ENCLOSING TREES

ERNESTO BRIBIESCA[1], ADOLFO GUZMÁN[2], LUIS A. MARTÍNEZ[3]

[1] *Departamento de Ciencias de la Computación,*

Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas,

Universidad Nacional Autónoma de México,

Apdo. 20-726, México, D.F.,

01000. Fax: (5255)5622-3620,

ernesto@leibniz.iimas.unam.mx

[2]*Centro de Investigación en Computación,*

Instituto Politécnico Nacional,

07738 México City, México,

a.guzman@acm.org

[3]*Instituto de Astronomía,*

Universidad Nacional Autónoma de México,

lamb@astroscu.unam.mx

(Dated: March 23, 2010)

Abstract

A method is described for representing voxel-based objects by means of enclosing trees. An enclosing tree is a tree which totally covers a voxel-based object, the vertices of the enclosing tree correspond to the vertices of the enclosing surface of the analyzed voxel-based object. An enclosing tree is represented by a chain of base-five digit strings suitably combined by means of parentheses. The enclosing-tree notation is invariant under *rotation* and *translation.* Furthermore, using this notation it is possible to obtain the *mirror image* of any voxel-based object with ease. The enclosing-tree notation preserves the *shape* of voxel-based objects, allowing us to know some of their topological and geometrical properties. Also, the proposed enclosing-tree notation is a good tool for storing of voxel-based objects.

Keywords: Voxel-based objects; enclosing trees; enclosing surfaces; chain coding; 3D tree representation

1 INTRODUCTION

Representation of voxel-based objects is an important topic in computer vision and pattern recognition. This paper deals with voxel-based object representation by means of enclosing trees. The enclosing trees totally cover voxel-based objects, the vertices of an enclosing tree correspond to the vertices of the enclosing surface of the analyzed voxel-based object. An enclosing tree is represented by means of a chain of base-five digit strings suitably combined by means of parentheses. In order to describe enclosing trees, we use the tree descriptor which was presented in [1]. Tree structures cover a wide variety of applications [2]. In the content of this work, we use volumetric representations for rigid solids by means of spatial occupancy arrays [3]. The solids are represented as 3D arrays of voxels which are marked as filled with matter. Spatial occupancy arrays are very common in computer-aided tomography [3]. The enclosing trees describe trees of maximum degree six in three dimensions. This is due to the fact that we exclusively consider face-connected voxels, i.e. voxels with *six-connectivity.*

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 [2]. However, in the proposed method for representing voxel-based objects via enclosing trees, the topology and geometry of 3D tree structures should be preserved. The enclosing-tree notation is invariant under *rotation* and *translation.* Furthermore, using this notation it is possible to obtain the *mirror image* of any voxel-based object with ease. The enclosing-tree notation preserves the *shape* of voxel-based objects, allows us to know their topological and geometrical properties. The proposed method for representing the topology and geometry of voxel-based objects using enclosing trees 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 enclosing trees. Section 3 gives the method for generating enclosing trees. Section 4 describes some properties of the enclosing trees. Section 5 presents some results using objects of the real world as examples.

Finally, in Section 6 we present some conclusions and discussions.

2 CONCEPTS AND DEFINITIONS

In order to introduce our proposed method of enclosing trees, a number of concepts and definitions are presented below:

- An important consideration is the assumption that an *entity* has been isolated from the real world. This is called the *object*, and is defined as a result of previous processing [4, 5].
- *Shape* is referred to as shape-of-object, and an object is considered to be a geometric entity composed of voxels.
- *Area* is a numerical value expressing 2D extent in a plane, but sometimes it is used to mean the interior region itself [6].
- The term *sphere* means the enclosing surface together with its interior.
- The *area* of every face of each voxel is considered equal to one.
- The *length* l of each edge of voxels is considered equal to one.
- There are three ways of connecting voxels: by edges, vertices, and faces. In the content of this paper, we only consider face-connected voxels, i.e. voxels with *six-connectivity.*
- Volumetric representations are used for rigid solids by means of spatial occupancy arrays. Thus, the solids are represented as 3D arrays of voxels which are marked as filled with matter.
- The *area of the enclosing surface* A of a rigid solid composed of a finite number n of voxels, corresponds to the sum of the areas of the external plane polygons of the voxels which form the visible faces of the solid.
- The *tree descriptor* of a 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 will be a useful tool to represent the proposed enclosing trees. A *graph* is composed of a set of points called *vertices* v, joined by *edges* (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 [2]. 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.

$$e = v - 1. \tag{1}$$

The *degree* of a vertex in a tree is the number of edges incident to it. The enclosing trees describe trees of maximum degree six in three dimensions. This is due to the fact that enclosing trees are only represented voxel-based objects, we exclusively consider face-connected voxels.

DEFINITION 1. *A chain a is an ordered sequence of n elements, and is represented by*

$$a = a_1 a_2 \dots a_n = \{a_i : 1 \leq i \leq n\}. \tag{2}$$

DEFINITION 2. *Branches of trees are composed of constant straight-line segments (the 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]. 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) summarizes the rules for labeling the vertices: to a straight-angle vertex, a “0” is attached; to a right-angle vertex corresponds one of the other labels, 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 Fig. 1(a)), and the side from the vertex to be labeled has direction d (from here on, by direction, we understand a vector of length 1), then the chain element or label is given by the following function,

$$\textit{chain element}(b, c, d) = \begin{cases} 0, & \text{if } d = c; \\ 1, & \text{if } d = b \times c; \\ 2, & \text{if } d = b; \\ 3, & \text{if } d = -(b \times c); \\ 4, & \text{if } d = -b; \end{cases} \tag{3}$$

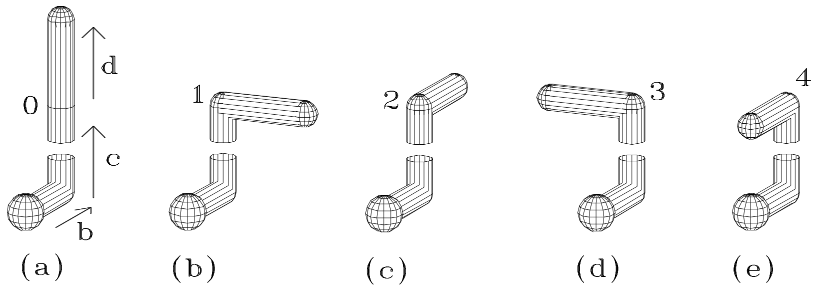
where $\times$ denotes the vector product in $\Re^3$.

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

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

(ii) *Compute* the chain elements of the tree. Fig. 1(f) shows the first element of the chain which corresponds to the chain element “3”. Note that the first direction change (which is composed of two contiguous straight-line segments) is used only for reference. The second element corresponds to the chain element “3” too, this is shown in Fig. 1(f). The third element corresponds to the chain element “3” again. The fourth element corresponds to the chain element “1”. The fifth element corresponds to the chain element “0”. The sixth element corresponds to the chain element “0”, too. Thus, in this stage our chain is as follows: 333100.

(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 the Fig. 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 Fig. 1(f), there are only three possible ways represented by the chain elements “1”, “2”, and “4”. Note that when we are traveling around a branch, in order to obtain



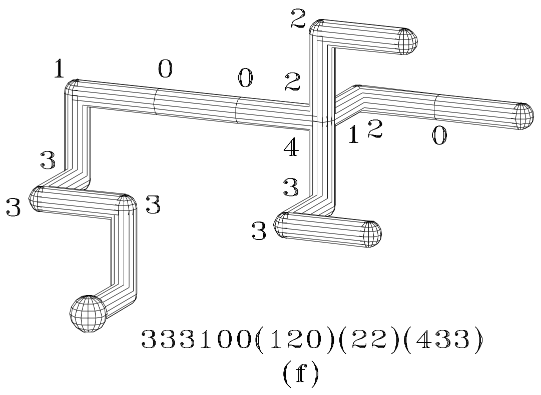


FIG. 1: An example of a tree on five vertices: (a) the chain element “0”; (b) the chain element “1”; (c) the chain element “2”; (d) the chain element “3”; (e) the chain element “4”; (f) the chain elements of an example of a tree and its tree descriptor.

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 the chain element “1” and we obtain 333100(120). The nested parenthesis describe the branch whose chain is (120). Then, we come back to the junction node and compute the chain of the next branch, which is (22), in this stage the tree descriptor is equal to 333100(120)(22). Finally, we come back to the junction node and compute the chain of the last branch, which is (433). So, the tree descriptor of the tree shown in Fig. 1(f) is as follows:

$$333100(120)(22)(433)$$

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

3 THE ENCLOSING TREES

The enclosing trees are trees which totally cover voxel-based objects and are represented by means of tree descriptors. In this manner, we only have a one-dimensional representation by each voxel-based object via a chain of base-five digit strings suitably combined by means of parentheses. The vertices of the enclosing tree correspond to the vertices of the enclosing surface of the analyzed voxel-based object. Topological and geometrical properties of voxel-based objects may be obtained from enclosing trees.

3.1 How to construct enclosing trees of voxel-based objects using the tree descriptor

The method to find enclosing trees of voxel-based objects using the tree descriptor is as follows:

(i) *Select* an arbitrary vertex of a voxel of the voxel-based object as the origin of the enclosing tree. In the generation of the enclosing tree, the selection of the origin is very important because different origins produce different enclosing trees for the same voxel-based object. Fig. 2(a) illustrates an example of a voxel-based object which is composed of only one voxel. Fig. 2(b) shows an arbitrary selected origin (which is represented by a sphere) as the beginning of the enclosing tree.

(ii) In order to construct the enclosing tree, *select* an arbitrary direction to define the chain elements. The first direction change is composed of two contiguous straight-line segments which are used only for reference. Different directions generate different enclosing trees for the same voxel-based object. Fig. 2(c) shows the arbitrary selected direction.

(iii) *Compute* the chain elements of the enclosing tree. We found a junction node (see figures 2(d) and (g)). We always select the chain elements in numerical order. Thus, in this junction node there are only two possible ways (following the enclosing surface) which correspond to the chain elements “3” and “4”, respectively. This is considered as the first stage of the enclosing tree and its tree descriptor is as follows: (3)(4).

(iv) *Compute* the chain elements which correspond to the second stage of the enclosing tree. We come back for the node which was generated by the chain element “3” and

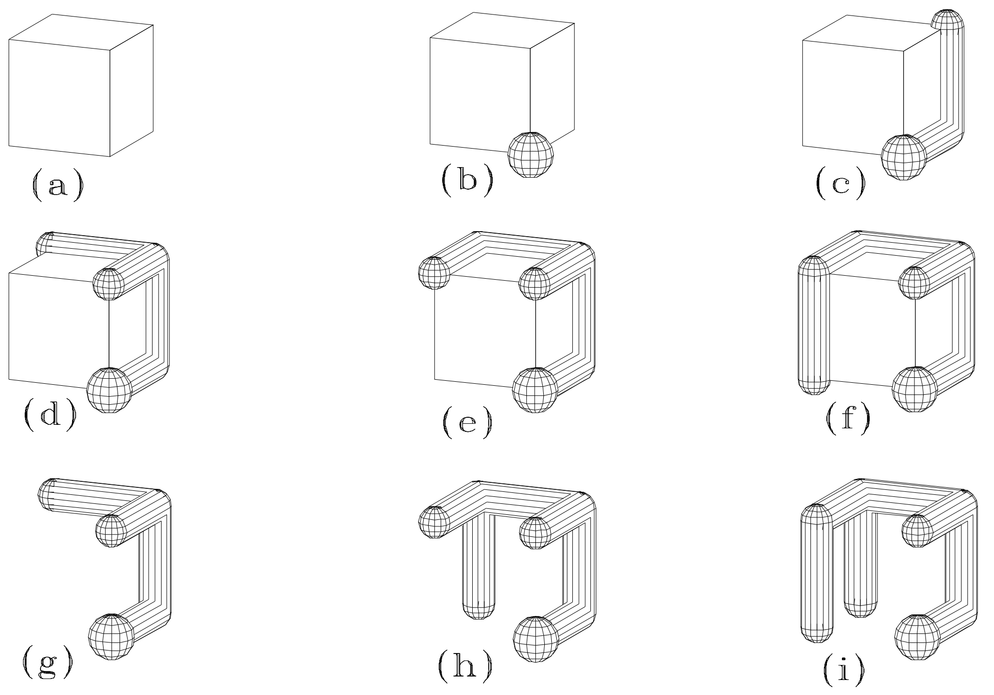


FIG. 2: An example of a voxel-based object which is composed of only one voxel and its enclosing tree: (a) an example of a voxel-based object; (b) a selected origin; (c) a selected direction; (d) the first stage of the enclosing tree and the voxel-based object; (e) the second stage of the enclosing tree and the object; (f) the enclosing tree of the above-mentioned voxel-based object; (g) the first stage of the enclosing tree; (h) the second stage of the enclosing tree; (i) the enclosing tree of the above-mentioned example.

we have only two possible ways represented by the elements "1" and "4". Now, we come back for the node which was generated by the chain element "4" and compute the chain elements (see figures 2(e) and (h)). Note that in this case the two possible ways were previously occupied by the growing enclosing tree, therefore chain elements are not generated. In this stage the tree descriptor is as follows: (3(1)(4))(4).

(v) *Repeat* the process until all vertices of the enclosing surface be computed. Figures 2(f) and (i) show the final stage of the enclosing tree, and its tree descriptor which is as follows: (3(1(3))(4))(4). In this example, we have finished the computation of all the vertices of the enclosing surface of the above-mentioned voxel-based object.

3.2 Examples of enclosing trees

Fig. 3 shows an example of an enclosing tree of a voxel-based object composed of 18 voxels. Fig. 3(a) illustrates the above-mentioned voxel-based object. Fig. 3(b) presents the origin and direction which were arbitrarily selected and the first stage of the enclosing tree. In this stage, the tree descriptor is as follows:

(0)(2)(4)

For the second stage (see Fig. 3(c)):

(0(0)(2)(4))(2(0)(4))(4(3))

For the third stage (see Fig. 3(d)):

(0(0(2)(3)(4))(2(0))(4(3)))(2(0(1)(4))(4(1)))(4(3(0)(1)))

For the fourth stage (see Fig. 3(e)):

(0(0(2(0)(1))(3(0)(1))(4))(2(0(1)))(4(3(0))))

(2(0(1(0)(3))(4))(4(1(0)(3))))(4(3(0(4)(1))(1)))

For the fifth stage (see Fig. 3(f)):

(0(0(2(0(1))(1(0)))(3(0(1)(4))(1))(4))(2(0(1(0))))(4(3(0))))

(2(0(1(0(3)(4))(3))(4))(4(1(0(3))(3))))(4(3(0(4)(1))(1)))

And for the sixth stage (see Fig. 3(g)):

(0(0(2(0(1(0)))(1(0(3))))(3(0(1)(4))(1))(4))(2(0(1(0))))(4(3(0))))

(2(0(1(0(3)(4))(3))(4))(4(1(0(3))(3))))(4(3(0(4)(1))(1)))

Fig. 4 shows the different stages of the enclosing tree of the voxel-based object presented in Fig. 3(a). Fig. 4(a) illustrates the first stage of the enclosing tree; Fig. 4(b) the second stage; Fig. 4(c) the third stage; Fig. 4(d) the fourth stage; Fig. 4(e) the fifth stage; and Fig. 4(f) the sixth stage, respectively.

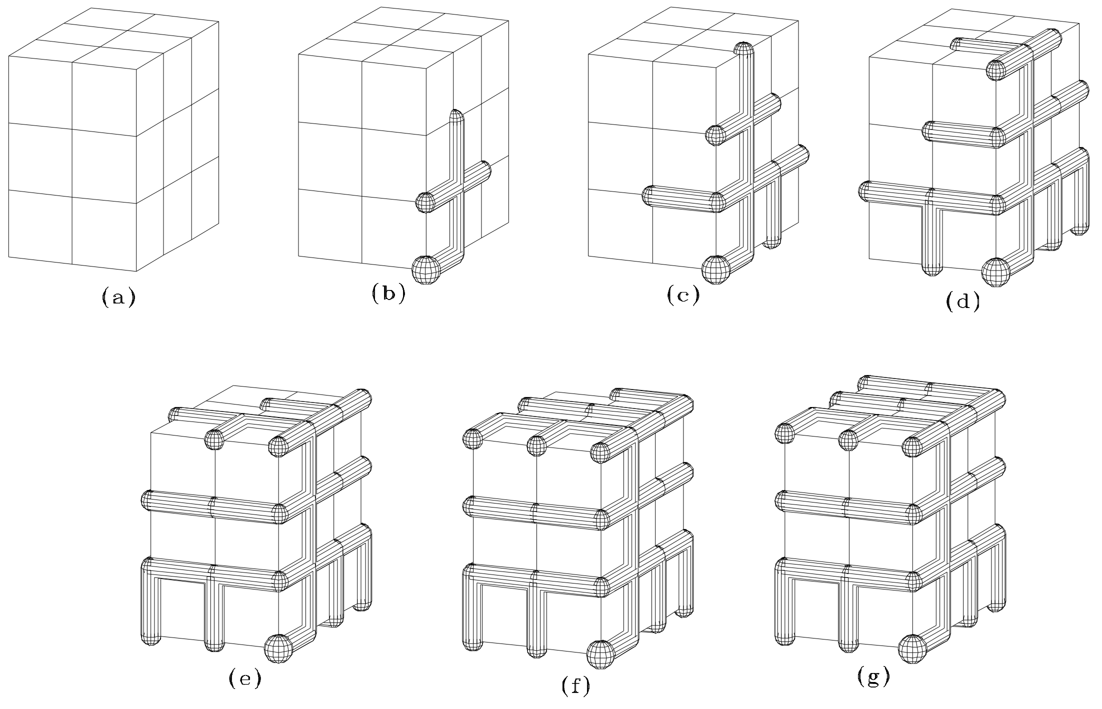


FIG. 3: An example of a voxel-based object which is composed of 18 voxels and its enclosing tree: (a) an example of a voxel-based object; (b) a selected origin and the first stage of the enclosing tree and the voxel-based object; (c) the second stage of the enclosing tree and the voxel-based object; (d) the third stage; (e) the fourth stage; (f) the fifth stage; (g) the sixth and final stage.

Fig. 5 shows another example of an enclosing tree of a voxel-based object composed of 8 voxels. Fig. 5(a) illustrates the above-mentioned object. Fig. 5(b) illustrates the origin and direction which were arbitrarily selected. Also, Fig. 5(b) shows the first stage of the enclosing tree and the above-mentioned voxel-based object. Figures 5(c)-(f) present the following stages of the enclosing tree of the voxel-based object shown in Fig. 5(a). In order to improve the understanding of the above-mentioned example, we have generated the Fig. 6 which shows the images of the Fig. 5 rotated 180°.

Finally, Fig. 7 shows the different stages (from Fig. 7(a) to (e)) of the enclosing tree presented in Fig. 5(a). The tree descriptor of the different stages of the enclosing tree is as follows:

$$(0)(1)(2)(3)(4)$$

$$(0(0)(1)(2)(3)(4))(1(3))(2(1))(3(0)(4))(4(0)(3))$$

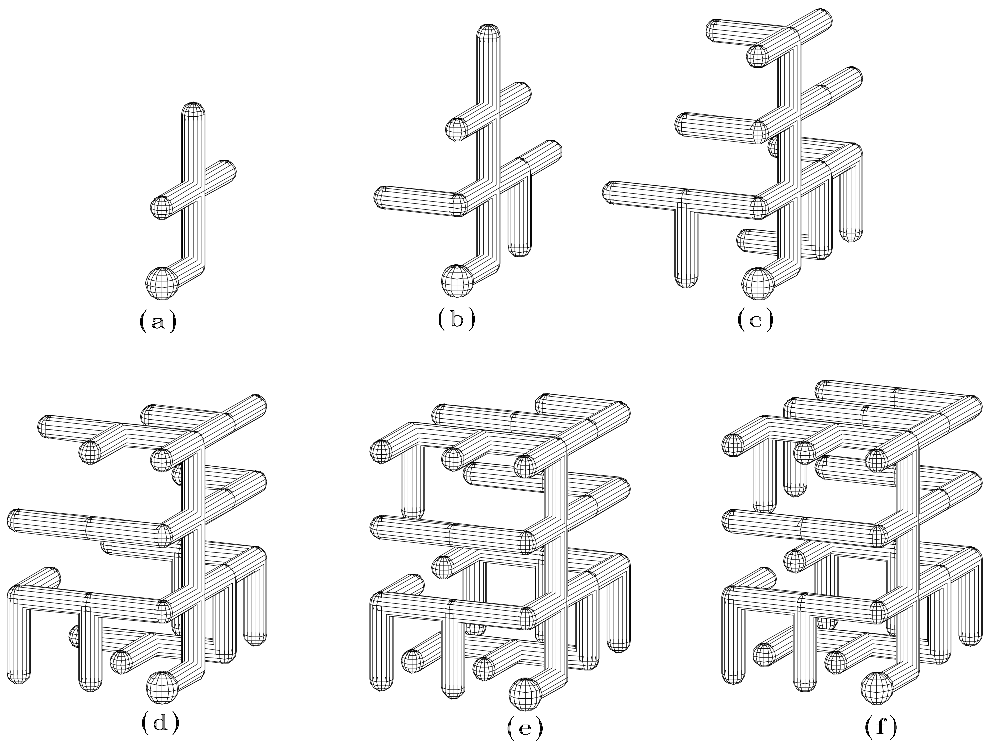


FIG. 4: The different stages of the enclosing tree of the voxel-based object shown in Fig. 3(a): (a) the first stage of the enclosing tree; (b) the second stage; (c) the third stage; (d) the fourth stage; (e) the fifth stage; (f) the sixth and final stage.

(0(0(3)(4))(1(3))(2(1))(3(0)(1))(4(0)))

(1(3))(2(1))(3(0(1))(4))(4(0(3))(3(1)))

(0(0(3(1))(4))(1(3))(2(1))(3(0(1))(1(0)))(4(0)))

(1(3))(2(1))(3(0(1(0)))(4))(4(0(3))(3(1)))

(0(0(3(1))(4))(1(3))(2(1))(3(0(1(0)))(1(0)))(4(0)))

(1(3))(2(1))(3(0(1(0)))(4))(4(0(3))(3(1)))

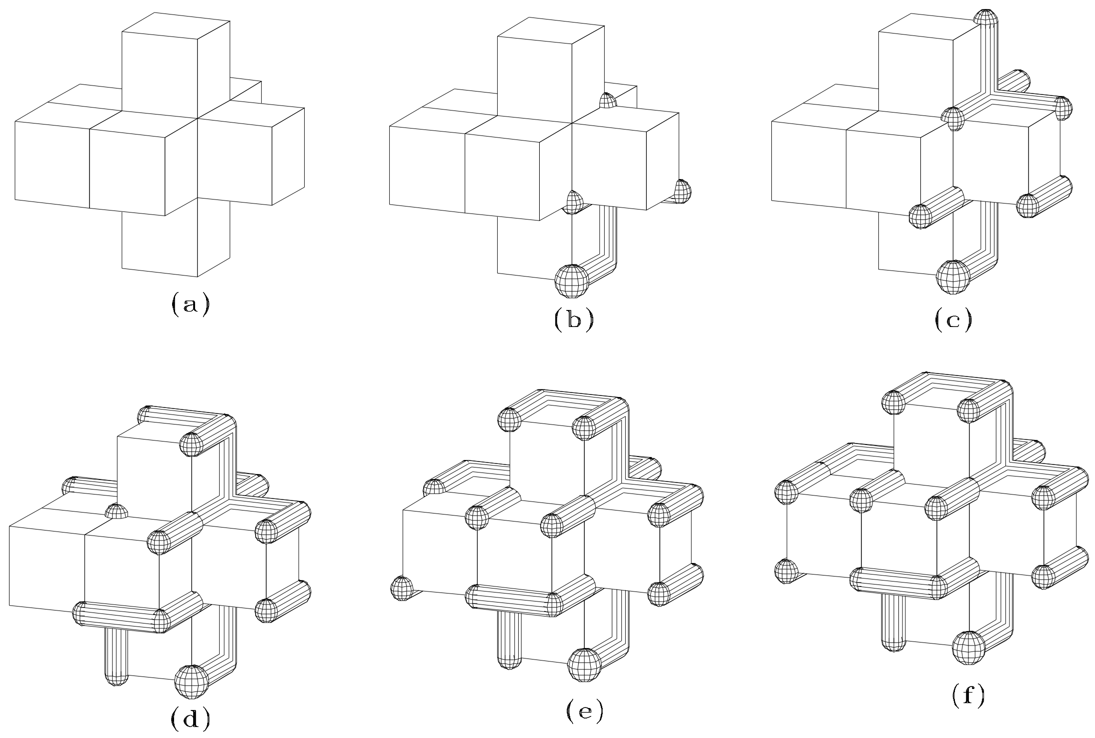


FIG. 5: Another example of a voxel-based object which is composed of 8 voxels and its enclosing tree: (a) another example of a voxel-based object; (b) the selected origin and direction, the first stage of the enclosing tree and the voxel-based object; (c) the second stage of the enclosing tree and the voxel-based object; (d) the third stage; (e) the fourth stage; (f) the fifth and final stage.

4 SOME PROPERTIES OF THE ENCLOSING TREES

The enclosing-tree notation preserves the *shape* of voxel-based objects, allows us to know their geometrical and topological properties.

4.1 Independence of rotation for enclosing trees

The enclosing-tree notation is invariant under rotation. Fig. 8 shows different rotations of the voxel-based object and its enclosing tree illustrated in Fig. 5(f). Fig. 8(a) presents a rotation on the axis "X" of the voxel-based object and its enclosing tree shown in Fig. 5(f) and its tree descriptor is equal to:

$$(0(0(3(1))(4))(1(3))(2(1))(3(0(1(0)))(1(0)))(4(0)))$$

$$(1(3))(2(1))(3(0(1(0)))(4))(4(0(3))(3(1)))$$

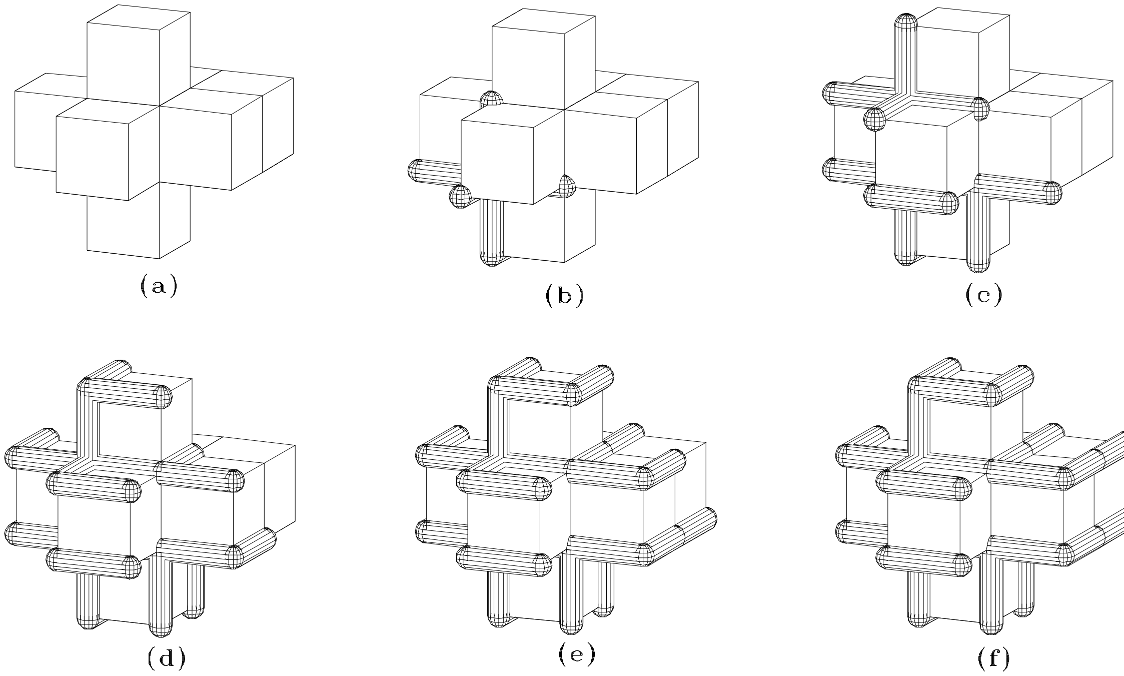


FIG. 6: The example of the voxel-based object and its enclosing tree shown in Fig. 5 rotated 180°: (a) the voxel-based object shown in Fig. 5(a) rotated 180°; (b) the voxel-based object and the enclosing tree shown in Fig. 5(b) rotated 180°; (c) in Fig. 5(c) rotated 180°; (d) in Fig. 5(d) rotated 180°; (e) in Fig. 5(e) rotated 180°; (f) in Fig. 5(f) rotated 180°, respectively.

Fig. 8(b) illustrates a rotation on the axis "Y" of the voxel-based object and its enclosing tree presented in Fig. 5(f) and its tree descriptor is as follows:

$$(0(0(3(1))(4))(1(3))(2(1))(3(0(1(0)))(1(0)))(4(0)))$$

$$(1(3))(2(1))(3(0(1(0)))(4))(4(0(3))(3(1)))$$

Fig. 8(c) shows a rotation on the axis "Z" of the voxel-based object and its enclosing tree presented in Fig. 5(f) and its tree descriptor is equal to:

$$(0(0(3(1))(4))(1(3))(2(1))(3(0(1(0)))(1(0)))(4(0)))$$

$$(1(3))(2(1))(3(0(1(0)))(4))(4(0(3))(3(1)))$$

Fig. 8(d) illustrates the enclosing tree of the voxel-based object shown in (a). Fig. 8(e) presents the enclosing tree of the object illustrated in (b). Fig. 8(f) shows the enclosing

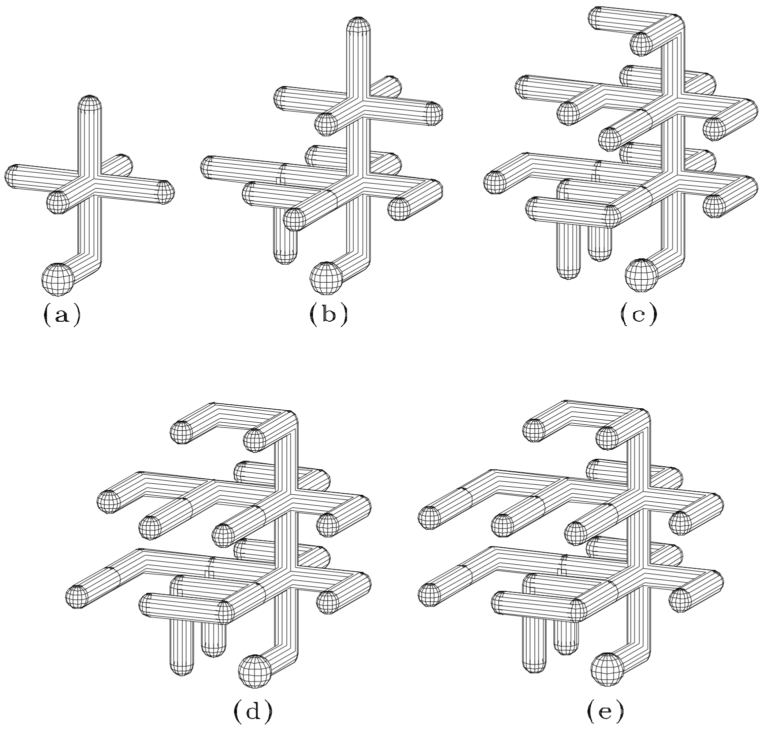


FIG. 7: The different stages of the enclosing tree of the voxel-based object shown in Fig. 5(a): (a) the first stage of the enclosing tree; (b) the second stage; (c) the third stage; (d) the fourth stage; (e) the fifth and final stage.

tree of the object presented in (c). Notice that all tree descriptors of the enclosing trees are equal. So, they are invariant under rotation.

4.2 Mirror images of enclosing trees

The mirror image of any enclosing tree is obtained with ease.

DEFINITION 4. *Let a be a tree descriptor of an enclosing tree, mirror a is the descriptor obtained by replacing in a each occurrence of* 1 *by* 3 *and vice versa.*

THEOREM 1. *Let a be a tree descriptor, then mirror (mirror a) = a*

Considering Definition 4 and Theorem 1 for tree descriptors: the descriptor of the mirror image of an enclosing tree is another descriptor (termed *mirroring descriptor*) whose elements “1” are replaced by elements “3” and vice versa, preserving its nested parentheses. Fig. 9 shows the mirror images of the enclosing tree shown in Fig. 7(e). In Fig. 9(a) the mirroring

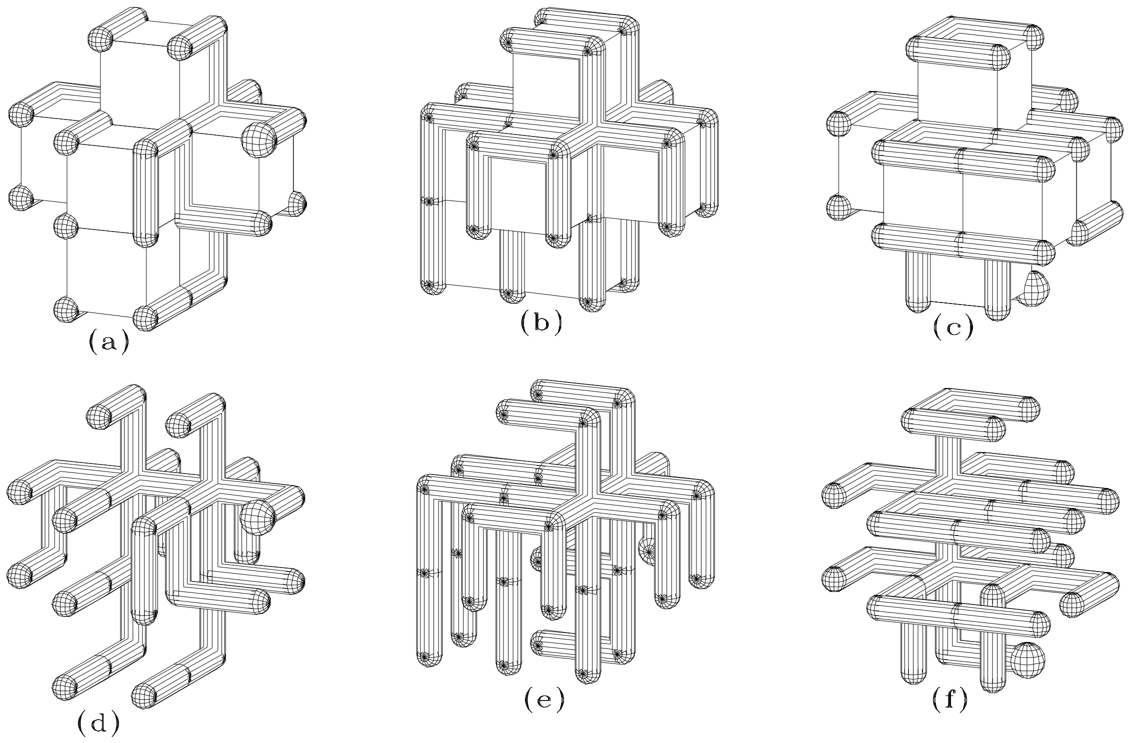


FIG. 8: Independence of rotation: (a) a rotation on the axis "X" of the voxel-based object and its enclosing tree shown in Fig. 5(f); (b) a rotation on the axis "Y" of the voxel-based object and its enclosing tree illustrated in Fig. 5(f); (c) a rotation on the axis "Z" of the voxel-based object and its enclosing tree presented in Fig. 5(f); (d) the enclosing tree of the voxel-based object shown in (a); (e) the enclosing tree of the object illustrated in (b); (f) the enclosing tree of the object presented in (c).

plane is aligned with the standard plane "XY". In Fig. 9(b) the mirroring plane is aligned with the plane "XZ" and in (c) with the plane "YZ", respectively. Thus, the tree descriptor of the enclosing tree presented in Fig. 7(e) is as follows:

$$(0(0(3(1))(4))(1(3))(2(1))(3(0(1(0)))(1(0)))(4(0)))$$

$$(1(3))(2(1))(3(0(1(0)))(4))(4(0(3))(3(1))).$$

And its mirroring descriptor is equal to:

$$(0(0(1(3))(4))(3(1))(2(3))(1(0(3(0)))(3(0)))(4(0)))$$

$$(3(1))(2(3))(1(0(3(0)))(4))(4(0(1))(1(3)))$$

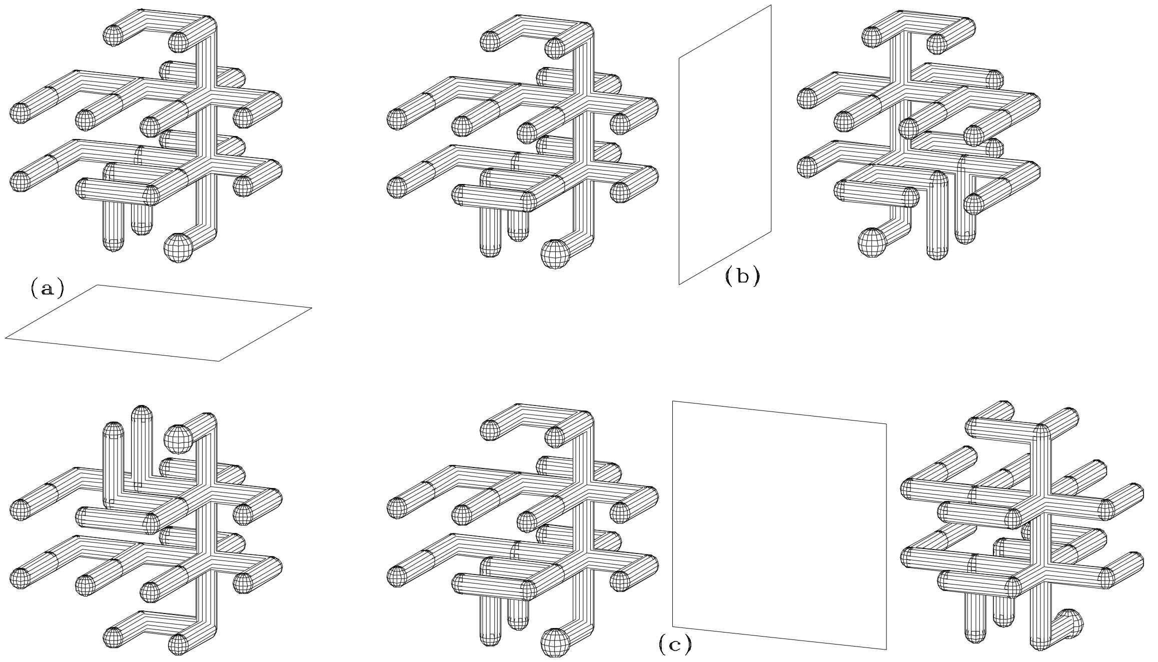


FIG. 9: Mirror images of the enclosing tree shown in Fig. 7(e): (a) the mirroring plane is aligned with the standard plane "XY"; (b) with the plane "XZ"; and (c) with the plane "YZ", respectively.

Notice that the 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 the enclosing trees, we have not sorted the parentheses (in the mirroring descriptor) in numerical order as it was mentioned in Subsection 2.1.

4.3 Some geometrical properties of enclosing trees

The vertices of the enclosing tree correspond to the vertices of the enclosing surface of the analyzed voxel-based object. This means that the enclosing tree representation is a lossless transformation of an object, since its surface is exactly described by all its vertices, and these are also exactly described by the tree.

DEFINITION 5. *Let a be a tree descriptor of an enclosing tree and m the number of vertices of an enclosing surface S of a voxel-based object. Then, $n = m - 3$, where n is the number of chain elements of the tree descriptor.*

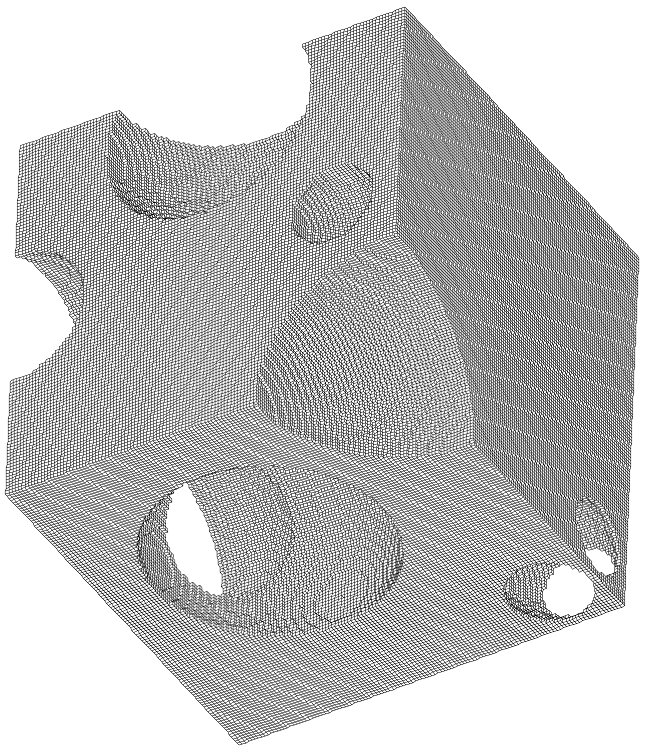


FIG. 10: An example of a voxel-based object which has several holes. It is possible to obtain the enclosing tree for this kind of objects.

The Definition 5 is due to the fact that the first three points of the tree descriptor correspond to the two contiguous straight-line segments which are only considered for reference.

The enclosing trees can be obtained for voxel-based objects which may have holes. For instance, Fig. 10 presents an example of a voxel-based object which has several holes, it is possible to find the enclosing tree of this kind of object. However, when objects have inner holes it is necessary to find the enclosing tree for each hole, this is due to the fact that this kind of objects have several inner enclosing surfaces. Note that each enclosing surface corresponds to an enclosing tree.

4.4 Compression efficiency of the enclosing trees to describe voxel-based objects

The encoding efficiency to represent voxel-based objects is very important for pattern recognition and computer vision. The proposed method of enclosing trees is a powerful tool for storing of voxel-based objects. This enclosing-tree notation may be used for lossless compression of voxel-based objects because it preserves information and allows considerable

Plane of growth	+ 0	+ 1 or + 3	+ 2 or + 4
x y	x y	y z	y x
y x	y x	x z	y x
x z	x z	z y	z x
z x	z x	x y	x z
y z	y z	z x	z y
z y	z y	y x	y z

TABLE I: Equations for the plane of growth of a tree.

data reduction. However, fixed-length coding is not fair in comparison to the variable-length Huffman codes. The well-known Huffman code is a widely used compression method that relies on the concept of entropy. So, it is necessary to apply Huffman method to compress enclosing trees by obtaining probability of appearance of each symbol. In this manner, it is possible to obtain a good compression rate.

4.5 Equations for changing the plane of growth of a tree

The "plane of grow of a tree" is defined as the plane of the "handle". For instance, for Fig. 11, the plane of the handle is $x - y$; hence, this is also the plane of growth of a tree. The plane of growth does not change as long as we keep adding 0s, 2s and 4s to the tree. (To be rigorous, a 2 changes the plane $a - b$ into the plane $b - a$, and a 4 changes $a - b$ also into $b - a$. But for most purposes, planes $a - b$ and $b - a$ are the same). By adding 1s or 3s, the plane of growth changes; these changes are shown in Table I.

These equations can be represented by:

$$a\ b + 0 = a\ b \tag{4}$$

$$a\ b + 1/3 = b\ c \tag{5}$$

$$a\ b + 2/4 = b\ a. \tag{6}$$

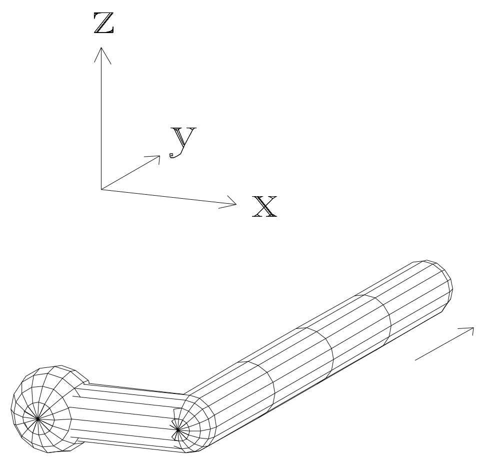


FIG. 11: The initial plane of grow of the tree is x y because the handle lies in that plane.

Where a, b and c represent three different axes, 1/3 represents "1 or 3" and "2/4" stands for "2 or 4". With them, it is easy to know in each voxel the plane of growth of a given (portion of a) tree.

5 RESULTS

5.1 Hamiltonian representation of vox-solids

Other method for describing voxel-based objects corresponds to the Hamiltonian representation of vox-solids [10]. Thus, a vox-solid is defined as a solid digitalized into voxels whose boundary is a surface. The face adjacency graph embedding of a voxel-based object is the graph whose vertices are the boundary faces of voxels, and the edges relate pairs of boundary faces sharing exactly one side. The main conjecture of this representation says that every face adjacency graph is Hamiltonian.

Fig. 12(a) shows an example of a voxel-based object composed of eight voxels. Fig. 12(b) presents the above-mentioned object and its enclosing tree. Fig. 12(c) illustrates the enclosing tree of the object shown in (a). Finally, Fig. 12(d) presents the tree descriptor of the object presented in (a). Fig. 13 presents an example of a vox-solid (Fig. 13(a)) and its face adjacency graph. Fig. 13(b) shows the labels of the visible faces of the above-mentioned vox-solid and in (c) the face adjacency graph is shown.

5.2 Comparison between the proposed tree descriptor and the Hamiltonian representation of vox-solids

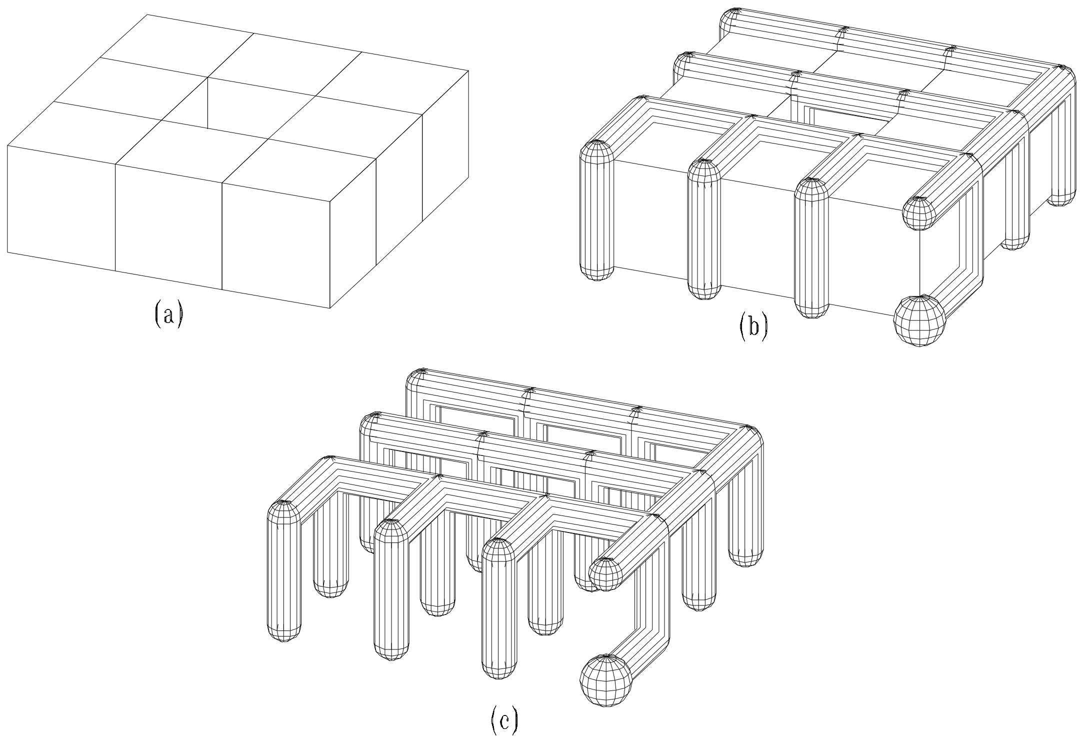
The enclosing-tree notation preserves the geometry of voxel-based objects. The face adjacency graph does not preserve the shape of vox-solids, two different objects may have the same face adjacency graph. Other difference: the proposed enclosing-tree notation is a good tool for storing of voxel-based objects, the tree descriptor is represented by a chain of base-five digit strings suitably combined by means of parentheses, it is a one-dimensional notation. On the other hand, the face adjacency graph is represented by a matrix, i.e. a two-dimensional array of numbers.

5.3 Examples of some voxel-based objects of the real world

In order to test our proposed enclosing-tree method, we present some voxel-based objects of the real world as examples. In this case, we analyzed Digital Elevation Model (DEM) data. DEMs are digital representations of the earth's surface. Generally speaking a DEM is generated as a uniform rectangular grid organized in profiles. In order to analyze DEM data by means of the proposed enclosing-tree notation, the models are represented as binary solids composed of regular polyhedrons (voxels).

DEMs have a large number of applications [11]. Some of these applications include: urban and regional planning, production of slope, aspect, hill shaded maps, engineering calculations, geomorphology, navigation, line-of-sight calculations, components in complex models, and geographic information systems. Thus, the importance of new representations on DEM data is evident.

In this work the DEMs are represented as three-dimensional (3D) arrays of cells (voxels) which are marked as filled with matter [3]. Several authors have been using different kinds of representations for solids: constructive solid geometry schemes are presented in refs [12]



(2(0(1(0(0(3))(3))(3))(4))(1(0(0(3))(3))(3))(4))(3(0(0(1(3))(4))(1(3))(4))(1(3))(4))(4)

(d)

FIG. 12: An example of a voxel-based object and its enclosing tree: (a) An example of an object composed of eight voxels; (b) the voxel-based object shown in (a) and its enclosing tree; (c) the enclosing tree of the above-mentioned object; (d) the tree descriptor of the object presented in (a).

and [13]; generalized cylinders as 3D volumetric primitives are shown in refs [14], [15], and [16]; rigid solids represented by their boundaries or enclosing surfaces are shown in refs [17] and [18]; and superquadrics in ref. [19].

Fig. 14 illustrates the DEM of “El Valle de México” (Valley of Mexico City). The left-hand side of the figure corresponds to the north of “El Valle de México”, the right-hand side to the south, the upper side to the east and the lower side to the west, respectively. On the left-hand side there is a group of hills called “La Sierra de Guadalupe”, and on the right-hand side there are more hills which correspond to “El Ajusco”. The elevation data values of the models presented in this study were increased to enhance their characteristics.

Fig. 15 shows the voxelized versions of the volcanoes: “La Malinche” and “Popocatepetl”. Fig. 16 illustrates the upper side of the volcano “La Malinche” and its enclosing tree. In this case, we have only considered seven layers of voxels of the volcanoes. In order to improve

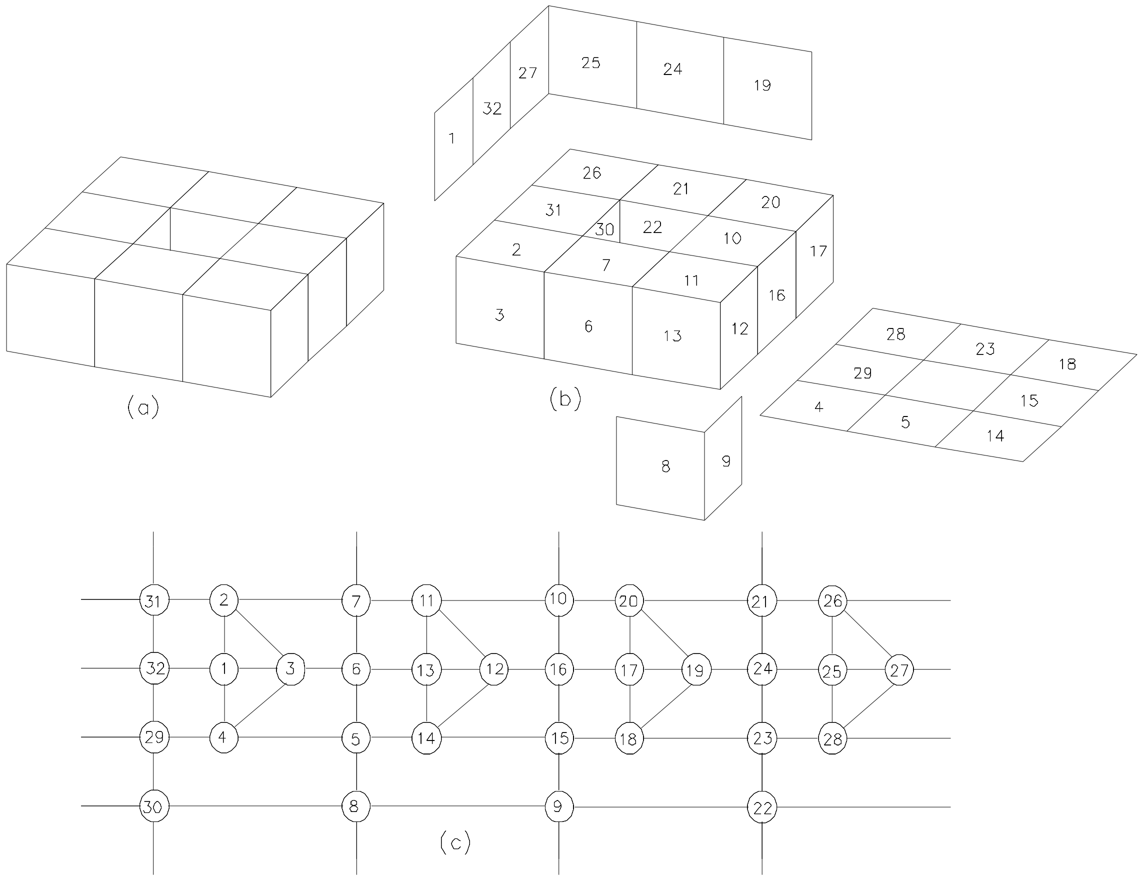


FIG. 13: An example of a vox-solid and its face adjacency graph: (a) an example of a vox-solid composed of eight voxels which corresponds to the voxel-based object shown in Fig. 12(a); (b) the labels of the visible faces of the above-mentioned vox-solid; (c) the face adjacency graph;

our examples 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. This may be observed on the electronic version of the paper. The tree descriptor of the upper side of the volcano "La Malinche" is as follows:

(3(1(0(2(0(2(0(1(1(0(0(2(0(2)(3(0(0(0(0(2(1))(3))(2(1))(3))(2(1)(4(1))))(3))(3)))(2)(3(0)
)))(1(0(1(0(2))(2(3(0)(2(2)))))(2(2(2))))(1))(2))(1)))(2(0(1(0(2(0)(2(2))(3(2))))(1(1(0)(
2(2))))(2))(2(0(2(0)(2(2))))(2)))(1)(2))(3))(1(1(0(0(2(0(2))(2)))(2))))(3(0(3(0(0(2(0(2))
(2)))(2))))(3(0))))(3(0(2(0(0(2(0(2))(2)))(2))))(2(0))))(1(0(2(0(0(2(0(2(0)(2(0(0(1(1)(2(
0(2(1))(3))(2(1)))))(1(1)))(1(1))(2))))(2))))(2(2))))(1)(2)))(3))(1(1(0(2(0(2(0(0(2(0(1(1
)(2(3)))(2))(1)(2)))(2))))(2(0))))(2)))(2)(3(0(3(0(2(0(0(0(1(1(0(1(0(1)(2(2(2))))(1)))(1)
))(2(2(2(2(2(2(0(2))(2))))))))(1(1(0)))(2(0(0))))(2(0(0))))(2(0))))(2)))(3)))(2(1(0(1(0(1

(0(0(1(1(0(1(0(0(2(2(1(1))(2))(3(0(2))(2))))(2(3(0(2)))))(1(1(2(0(2))(2)))(2)))(2(0(2(2(2
))))(2)))(1(0))(2)))(2(2(0(2(0(2(0(2))(2)))(2)))(2))))(1(1(2(2(2(2(2(2))))))))(1(1)))(2(
0(2(1(1(1(0(2(0(2))(2)))(2)))(2))(2(2(2(2(2(2)))))))(2)))(1(0))(2)))(1(1(0(2(0(2(0(2(0(2
(0(2))(2)))(1)(2)))(2)))(2)))(2))))(2(2(2(2(2(0(2))(2)))))))(3(0(2(0(2(0(2(2(1(1(0(1(0(1(
1))(2))(1(0(0(1))(1))(1)))(2(0(0(0(0(2))(2))(2)))))(1)(2(0))))(2(2(2(2(2))))))(3(0(2(2(0(
0(2(2(0(0(0(0(2))(2))(2))(2))(2))))(2))(2))))(2))))(1(0(1(0(2(0(0(0(0(0(0(2(2(2))))(2))(2
))(2))(2)))))(2(0))))(1))(2))(3(0(0(0(2(0(2(0(0(0(0(2(2(2)))))))))(2(0(0)))))(2))(1)(2)))
)(2)(3(0))))(1(0(1(2(0(2(0(2(0(0(0(2(2(2(2(2))))))(2)))))(2(0))))(2))))(1))(2))(4(0(2(2(0
(2(2(0(2(2(0(2(0(2(0(2))))(2)))(2)))))))(2))))))(4)

Fig. 17 shows the upper side of the volcano "Popocatepetl" and its enclosing tree. The tree descriptor of the upper side of the volcano "Popocatepetl" is as follows:

(0(2(0(0(1(0(1(1(1(0(1(0)(1(0(0)))(2(2(0(2(0)))(1(0(0(0(1(0)(1(2))))(1(1(2))))(1(1(2))))(
1(1(2))))(2))(3(0(0(0(2(0))))))))(2(0(0(2(0(2(2)))(2)))(2))))(1)(2)))(2(2(2(2(2(2(2(0))))
)))))(2(2(2(2(2(2(2(2(2(2))))))))))(2)))(1(0)))(1(0(1(1(1(1(2(2(2(2(2(0)))))))))(2(2(2(2
(2(2(2(0)))))))))))(3(0(0(1(0(0(0(0(2(2(1(1(0(1(1)(2))(2(2)))(1)(2)))(2(2(2(0)))))(3(2(2
(2(0)))))))(2(3(2(2(2(0)(2(2))))))))(1(1(0(2(0(2(0)))(2)))(2)))(2))(1(1(2(2(2(2(0)))))))
(1(1(2(2(2(2(0)(2(2)))))))))(2(0(2(0(2(0(2(0)))(2)))(2)))(2)))(1(0(0)))(2))(1))(4))(2(0(0
(1(0(0(2(0(1(1(0(1(0(2(0)(2(2))(3(2))))(2(3)))(2(0(0(2(0)(1(2))(2(2)))(2)(3(3(0)(2(2))))
)(3)))(1)(2(0))))(2(2(2(2(2(2(2))))))))(1(1(2(0(2(0(2(0(2))(2)))(2)))(2)))))(3(0(2(0(2(0(
2(0)(2(2))))(2)))(2)))(2)))(2)(3))(2)))(1(0(3(2(2(2(2(2(2(2(2(2)))))))))))))(1(0(3(2(2(2(
2(2(2(2(2(2)))))))))))))(3(0(4(1(0(0(0(1(1(1(1(1(1(1(0(1(1(1))(2))(2(2)))(1)(2)))(2(2(2(0
)))))(2(2(2(2(0)))))))(2(2(2(2(2(0)))))))(2(2(2(2(2(2(0))))))))(1(1(2(2(2(2(2(0))))))))
(1(1(2(2(2(2(2(0))))))))(1(1(2(2(2(2(2(0))))))))(2(2(2(2(2(2(2(0)(2(2)))))))))))(4(1(0)
)))(4(3(0(2(0(0(2(2(2(2(2(2(2(0)))))))))(2))))(2))))

Finally, Fig. 18 illustrates the enclosing trees of both volcanoes. Table II shows the comparison between the tree descriptor and the Hamiltonian representation of the above-mentioned volcanoes. The advantages of the enclosing trees are evident.

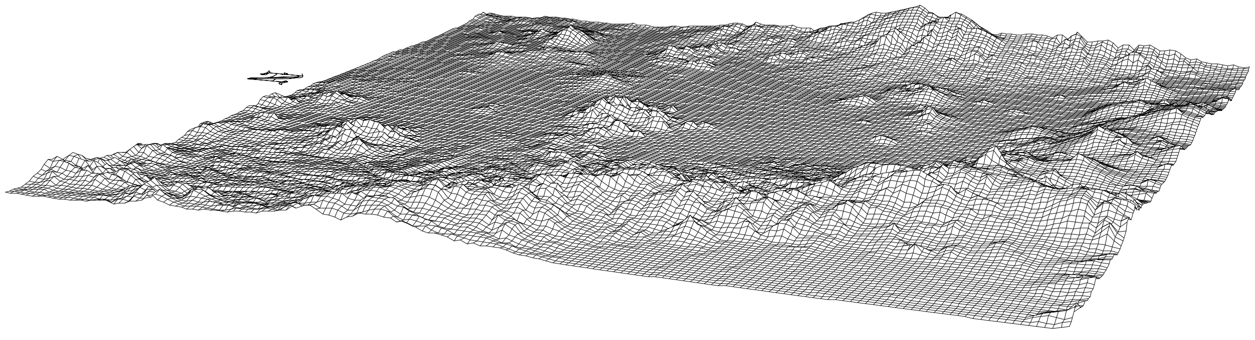


FIG. 14: The DEM of “El Valle de México” (Valley of Mexico City).

Volcano	Vertices of adjacency graphs	Elements of graph matrices	Elements of tree descriptors
La Malinche	436	190096	1386
Popocatepetl	350	122500	1121

TABLE II: Comparison between the tree descriptor and the Hamiltonian representation of the above-mentioned volcanoes.

6 CONCLUSIONS AND DISCUSSIONS

6.1 Conclusions

A method has been described for representing voxel-based objects by means of the enclosing trees which are represented by chains of base-five digit strings suitably combined by means of parentheses. Enclosing trees totally cover voxel-based objects, the vertices of the enclosing tree correspond to the vertices of the enclosing surface of the analyzed voxel-based object. The enclosing-tree notation preserves the *shape* of voxel-based objects, allows us to

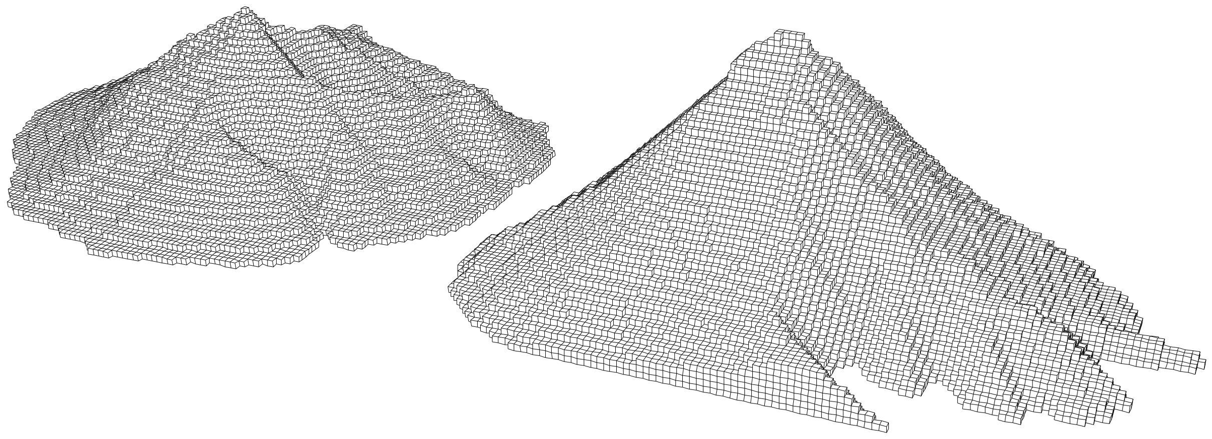


FIG. 15: The voxelized versions of the volcanoes: "La Malinche" and "Popocatepetl", respectively.

know their topological and geometrical properties. Also, it is possible to obtain enclosing trees of voxel-based objects with holes. The enclosing-tree notation is invariant under *rotation* and *translation*. The article also tells how to compute with ease the mirror image of any voxel-based object. Finally, the proposed enclosing-tree notation is a good tool for storing of voxel-based objects.

6.2 Discussions

6.2.1 Shape comparison

The enclosing trees just found are good to describe three dimensional objects. If we want to compare the shape of two of these objects through their respective enclosing trees, some additional normalizations are necessary.

- *Rotation of axes.* Subsection 4.1 shows that rotation of an object preserves its enclosing tree. Nevertheless, if the object is rotated with respect to the coordinate axes, then

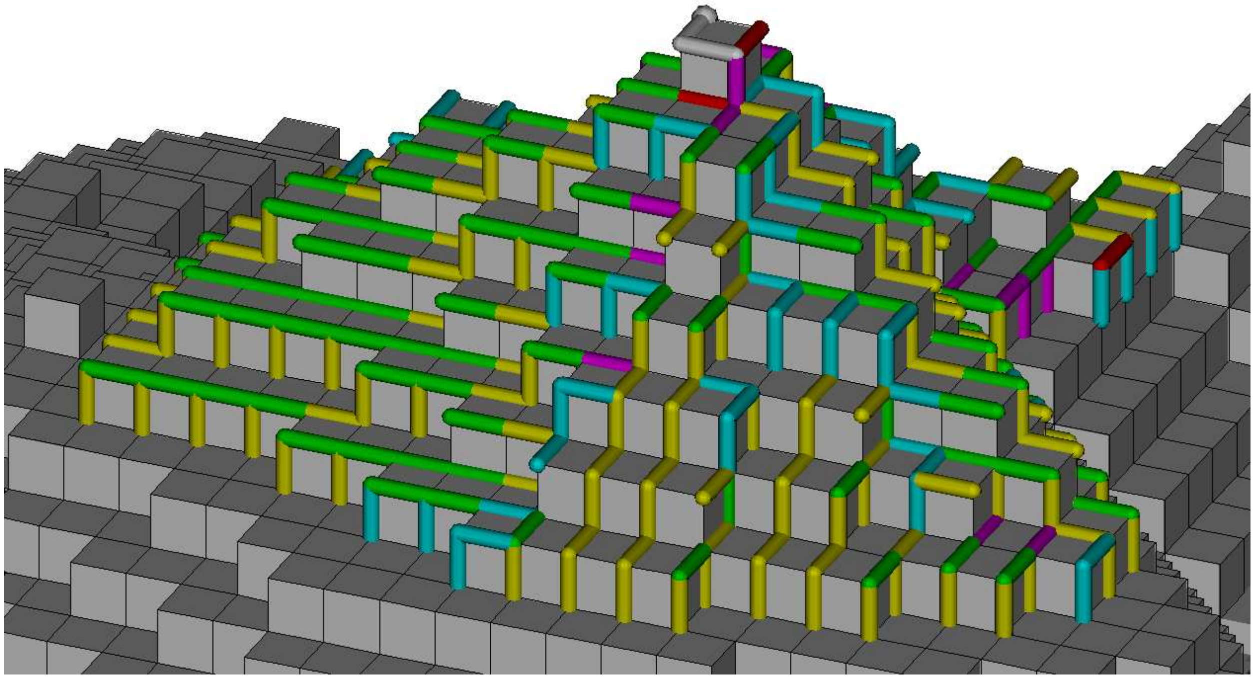


FIG. 16: The volcano “La Malinche” and its enclosing tree, we have only considered seven layers of voxels of the volcanoes.

the tree representation changes, in general. Thus, it is desirable to derive from the object itself the three orthogonal directions that define the axes. For instance, we can find the two voxels furthest apart, the axis passing through them could define the x axis [20]. A better way to find the three axes deducing them from the object itself is outlined in [21, 22]. In this way, rotating the object will make the axes rotate, too, and full invariance with respect to rotations is achieved.

- *Voxel normalization.* The enclosing tree of an object will change if we change the resolution (the size of a voxel). To normalize against size change, once the axes are found, we could enclose the object inside the bounding box that just encloses it, and declare that the sides of this enclosure are each of size 100, say. In this way the enclosing prism contains 1 million voxels while the object itself will occupy less. The size of the voxels shall be selected so as to capture all desired nuances (valleys, protuberances, holes) of the object, while keeping a reasonable number of

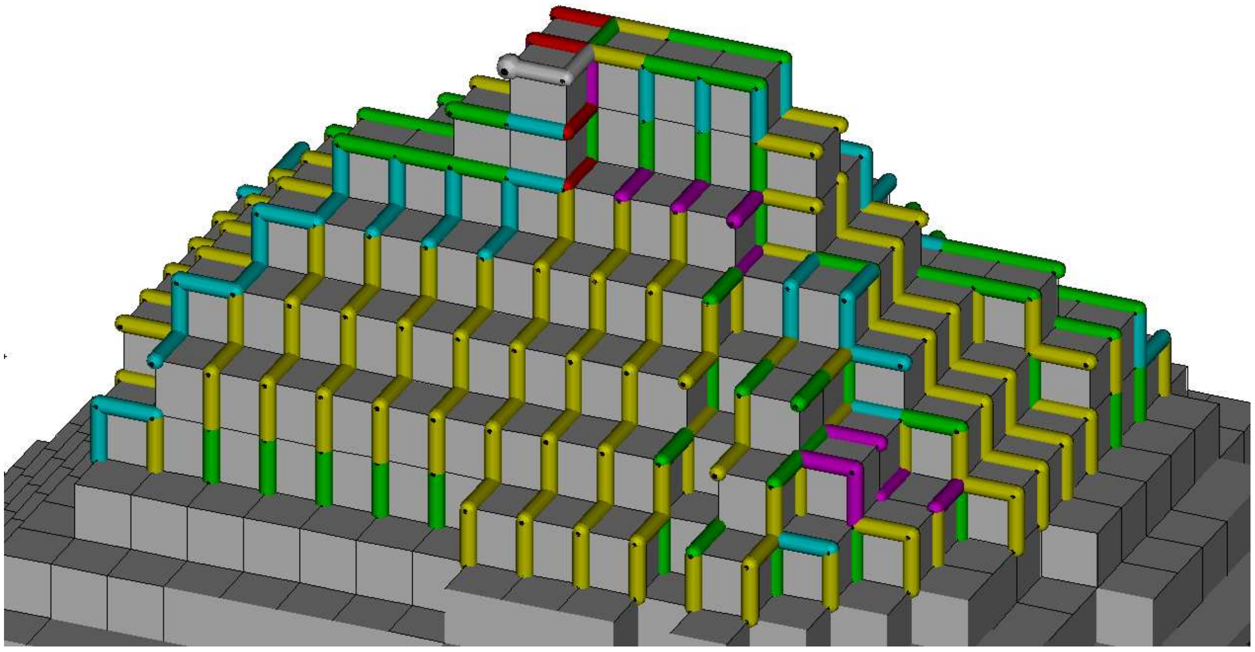


FIG. 17: the volcano “Popocatepetl” and its enclosing tree, we have only considered seven layers of voxels of the volcanoes.

voxels occupied by the object. See [20] for this technique for 2D objects.

- *A unique origin for the enclosing tree.* Changing the place where the tree starts growing (the “root” of the tree) has the effect of changing the tree representation. It is, therefore, desirable to locate in the object an origin that can be easily found if the object rotates or changes size. If possible, the origin location should also be somewhat invariant to small changes in the surface of the object. A good candidate for the root is the center of mass of the 3D object. Unfortunately, the center of mass will not be on the surface, in general. We propose as root of the tree the surface point closest to the center of mass.

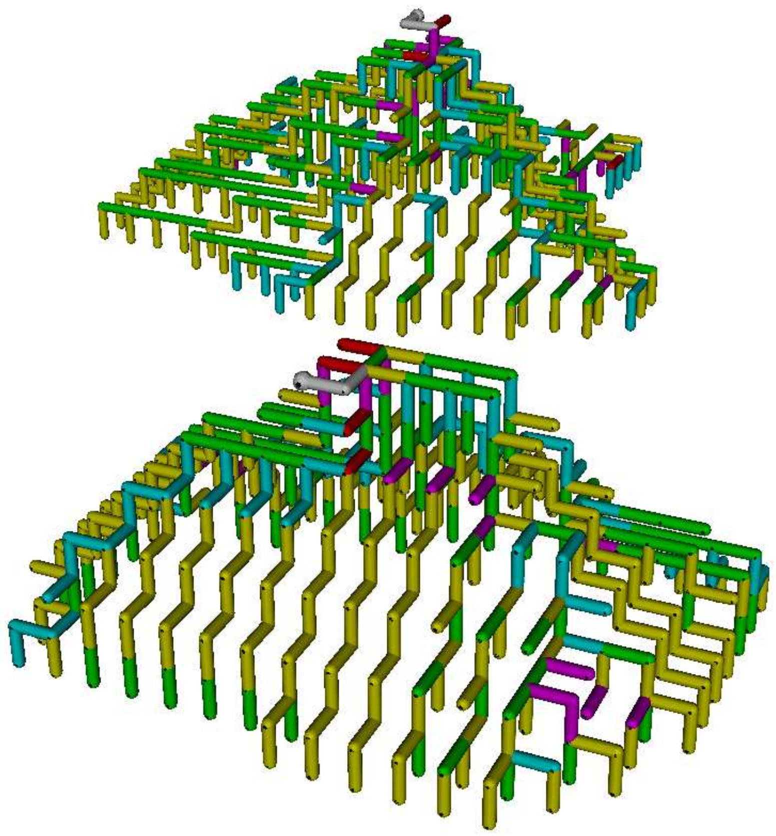


FIG. 18: The enclosing trees of the volcanoes “La Malinche” and “Popocatepetl”, respectively.

6.2.2 Detecting flat surfaces

A flat surface is spanned by trees that are of the form 0/2/4 -; 0/2/4 (an arbitrarily long combination of 0s, 2s and 4s). The surface is parallel to the plane $a - b$ (a is one of x, y, z and b is one of the remaining two axes). Remark. Although the “plane a b” and the “plane b a” are the same plane in space, notice that $a\ b + 2/4 = b\ a$. That is, growth when we had a b as the “handle” was going along the b axis, but the addition of a 2 (or a 4) makes the growth (still in the a b plane) to be now along the a axis.

6.2.3 Detecting surfaces that are considerably parallel to a given $a - b$ plane

An almost flat surface (which sometimes we call a planoid) is a surface that is for most of its area parallel to some plane (to plane $a - b$, to be general) and occasionally curves out of the plane, to return to it shortly thereafter. A tree that grows in the $a - b$ plane and then deviates from it has the form ($a - b-$ some 0, 2 or 4s 1/3 . . .) that is, $a - b-$ some 0s and

2s and 4s followed by a 1 or a 3.

- If a tree starts growing in the a b plane, an even number of 2s and 4s returns the growth to the a b plane, whereas an odd number of 2s and 4s returns the growth to the b a plane. In any case, 0s interspersed among 2s and 4s do not change.
- If a tree originally growing in the a b plane adds 0s, 2s and 4s, and then adds an 1 or a 3,
  - A. The plane of growth changes to the b c plane if the number of 2s and 4s (appearing before the 1 or the 3) is even (refer to Fig. 19);
  - B. But it changes to grow in the a c plane if the number of 2s and 4s (appearing before the 1 or the 3) is odd.

  To go back to grow in the (original) a b plane,

  1. For case A (even number of 2s and 4s), the tree needs two more numbers different from 0 (immersed in any quantity of 0s), the first must be a 1, 2, 3 or 4 (to produce a change from the b c to the c a plane), the second number must be a 1 or a 3. That causes the growth resume at plane a b.
  2. For case B, odd number of 2s and 4s, the tree needs two more numbers different from 0 (immersed in any quantity of 0s), the first must be a 1, 2, 3 or 4 (to produce a change from the a c plane to c b plane), the second number must be a 1 or a 3, to change from c b to the b a plane).

  Thus, in any case, after a 1 or a 3 (immersed in 0s, 2s and 4s, perhaps) occurs, two more numbers (immersed in any quantity of 0s), a 1, 2, 3, 4 followed by a 1 or a 3, restore the original plane a b (or b a).

6.2.4 Symmetries

Fig. 20 shows clearly the symmetries between trees.

Acknowledgements

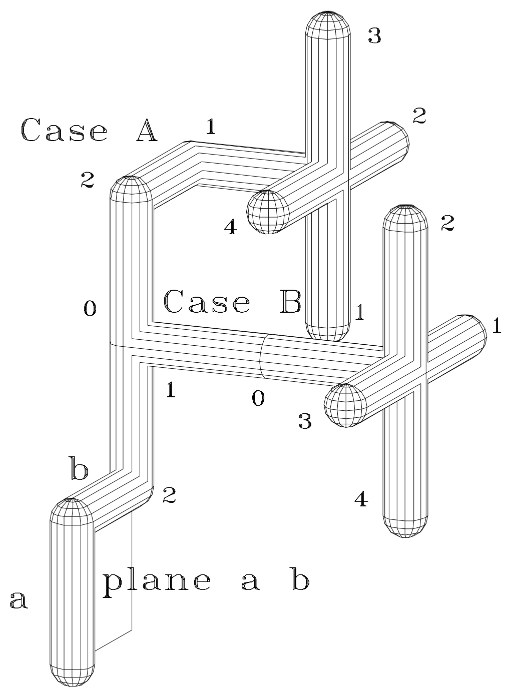


FIG. 19: Cases A and B of paragraph 5.2.3. Case A shows an even number of 2s and 4s (just two 2s) between the handle and the first 1 or 3 (in this case, an 1). After that, either direction 1, 2, 3 or 4 returns the growth of the tree to the *a b* (or *b a*) plane. Case B shows an odd number of 2s and 4s (just one 2) between the handle and the first 1. After that, either direction 1, 2, 3 or 4 returns the growth of the tree to to the *a b* (or *b a*) plane.

This work was supported by IIMAS-UNAM and by SNI-CONACyT. LAM acknowledges to PASPA-UNAM program. DEM data used in this study was provided by INEGI.

REFERENCES

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

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

[3] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, New Jersey (1982).

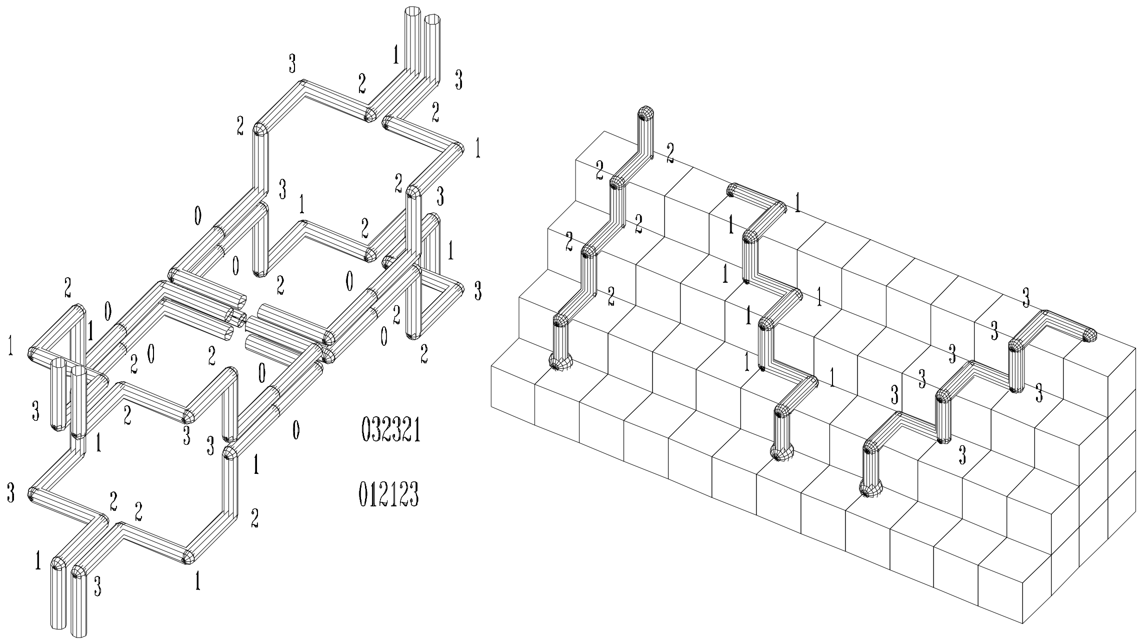


FIG. 20: The left figure shows a tree 0 3 2 3 2 1 and its symmetric tree, 0 1 2 1 2 3. As the text of the article shows, the mirror image (irrespective of the plane of the mirror) of one of these trees is the other tree. Notice also that the tree symmetric with respect to the origin to one of these trees is the other tree. The right drawing shows the trees 2 2 2 2 . . . , 1 1 1 1 . . . and 3 3 3 3 . . . , respectively.

[4] R.O. Duda, P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc., New York (1973).

[5] R.C. Gonzalez, R.E. Woods, *Digital Image Processing*, Second Edition, Prentice Hall, Upper Saddle River, New Jersey 07458 (2002).

[6] W. Karush, *Webster's New World Dictionary of Mathematics*, Webster's New World, Simon & Schuster, Inc. New York (1989).

[7] A. Guzmán, "Canonical shape description for 3-d stick bodies", MCC Technical Report Number: ACA-254-87, Austin, TX. 78759 (1987). In http://www.divshare.com/download/6279793-67e.

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

[9] A. Cayley, “A theorem on trees”, *Quart. J. Math.* **23**, 376-378 (1889).

[10] F. Sagols. “Hamiltonian representation of vox-solids”, *Computación y Sistemas* **4**, 213-217 (1999).

[11] L. Kikuchi, J. A. Guevara, D. Mark, and D. F. Marble, Rapid display of digital elevation models in a mini-computer environment, in *Proceedings of ISPRS Commission IV, Environmental Assessment and Resource Management*, Crystal City, Virginia, U.S.A., 1982, pp. 297-307.

[12] H. B. Voelcker and A. A. G. Requicha, Geometric modeling of mechanical parts and processes, *Computer* **10**, 1977, 48-57.

[13] J. W. Boyse, Data structure for a solid modeller, *NSF Workshop on the Representation of Three-Dimensional Objects*, Univ. Pennsylvania, 1979.

[14] B. I. Soroka, Generalised cylinders from parallel slices, in *Proc., PRIP*, 1979, pp. 421-426.

[15] B. I. Soroka and R. K. Bajcsy, Generalized cylinders from serial sections, in *Proc., 3rd IJCPR*, 1976, pp. 734-735.

[16] R. Brooks, Model-based 3-D interpretations of 2-D images, *IEEE Trans. Pattern Anal. Mach. Intelligence* **5** (2), 1983, pp. 140-150.

[17] A. A. G. Requicha, Representations of rigid solid objects, *Computer Surveys* **12**, 4, 1980.

[18] P. Besl and R. Jain, Three-dimensional object recognition, *ACM Comput. Surv.* **17** (1), 1985, 75-145.

[19] A. Pentland, Perceptual organization and the representation of natural form, *Artificial Intelligence* **28**, 1986, 293-331.

[20] E. Bribiesca, A. and Guzmán, How to describe pure forms and how to measure differences in shapes using shape numbers. (1980) Invited paper to the IEEE Conference on Pattern Recognition and Image Processing. Chicago, USA. Also in *Pattern Recognition* **12**, 101-112 (1980).

[21] J. M. Galvez, M. Canton, Normalization and shape recognition of three-dimensional objects by 3D moments, *Pattern Recognition* **26**, 667-681 (1993).

[22] H. Bullow, L. Dooley, D. Wermser, Application of principal axes for registration of NMR image sequences, *Pattern Recognition Letters* **21**, 329-336 (2000).