Voxel-based object representation by means of edging trees

L. A. Martínez[1], E. Bribiesca[2], and A. Guzmán[3]
[1]Instituto de Astronomía,
Universidad Nacional Autónoma de México,
México, D. F., México
[2]Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas,
Universidad Nacional Autónoma de México,
México, D. F., México
[3]Centro de Investigación en Computación,
Instituto Politécnico Nacional,
México, D. F., México

Abstract—*A method is described for representing voxel-based objects (VBOs) by means of edging trees (EdTs). Given a VBO, an EdT is a tree which traces the borders of the object. The vertices of the EdT correspond to the vertices of the enclosing surface where some of them have been conveniently hidden in order to get a 1D representation. The computed EdT is represented by a base-five digit chain code descriptor suitably combined by means of parentheses. The EdT notation is invariant under rotation and translation, using this notation it is possible to obtain the mirror image of any VBO with ease. The EdT notation preserves the shape of VBOs. The proposed EdT notation is a good tool for storing of VBOs. Due to their features, EdTs can be considered as a 1D alternative to skeletons for representing VBOs.*

Keywords: Voxel-based objects, edging trees, chain coding, 3D tree representation

1. Introduction

Representation of voxel-based objects (VBOs) is an important topic in computer vision and pattern recognition; accordingly, getting means of representation that provide an object of lower dimension that can be used for analysis and recognition has attracted attention of many research groups. Several methods with different approaches have been proposed to get representations of this kind of objects. One of such representations consists of line-like that can be transformed into 1D representations [1], by means of chain codes, convenient for tasks related with pattern recognition.

Recently a new method to represent VBOs was proposed by Bribiesca *et. al.* [2]. In this representation a base-five digit chain code so-called 5OT that describes orthogonal direction changes of straight-line segments is used to define an enclosing tree (EcT) that traverses all the vertices of a VBO. The vertices of an EcT correspond to the vertices of the enclosing surface of the analyzed VBO. Although EcTs can be used for pattern recognition, they may over-represent a VBO specially on planar faces where convenient hidden vertices lead to a simplified tree without loss of information.

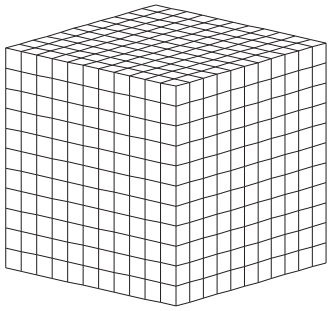
We are proposing a method for representing VBOs by means of edging trees (EdTs), as a first step in the development of an optimal representation based on the EcTs idea, which trace the borders of the object. The main aim of EdTs is to obtain a rough draft, as is the case of skeletons, of an object and representing it by means of the 5OT code. The EdT notation has several interesting properties such as to be invariant under rotation and translation, it is possible to obtain the mirror image of any VBO with ease. EdTs preserve the geometrical information of the underlying object and it can be recovered with ease from its EdT. In this connection, it should be noted that the 5OT chain code has shown be useful to represent 3D tree objects [3], to define a measure for shape dissimilarity of 3D curves [4] as well as to conduct compression efficiency studies of three-dimensional discrete curves [5], among other applications.

In order to offer a preliminary comparison, Figure 1 shows different representations for a given 3D object consisting of $11 \times 11 \times 11$ voxels (a): (b) is the skeleton obtained with the clasical prairie fire transformation [6], (c) is an EcT, and (d) the proposed EdT.

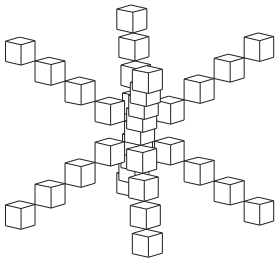
The paper is organized as follows. Section 2 presents the 5OT chain code and some preliminary definitions. Section 3 gives the method for generating EcTs. In Section 4, the definition of EdTs is presented, as well as some examples. Some properties of EdTs are given in Section 5.

2. The 5OT chain code

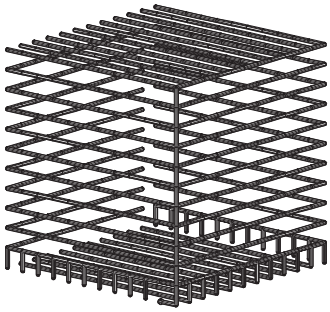
Before moving into the description of the 5OT chain code there are some preliminary concepts and remarks to be presented in this section. It is assumed that we will work only with VBOs. The length of each edge of voxels is considered equal to one, therefore the area of every face of each voxel is considered equal to one. There are three ways of connecting voxels: by edges, vertices, and faces. In the context of this



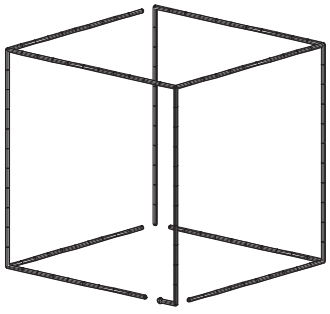
(a)



(b)



(c)



(d)

Fig. 1: Comparison of different representations of a given 3D voxel-based object (a): (b) is the praire fire skeleton, (c) is an enclosing tree and (d) an edging tree.

paper, we only consider face-connected voxels, i.e. voxels with six-connectivity. The area of the enclosing surface of an object composed of a finite number of voxels, corresponds to the sum of the areas of the voxels located on the visible faces of the solid. The chain descriptor of an object is defined by the computation of the chain elements using the nested-parentheses notation for trees. Following graph theory basic definitions, EcTs and EdTs describe trees of maximum degree six in three dimensions, this is due to the fact that EdTs only represent face-connected VBOs.

A chain a is an ordered sequence of n elements, and is represented by $a = a_1 a_2 a_3 \dots a_n = \{a_i : 1 \leq i \leq n\}$. An element $a_i \in \{0, 1, 2, 3, 4\}$ of a chain in the 5OT code indicates the orthogonal direction change of the contiguous straight-line segments of the 3D branch in that element position. Two contiguous straight-line segments of a branch define a direction change and two-direction changes define a chain element. If the consecutive sides of the reference angle have directions b and c as shown in Figure 2(b), 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 is given by the following function:

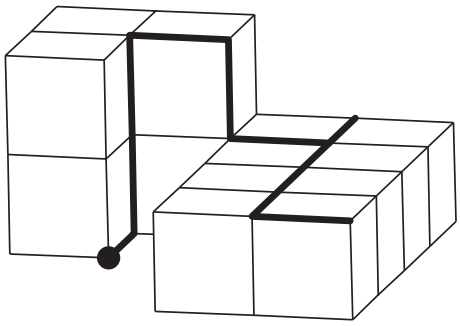
$$\text{chain element}(b, c, d) = \begin{cases} 0 & \text{if } d = c, \\ 1 & \text{if } d = b \times c, \\ 2 & \text{if } d = b, \\ 3 & \text{if } d = -(b \times c), \\ 4 & \text{if } d = -b, \end{cases} \tag{1}$$

where $\times$ denotes the vector product in $\mathbb{R}^3$. The elements b and c constitute *the handle*.

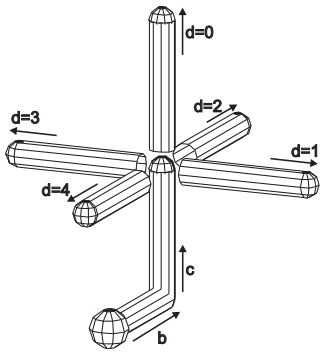
Figure 2 summarizes the rules for labeling the vertices depending on the position of such an angle with respect to the preceding handle in the path. Figure 2(a) shows an example of a tree plotted over a VBO, and how function (1) is used to define its 5OT chain code descriptor. The dot indicates the initial tree vertex. Using the only five possible chain elements of Figure 2(b) given by function (1), a tree descriptor is constructed as new tree vertices are being discovered.

The procedure to find the tree descriptor is as follows:

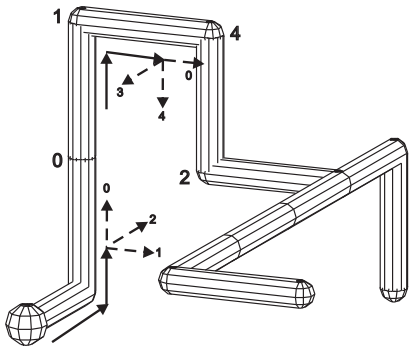
1) Select an arbitrary end vertex of the tree as the origin. In Figure 2(c) the selected origin is represented by a sphere.
2) Compute the chain elements of the tree. Figure 2(c) shows that the first computed element of the chain corresponds to a "0" because the first straight-line segment follows the direction of the last segment. The second element corresponds to the chain element "1" and the handle is still the same as for the first chain element. Note that the dotted arrows indicate only



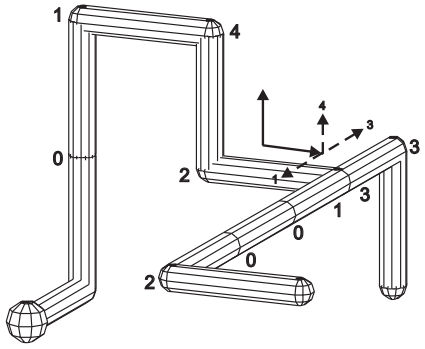
(a) An example of tree plotted over a voxel-based object.



(b) Given the handle and vectors b and c, the five possible chain elements are shown.



(c) The first four chain elements of an example of a tree and its corresponding partial descriptor: 0142.



(d) The chain elements of the same tree and its complete tree descriptor: $0142(1002)(33)$.

Fig. 2: Example of a chain elements computation defined by function (1).

three of the five directions. At the end of this stage the chain is as follows: 0142.

3) It is a known fact that trees can be represented by a notation that uses nested parentheses. Using this notation the next chain element of the tree will be computed. In Figure 2(d) a vertex which is a junction has been reached. In order to decide what direction to go, in the case of Figure 2(d), there are only two possible ways represented by the chain elements "1" and "3" as indicated by the function (1) applied to the new handle. Note that around a branch, it is necessary to know what nonzero element was the last one to define the next element. This ensures that orientation is not lost. The directions are selected in numerical order. Thus, the first selected direction is represented by the chain element "1" and $0142(1002)$ is the descriptor at this stage. The nested parentheses describe the branch whose chain is (1002). After coming back to the junction node and compute the chain of the next branch, the tree descriptor is equal to $0142(1002)(33)$.

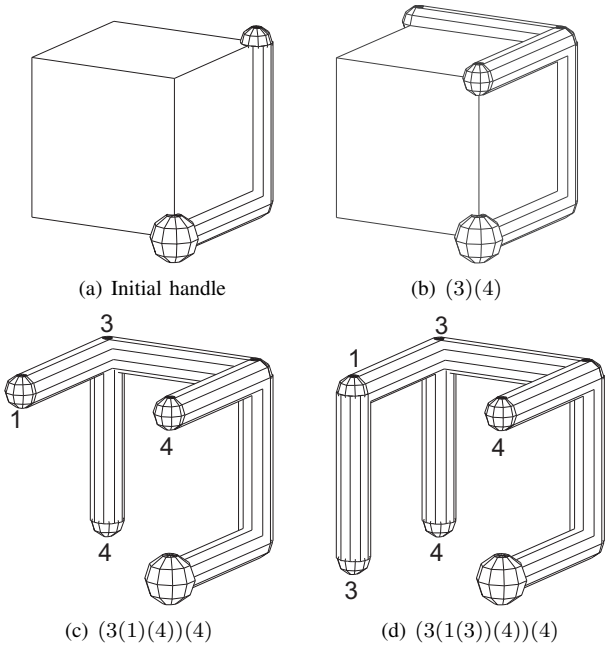
3. Enclosing trees

The enclosing trees (EcTs) were proposed as a 5OT chain code based alternative to represent VBOs [2]. EcTs are trees which cover each vertex of the visible surface of VBOs and provide a base-five digit strings suitably combined by means of parentheses 1D representation. The vertices of the EcTs correspond to the vertices of the enclosing surface of the analyzed object.

The EcTs computation process using the trees descriptor is as follows:

1) Select an arbitrary vertex of the enclosing surface of the VBO as the origin of the EcT.
2) Choose an arbitrary direction to define the chain elements. The first direction change is composed by two contiguous straight-line segments that will form the first reference handle.
3) Compute the chains that form the enclosing tree:
   a) Following the numerical order determined by the directions obtained in the previous step, search for the neighbor vertices of the enclosing surface of the object.
   b) Repeat the above step until all vertices of the enclosing surface have been reached by the enclosing tree.

Figure 3 illustrates the example of the smallest EcT that can be computed. This is the case for an object consisting of only one voxel. Figure 3(a) shows the original voxel, the initial vertex and the arbitrary direction chosen to conform the first handle to starting the EcT. Figure 3(b) displays the first junction node with the only two possible ways to continue the tree which coincide with chain elements "3" and "4". In this stage the tree descriptor is (3)(4). In the



(a) Initial handle (b) (3)(4)

(c) (3(1)(4))(4) (d) (3(1(3))(4))(4)

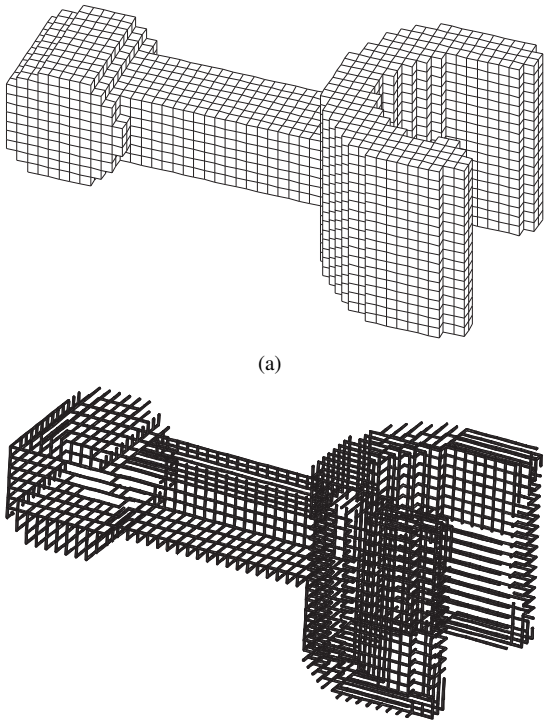
Fig. 3: Enclosing tree computation process for a voxel.

next stage, the processing restarts over the path indicated by the chain "3" because it is the lowest number. Figure 3(c) shows the neighbor vertices reached through the vertex that represents the chain element "3". Those vertices match with the chains "1" and "4". Thus the tree descriptor in this stage is (3(1)(4))(4). The tree descriptor cannot follow through the chain element "4" because all its neighbor vertices have been previously visited. Figure 3(d) shows the final stage of the EcT, and its tree descriptor is as follows: (3(1(3))(4))(4).

Figure 4 presents another example of EcT, in this case the underlying object is a pivoted lever. For a complete review of EcTs and their properties, see Ref. [2].

4. Edging trees

There are solids whose surfaces can be represented without going through all the vertices that make up those surfaces. In these cases, it can be considered that EcTs over-represent the VBO. Figure 1(c) shows an EcT and Figure 1(d) presents an EdT for a $11 \times 11 \times 11$ voxel VBO in which the EcT is considerably more complex than the EdT. The aim of EdTs is to cover VBOs traveling through their edge and represent it by means of a tree descriptor. The basic idea is to obtain a representation similar to a 1D *skeletal* representation via the 5OT chain code in which the planar faces of the 3D object have replaced by the border of the face. Figure 5 shows an example in which its EdT can be a more convenient representation than the corresponding EcT, especially for manufactured parts. Figure 7 presents another example of VBO and its respectie EdT. The chain descriptors of Figures 4, 5 and 7 were omited.



(a)

(b)

Fig. 4: A pivoted lever (a) with its computed enclosing tree (b).

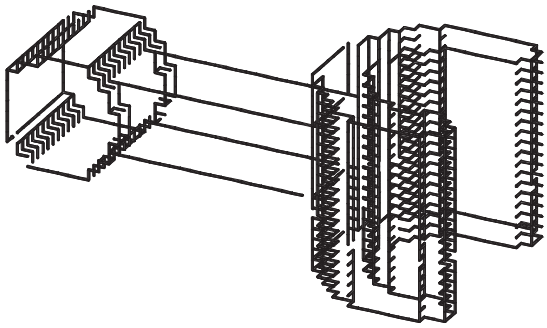
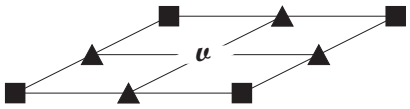


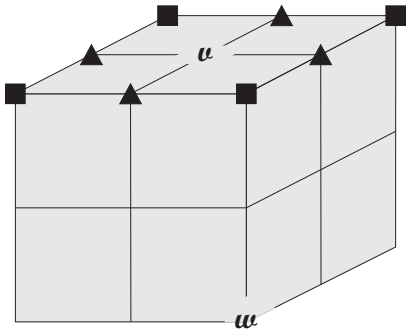
Fig. 5: Edging tree computed for the pivoted lever shown in Figure 4.

In order to compute an EdT it is necessary to omit some vertices in the computation process described in section 3. The candidates are those that have a planar neighborhood. The *xy-planar neighborhood* of v, denoted by $N_{xy}(v)$, is the set of 6 and 18-neighbors of v which lie on a plane parallel to the plane $z = 0$. $N_{xz}(v)$ and $N_{yz}(v)$ are the planar neighborhoods parallel to planes $y = 0$ and $x = 0$, respectively (see Figure 6(a) for an example of planar neighborhood). Figure 6(b) shows a planar neighborhood of v. Also it is a shown a vertex w which has no planar neighborhoods.

It should be noted that if a vertex has a planar neigh-



(a) Planar neighborhood of v.



(b) Planar neighborhood of w on a $2 \times 2 \times 2$ voxel-based object. The vertex w has no planar neighborhoods.

Fig. 6: Planar neighborhoods for a vertex v which lie over the surface of a VBO.

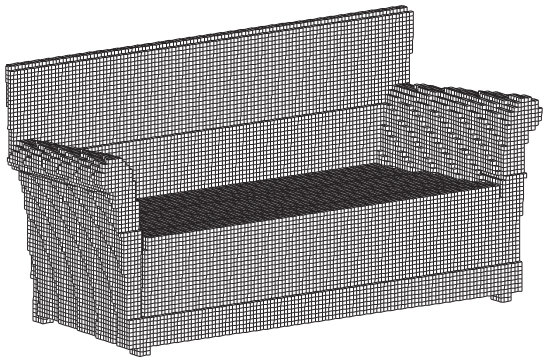
borhood then it belongs to a face on the surface of the VBO like v in Figure 6(b). Vertices that have no a planar neighborhood like w will constitute the border. Thus, the EdT computation process is the same as that used to compute EcTs except it includes a step 0 in which vertices that have a planar neighborhood are detected and omited from the VBO vertices list before applying the procedure described in section 3.

5. Some properties of edging trees

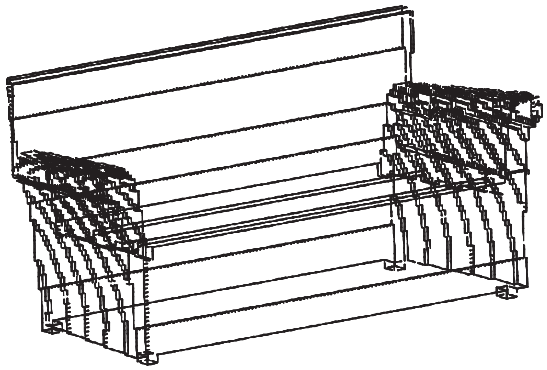
Given a VBO its EdT chain code notation is invariant under rotation. Once the starting vertex and the handle have been determined the EdTe is constructed using the relative direction changes based on the local handle which is not affected by a rotation of the underlying object [3].

Using the tree descriptor, the mirror image of any tree is obtained with ease. If a is a tree descriptor, its mirror descriptor can be obtained by replacing in a each occurrence of "1" by "3" and vice versa [3].

Given the descriptor of an EdT, the surface vertex list of the underlying 3D object can be recovered with the exception of those vertices omited because having planar neighborhoods. Due to the fact that the descriptor starts after the initial handle, the coordinates recovering will depend on the initial directions selected by the user in the first step of the process. Notice that every opening parenthesis "(" represents a tree node. The nested parentheses notation for trees used by the 5OT chain code corresponds to a pre-order *depth-first search* traversing [7], as a result the list shall have the same ordering.



(a)



(b)

Fig. 7: Another example of VBO and its corresponding enclosing tree.

Whereas EcT notation may be used for lossless compression of VBOs because it preserves information and allows considerable data reduction [2]. EdTs can get a higher compression ratio. A priori is difficult to establish the reduction in the descriptor length when EdTs are used instead of EcTs due to the fact that the number of vertices to be deleted changes and varies according to the processed VBO. As a consequence of the selective deleting of surface vertices on planar faces, EdTs can reach every vertex in zones of VBOs that cannot be simplified. In fact, in the worst case when no vertices can be suppressed, EdTs are the same as EcTs.

6. Acknowledgments

L.A.M. acknowledges a DGAPA-UNAM grant and support from Instituto de Astronomía, Universidad Nacional Autónoma de México. This work is part of a doctoral dissertation under the direction of Prof. Bribiesca.

References

[1] A. Guzmán, "Canonical shape description for 3-d stick bodies", MCC Technical Report, Austin, TX. 78759, Tech. Rep. ACA-254-87, 1987.

[2] E. Bribiesca, A. Guzmán A and L. A. Martínez, "Enclosing trees", *Pattern Anal. Applic.*, vol. 15, pp. 1-17, 2012.

[3] E. Bribiesca, "A method for representing 3D tree objects using chain coding", *J. Visual Commun. Image Represent.*, vol. 19, pp. 184-198, 2008.
[4] E. Bribiesca and W. Aguilar, "A measure of shape dissimilarity for 3D curves", *Int. Journal of Contemp. Math. Sci.*, vol. 15, pp. 727-751, 2006.
[5] H. Sánchez-Cruz and E. Bribiesca, "Study if compression efficiency for three-dimensional discrete curves", *Opt. Eng.* . 47 (7), 077206, july 2008.
[6] R. O. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
[7] D. Knuth, *The Art of Computer Programming*, Volume 1: *Fundamental Algorithms*, 3rd ed., Addison-Wesley, 1997.