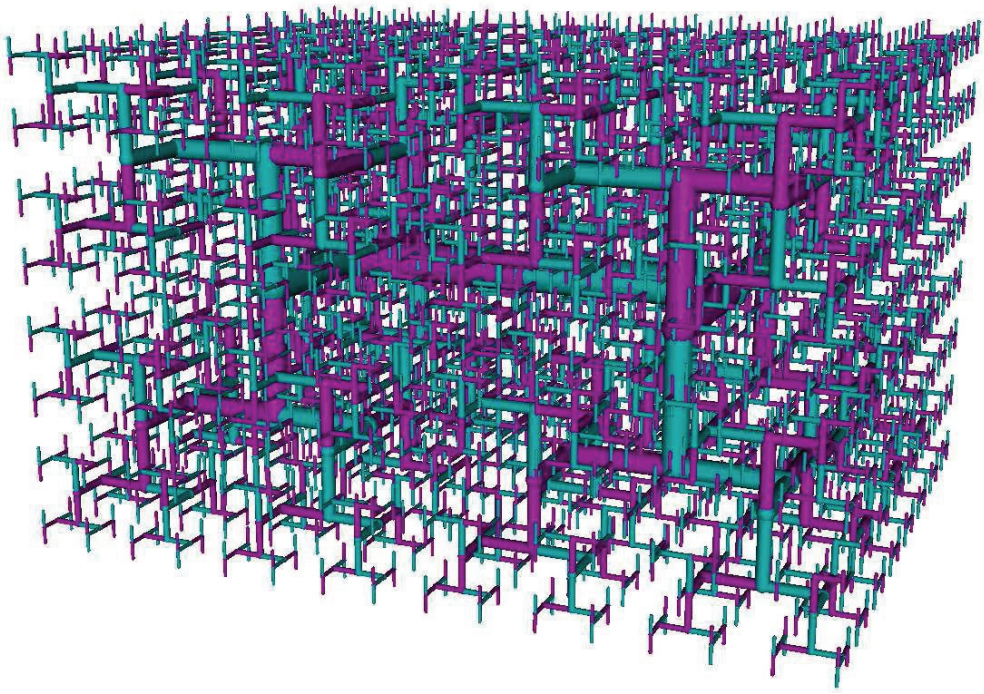

Figure 8. The eleventh stage of a space-filling tree.

References

[1] E. Bribiesca, A method for representing 3D tree objects using chain coding, Journal of Visual Communication and Image Representation, 19, 2008, pp. 184-198.

[2] B. B. Mandelbrot, The Fractal Geometry of Nature, Freeman, New York, 1983.

[3] B. B. Mandelbrot and M. Frame, The canopy and shortest path in a self-contacting fractal tree - exactly what tangles the branches can get into, Math Intelligencer, 21,1999, pp. 18.

[4] J. A. Gonzalez Rodriguez, A tutorial and recipe for moving fractal trees, Comput. & Graphics, 22, 1998, pp. 301-305.

[5] R. M. Frongillo, E. Lock, and D. A. Brown, Symmetric fractal trees in three dimensions, Chaos, Solitons & Fractals, 32, 2007, pp. 284-295.

[6] J. A. Bondy and U. S. R. Murty, Graph Theory with Applications, MacMillan Press, London, 1976.

[7] A. Guzmán, Canonical shape description for 3-d stick bodies, MCC Technical Report Number: ACA-254-87, Austin, TX. 78759, 1987.

[8] E. Bribiesca and C. Velarde, A formal language approach for a 3D curve representation,Computers & Mathematics with Applications, 42, 2001, pp. 1571-1584.

[9] A. Cayley, A theorem on trees, Quart. J. Math. 23, 1889, pp. 376-378.

Authors´ Biographies

Ernesto BRIBIESCA

Dr. Bribiesca received his BSc degree in electronics engineering from Instituto Politécnico Nacional (Polytechnic National Institute) in 1976 and his PhD degree in mathematics from Universidad Autónoma Metropolitana (Metropolitan Autonomous University), UAM, in 1996. He was a researcher at the IBM Latin American Scientific Center and at Dirección General de Estudios del Territorio Nacional (General Directorate of National Territory Studies), DETENAL. He is the associate editor of the *Pattern Recognition* Journal. He has twice been chosen Honorable Mention Winner of the Annual Pattern Recognition Society Award. He is currently a professor at Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (Research Institute for Applied Mathematics and Systems), IIMAS, of Universidad Nacional Autónoma de México (National Autonomous University of Mexico), UNAM, where he teaches graduate courses on pattern recognition.

Nayeli ESPINOSA

Nayeli Espinosa was born in Mexico City. She worked as a grant holder at Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, IIMAS, of Universidad Nacional Autónoma de México, UNAM, in 2006. She graduated as a bionics engineer from Instituto Politécnico Nacional, IPN, in 2007. Currently, she is a master's degree student in microsystems engineering at the Furtwangen Hochschule University, FHU, and collaborates with Institut für Mikrosystemtechnik, IMTEK, in Germany.