A new relative chain code in 3D

Hermilo Sánchez-Cruz[a], Hiram Habid Lopez-Valdez[a] and Francisco Javier Cuevas[b]

a *Universidad Autónoma de Aguascalientes. Department of Computer Science.* Av. Universidad 940, Col. Ciudad Universitaria, Aguascalientes 20131, Aguascalientes, México

b *Centro de Investigaciones en Óptica, A. C., Department of Optical Metrology*, Loma del Bosque 115, Col. Lomas del Campestre, León 37150, Guanajuato, México *

Abstract

A new chain code to represent 3D discrete curves is proposed. The method is based on a search for relative changes in the 3D Euclidean space, composed of three main vectors: a reference vector, a support vector, and a change direction vector, utilized to obtain a directed simple path in a grid of 26 connected components. A set of rotation transformations is defined in the 3D Euclidean space, and despite encountering a complete set of 73 different basic pattern chains, an alphabet of only 25 symbols is required to represent any face, edge or vertex-connected discrete curve. Also, we used the code to represent trees, using parentheses and a lexicographical order to be able to traverse each of their branches. Important properties of this code are found: independence under translation, rotation and mirror transformations, as well as high compression levels. Finally, a set of 3D curve-skeleton and digital elevation model data to study the terrain were utilized to prove the proposed code.

Keywords: 3D chain code; invariant, simple paths, curve-skeletons, trees, compression, relative code

1 Introduction

Nowadays, research on chain codes is an important and very active field in computer vision and pattern recognition. In recent years, a vast number of papers about contour representations by means of chain codes have been written ([1] - [24]).

On the other hand, the first approach to represent 3D digital curves using chain codes was introduced by Freeman in 1974 [22]. Line structures are quantized on a cubic lattice (voxels). For each data node, there are 26 possible directions to the next data node. So, this chain code is based on face, edge and vertex-connectivity and depends absolutely on Cartesian coordinates (see Fig. 1).

To determine whether or not a discrete curve is a digital line segment, in 1983 Kim [2] defined digital arcs in 3D digital pictures.

Guzman [3] proposed a 3D coding for stick bodies. In such a work, Guzman considered stick bodies divided in two parts: "limb" and "junction". Limbs are the elongated parts (generalized cylinders) protruding in three roughly orthogonal directions; whereas junctions are the places where limbs meet. It is stated that the structural approach of pattern recognition and scene understanding rely on the decomposition of a scene into identifiable parts and its subsequent syntactical analysis.

Digital representation schemes for 3D curves were presented in 1997 by Jonas et al. [23]. In 2003, Safonova and Rossignac [24] proposed a compact approximation scheme for 3D curves; they showed that piecewise circular curves have an advantage over polygonal and b-spline curves. In [25], a method is described for reconstructing a 3D rigid curve from a sequence of uncalibrated images using 3D epipolar

*Part of this work was made by the first author during a sabbatical stay at *Centro de Investigaciones en Óptica, A. C., Department of Optical Metrology*, León, Guanajuato, México

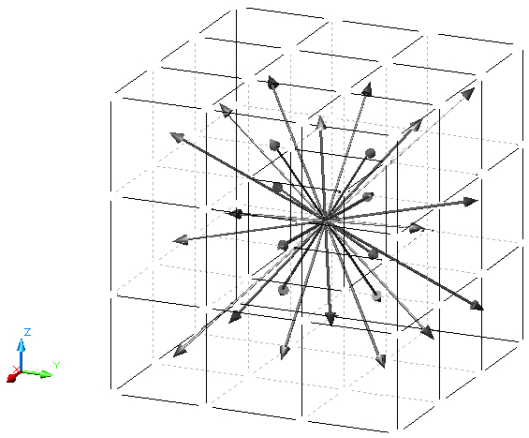


Figure 1: The 26 directions of the Freeman code

parameterization; it is shown that 6-connected chain codes match all the identified requirements and should be preferred.

The orthogonal direction change chain code (5OT) for representing 3D discrete curves, composed of five symbols, was proposed by Bribiesca [13]. Recently Bribiesca [14] has utilized 5OT chain code to apply on tree structures. 5OT is a code to represent *face-connectivity*.

On the other hand, there is something interesting to note in previous papers about chain codes, particularly in two dimensions: chain code representations are suitable to handle not only edges of pixels, but, also, their vertices. In literature, there are chain codes to represent four-connectivity: F4 (Freeman chain code of four directions, [1]), 3OT (three orthogonal chain directions, [5]), VCC (vertex chain code, [4]) and C-VCC (compact vertex chain code, [8]), whereas for eight-connectivity there are F8 (Freeman chain code of eight directions, [1]), AF8(Angle Freeman Chain code of eight directions [7]) and MDF9 (Modified Differential Freeman chain code of nine symbols, [15]). For 3D chain codes, the most used is Freeman chain code of 26 directions [22], whereas the relative code for face-connectivity is 5OT [13]. As can be noted, an important question arises for 3D chain code representations: is it possible to find a relative chain code, not only for face connectivity but for edge and vertex connectivity, *i.e.*, for the 26 vicinity in a cubic grid representation?

In this work, we have answered this question affirmatively; moreover, we have applied a proposed 3D chain code to curve-skeletons and to digital elevation model data. Also, we found that it is possible and feasible to propose a code in three dimensions that is invariant under transformations of translation, rotation, mirror and independent to the vicinity of the voxels: face, edge or vertex connectivity. Another important advantage is that the new code requires little storage in memory.

In Sections 2, we introduce important concepts and definitions that are used throughout the paper: Simple Paths, Generalized Elemental Paths (GEPs), Rotation Transformations and Equivalent Elemental Paths. In Section 3, absolute and relative codes, as well as our proposed three-dimensional relative code (3DRC), are addressed. In Section 4, we apply our proposed 3DRC code to simple paths and tree structures; we also explain how to decode. In Section 5, invariant under translation, rotation and mirror transformations are proved, whereas in Section 6 the method is used to encode 3D objects, and an analysis and comparison with existing codes are carried out. Conclusions and further work are given in Section 7.

2 Important concepts and definitions

In this section, we introduce the most important concepts and definitions used throughout the paper. Mainly, they are concerned with elemental paths, generalization of elemental paths, and equivalent elemental paths.

2.1 Elemental paths and generalized elemental paths (GEPs)

Let us consider a grid in a 3D space [26], where the *grid point* set is in $\mathbb{Z}^3$.

Definition 1 A *voxel, v,* is a resolution cell of a grid in a 3D space with Cartesian coordinates $c(x, y, z)$, and an intensity value $I_v = \{0, 1\}$. If $I_v = 0$, we say that the voxel is 0-voxel; on the contrary, if $I_v = 1$, we say 1-voxel. Unless otherwise stated, in this work we consider 1-*voxel* simply as *voxel*.

Definition 2 We say two voxels v_1 and v_2 are *adjacent* if $v_1 \neq v_2$ and $v_1 \cap v_2 \neq \phi$.

With this definition, we note that there are only three configurations where two voxels obey this condition. See Fig. 2.

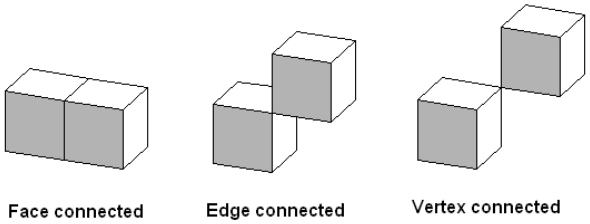


Figure 2: Three types of adjacency depending of the voxel neighborhood.

Definition 3 If c_1 are the coordinates of v_1, c_2 the coordinates of v_2, and $b_1 = c_2 - c_1$, note that v_1 and v_2 are adjacent if and only if

$$b_1 \in \mathcal{B} = \{\langle i, j, k\rangle \mid i, j, k \in \{-1, 0, 1\}\} \setminus \langle 0, 0, 0\rangle$$

The set $\mathcal{B}$ is called a *grid basis*.

Definition 4 A *path* is a sequence of adjacent ordered voxels $P = \{v_1, v_2, \cdots, v_n\}$, such that v_1 is adjacent to v_2, v_2 is adjacent to v_3, ..., v_{n-1} is adjacent to v_n. The set of vectors

$$P_{\mathcal{B}} = \{b_1, b_2, \cdots, b_{n-1}\} \subset \mathcal{B}$$

is called *basis of path P*.

The paths we start to codify are called *simple paths* (**Definition 9**), which are composed by *generalized elemental paths* (**Definition 8**).

Definition 5 If v_1, v_2 and v_3 are ordered voxels, we say (v_1, v_2) is *better connected* than (v_1, v_3) if $c_2 - c_1, c_3 - c_1 \in \mathcal{B}$, and also, one of the following conditions is satisfied:

1. v_1 is not adjacent to v_3
2. if v_1 is adjacent to v_3, then v_1 is face with v_2 and v_1 is edge with v_3,
3. if v_1 is adjacent to v_3, then v_1 is face with v_2 and v_1 is vertex with v_3,
4. if v_1 is adjacent to v_3, then v_1 is edge with v_2 and v_1 is vertex with v_3

Definition 6 An *elemental path* is an ordered set of four voxels,

$$P = \{v_1, v_2, v_3, v_4\},$$

such that $b_1, b_2, b_3 \in \mathcal{B}$, $b_1 \neq b_2$; $b_2 \neq b_3$, (v_1, v_2) is better connected than (v_1, v_3), and (v_2, v_3) is better connected than (v_2, v_4).

By **Definition 4** the basis of P is $P_{\mathcal{B}} = \{b_1, b_2, b_3\}$, and let us call it *basis of the elemental path.*

Definition 7 Let the elements b_1, b_2, b_3 be called *reference, support and change* of P, and denote them by ref, *supp* and *chng*, respectively.

The examples of Fig. 3 present elemental paths and non elemental paths. Note that elemental paths agree with Def. 5 and Def. 6. Whereas the non elemental paths do not obey Def. 5 or Def. 6.

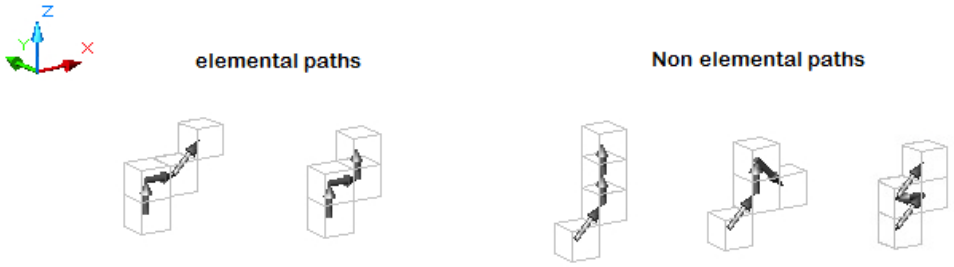


Figure 3: Example of elemental and non elemental path. Note that Def. 5 and 6 guarantee only one path for a four voxel array.

Note that the order in which the voxels are covered is very important. A path can be covered in a certain way to obtain elemental paths.

Of course, the paths in this paper are not necessarily composed of only elemental paths, but they can be tried by a little more complex structures: generalized elemental paths.

Definition 8 A *generalized elemental path* (GEP for short) is an ordered set of $m \geq 4$ voxels

$$P = \{v_1, v_2, \ldots, v_m\},$$

where $b_1, b_2, \ldots, b_{m-1} \in \mathcal{B}$, $b_1 \neq b_2; \underbrace{b_2 = b_3 = \cdots = b_{m-2}}_{m-3}; b_{m-2} \neq b_{m-1}$, (v_1, v_2) is better connected than (v_1, v_3), and (v_{m-2}, v_{m-1}) is better connected than (v_{m-2}, v_m).

Something noteworthy of **Definition 8** is that if $ref_i, supp_i$ and $chng_i$ are the vectors associated with a GEP, then $ref_{i+1} = supp_i, supp_{i+1} = chng_i$, and $chng_{i+1}$ are the vectors associated to the GEP P_{i+1}, $i = 1, 2, \cdots, k-1$.

As in **Definition 7**, let us call the elements b_1, b_2, b_3 *reference, support and change* vectors from the GEP and denote them by ref, *supp* and *chng*, respectively.

Of course, a GEP is an elemental path when $m = 4$. As GEPs examples we have Fig. 4.

A "simple path" is essentially a path composed of GEPs. The formal definition can be written as follows:

Definition 9 A path $P = \{v_1, \ldots, v_n\}$ is called a *simple path* of *length* n if there are integers $m_1, \cdots, m_{k+1}, m_{k+2}$ greater than zero, such that the basis of P is of the form

$$P_{\mathcal{B}} = \left\{b_1^{m_1}, b_2^{m_2}, \ldots, b_{k+1}^{m_{k+1}}, b_{k+2}^{m_{k+2}}\right\},$$

and the sets

$$P_{\mathcal{B}}^1 = \{b_1, b_2^{m_2}, b_3\}, \qquad P_{\mathcal{B}}^2 = \{b_2, b_3^{m_3}, b_4\}, \qquad \cdots \qquad , P_{\mathcal{B}}^k = \left\{b_k, b_{k+1}^{m_{k+1}}, b_{k+2}\right\}$$

(where $b_i \neq b_{i+1}$) are GEP basis. If these conditions are fulfilled, we call voxels v_1 and v_n *start* and *final of the path*, respectively, whereas we refer to $P_{\mathcal{B}}$ as *basis of the simple path*, or simply *basis* of P.

Fig. 5 shows the GEPs and the basis of a simple path.

Other concepts used trough this work are concerned with rotation transformations. We define the *vox rotation* as follows.

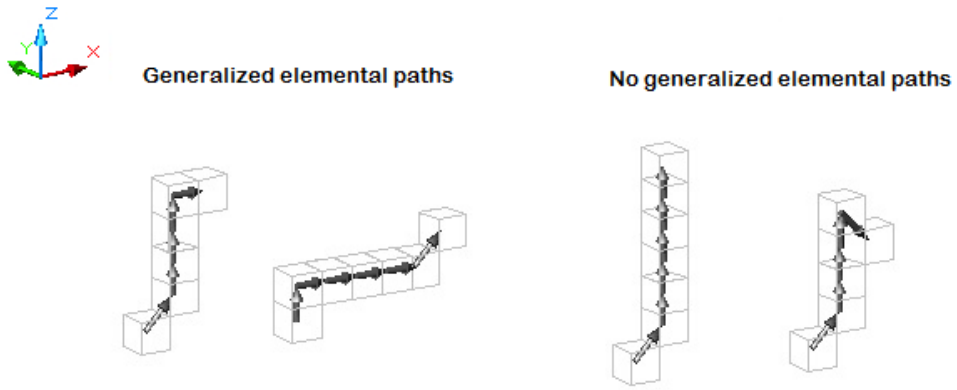
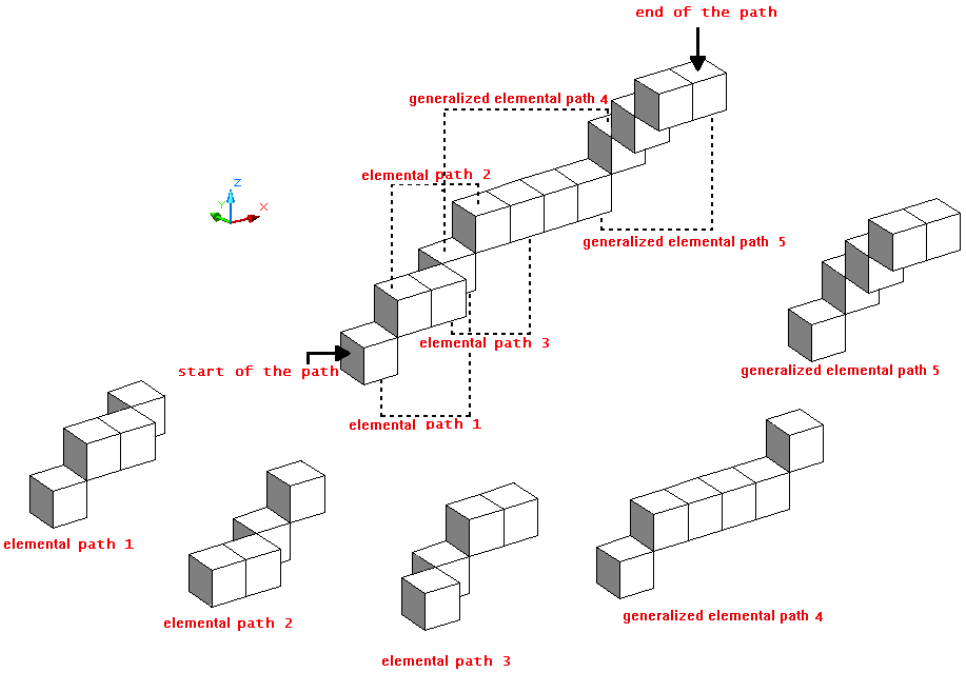


Figure 4: Examples of GEPs and paths that are not GEPs.



$$
\begin{aligned}
P_{\mathcal{B}} &= \{(1,0,1),(1,0,0),(1,1,0),(1,0,1),(1,0,0),(1,0,0),(1,0,0),(1,0,1),(0,-1,1),(0,-1,1),(1,0,0)\} \\
&= \left\{(1,0,1),(1,0,0),(1,1,0),(1,0,1),(1,0,0)^3,(1,0,1),(0,-1,1)^2,(1,0,0)\right\}
\end{aligned}
$$

Figure 5: An example of a simple path and its basis. The simple path is composed of GEPs

2.2 Rotation transformations

To achieve our objective, we use a discrete type of rotations. On the one hand, the known usual rotations are as follows:

Definition 10 Let $v = \langle v_x, v_y, v_z \rangle$ be a vector in $\mathbb{R}^3$.

We denote the *usual rotations* of θ degrees around the axis x, y and z by $v^{x,\theta}, v^{y,\theta}, v^{z,\theta}$, respectively, and they are given by:

$$\text{i)}\ v^{z,\theta} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$\text{ii)}\ v^{y,\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$\text{iii)}\ v^{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

The following is a different rotation definition, very useful for our proposed relative encoding.

Definition 11 Let $u = \langle u_x, u_y, u_z \rangle$ be a vector in $\mathbb{R}^3$.

A *Vox Rotation of θ degrees around the x axis* from vector u is defined by

$$\mathbf{Rot_x}\,(u, \theta) = \langle u_x, t_y, t_z \rangle\,,$$

where the angle of projection from $\langle u_y, u_z \rangle$ to $\langle t_y, t_z \rangle$ (given in counter-clockwise direction) is θ degrees. Giving an example of this function, see the pictures of Fig. 6.

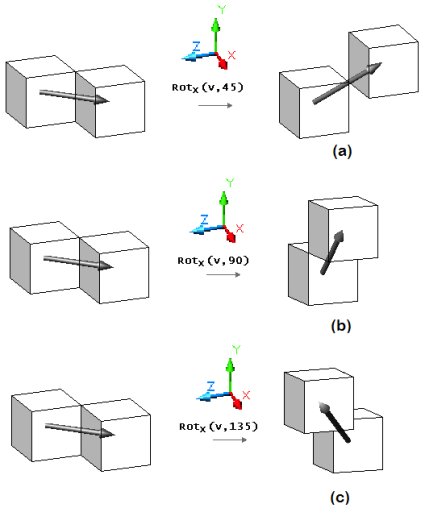


Figure 6: Rotation definitions. A projection on the plane y-z, of (a) 45 degrees, (b) 90 degrees and (c) 135 degrees.

In an analogue way, let us define the rotations around the y axis and around the z axis. *Vox Rotation of θ degrees around the y* axis of vector v is given by

$$\mathbf{Rot_y}\,(u, \theta) = \langle t_x, u_y, t_z \rangle\,,$$

where the angle from $\langle u_x, u_z \rangle$ to $\langle t_x, t_z \rangle$ (given in counter-clockwise direction) is θ degrees.

Vox Rotation of θ degrees around z axis of the vector u is given by

$$\mathbf{Rot_z}\,(u, \theta) = \langle t_x, t_y, u_z \rangle\,,$$

where the angle from $\langle u_x, u_y \rangle$ to $\langle t_x, t_y \rangle$ (given in counter-clockwise direction) is θ degrees.

Definition 12 For a vector $u = \langle u_x, u_y, u_z \rangle$ in $\mathbb{R}^3$, *Vox Rotations of θ degrees around $-x$ axis, around $-y$ axis and around $-z$ axis* are defined by

$$\begin{aligned} \mathbf{Rot}_{-\mathbf{x}}(u,\theta) &= \mathbf{Rot}_{\mathbf{x}}(u,-\theta), \\ \mathbf{Rot}_{-\mathbf{y}}(u,\theta) &= \mathbf{Rot}_{\mathbf{y}}(u,-\theta), \\ \mathbf{Rot}_{-\mathbf{z}}(u,\theta) &= \mathbf{Rot}_{\mathbf{z}}(u,-\theta), \end{aligned}$$

respectively.

As we can notice, when a vox rotation is carried out, vectors of the basis path may change in direction and/or magnitude. However, while this happens, we say that the bases are equivalent (see **Definition 13**). Moreover, although voxel arrays are different in the shapes after rotations, we can say that they are also equivalent (see **Definition 14** below), because so are their associated bases. These concepts are discussed in the next sub section.

2.3 Equivalent elemental paths

A new concept can be derived from rotation transformations defined above. We notice that although different paths can have different shapes, we can get one from another, simply by doing some rotations. When that happens, we say that the paths are equivalent. To be more precise, we have the following definitions:

Definition 13 Two *bases* $P_{\mathcal{B}} = \{b_1, \ldots, b_n\}$ and $P'_{\mathcal{B}} = \{b'_1, \ldots, b'_n\}$ of length n are *equivalent* if after applying a rotation to $b_1, b_2, \ldots, b_n$, we obtain $b'_1, b'_2, \ldots, b'_n$, respectively.

With this definition, we can now tell when two paths are equivalent.

Definition 14 Two *paths* P_1 and P_2 are *equivalent* if their bases are.

Equivalence implies that the shape produced by three directed segments are similar, *i.e* they are equal except for a deviation of less than 45 degrees.

With the vox rotations defined, we can conclude that given two elemental paths of different shapes, they can be equivalent if their corresponding bases are equivalent.

In fact, given two equivalent elemental paths, there are projections on the planes that have the same shape, independently of orientations and scales.

Note that if an elemental path is rotated under last definitions, and is face or edge connected, the basis remains with the same shape in the projection perpendicular to the axis of rotation, independently of scale. On the contrary, if the elemental path is vertex connected, the angles do not remain exactly the same. However, there are projections on the plane that allow us to visualize the invariance in the shape of the associated curve (bold directed segments in Fig. 7).

From example of Fig. 7, the voxel arrays of the columns (b) and (e) are equivalent. Independently of the orientation and scale, both paths on the first row fit themselves perfectly; in particular, they have $90°$ and $135°$ of direction changes. The arrays of the second row are equivalent and have exactly the same shapes, independently of the orientation and scale, of course. Both have $45°$ in the two changes of directions. In the case of the third row, they are also equivalent, despite the presence of a small difference of $angcos(1/2) - angcos(1/\sqrt{3}) \simeq 5.3°$ in the second change direction. It can be easily verified taking into account the voxel coordinates.

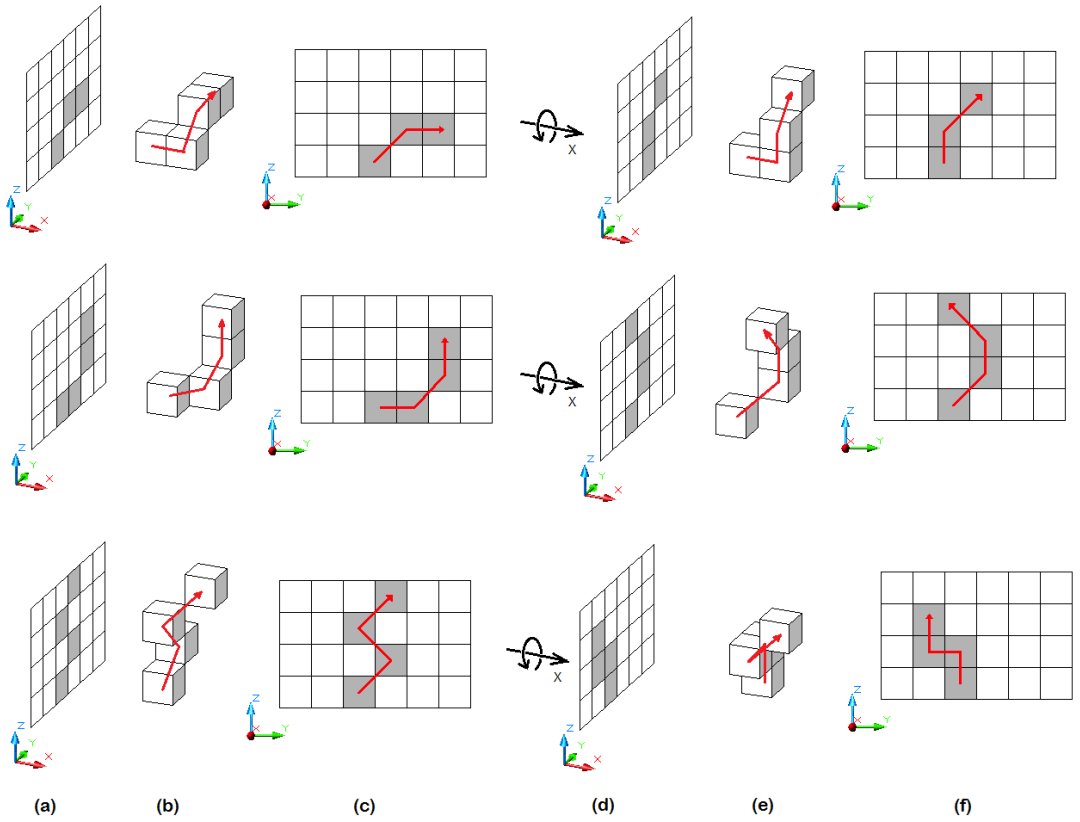


Figure 7: Equivalent elemental paths: a vox rotation of 45° performed on elemental paths in column (b) to obtain voxel arrays in column (e); (a) and (d) are the projections on the plane Z-Y of the objects in column (b) and (e) respectively, (c) and (f) are the viewpoints on the plane Z-Y of the directed path of columns (b) and (e), respectively.

3 Absolute and relative chain codes

By absolute chain code we understand a chain code that is produced when every change direction to be given in the contour is dependent absolutely on the Cartesian coordinates, no matter what is the change direction of the previous vector. On the contrary, by relative chain code we understand a chain code that is produced when every change direction does not depend directly on the Cartesian coordinates, but on the change produced in the last movement while covering the contour chain, *i.e.* "the way the previous movement was, the way the next one will be".

3.1 Absolute $F26$ coding

As it is known, $F26$ coding depends on the absolute Cartesian system, *i.e.*, it is not invariant under rotation transformations. The first codification of a simple path arises in a natural way. If we define the elements of $\mathcal{B}$ as following,

$$\underbrace{\langle 1,0,0\rangle}_{\text{a}},\underbrace{\langle 1,1,0\rangle}_{\text{b}},\underbrace{\langle 0,1,0\rangle}_{\text{c}},\underbrace{\langle -1,1,0\rangle}_{\text{d}},\underbrace{\langle -1,0,0\rangle}_{\text{e}},\underbrace{\langle -1,-1,0\rangle}_{\text{f}},\underbrace{\langle 0,-1,0\rangle}_{\text{g}},\underbrace{\langle 1,-1,0\rangle}_{\text{h}},\underbrace{\langle 1,0,1\rangle}_{\text{i}},$$
$$\underbrace{\langle 1,1,1\rangle}_{\text{j}},\underbrace{\langle 0,1,1\rangle}_{\text{k}},\underbrace{\langle -1,1,1\rangle}_{\text{l}},\underbrace{\langle -1,0,1\rangle}_{\text{m}},\underbrace{\langle -1,-1,1\rangle}_{\text{n}},\underbrace{\langle 0,-1,1\rangle}_{\text{o}},\underbrace{\langle 1,-1,1\rangle}_{\text{p}},\underbrace{\langle 1,0,-1\rangle}_{\text{q}},\underbrace{\langle 1,1,-1\rangle}_{\text{r}},$$
$$\underbrace{\langle 0,1,-1\rangle}_{\text{s}},\underbrace{\langle -1,1,-1\rangle}_{\text{t}},\underbrace{\langle -1,0,-1\rangle}_{\text{u}},\underbrace{\langle -1,-1,-1\rangle}_{\text{v}},\underbrace{\langle 0,-1,-1\rangle}_{\text{w}},\underbrace{\langle 1,-1,-1\rangle}_{\text{x}},\underbrace{\langle 0,0,1\rangle}_{\text{y}}\underbrace{\langle 0,0,-1\rangle}_{\text{z}},$$

then, the alphabet that we use is:

$$F26 = \{\text{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}\}\,, \tag{1}$$

and the coding is obtained when each b_i is changed from $P_{\mathcal{B}} = \{b_1, b_2, \ldots, b_{n-1}\}$ to its respective name in $F26$. This is what we call $F26$ *codification*.

Example. Taking into account that the simple basis path of Fig. 5 is

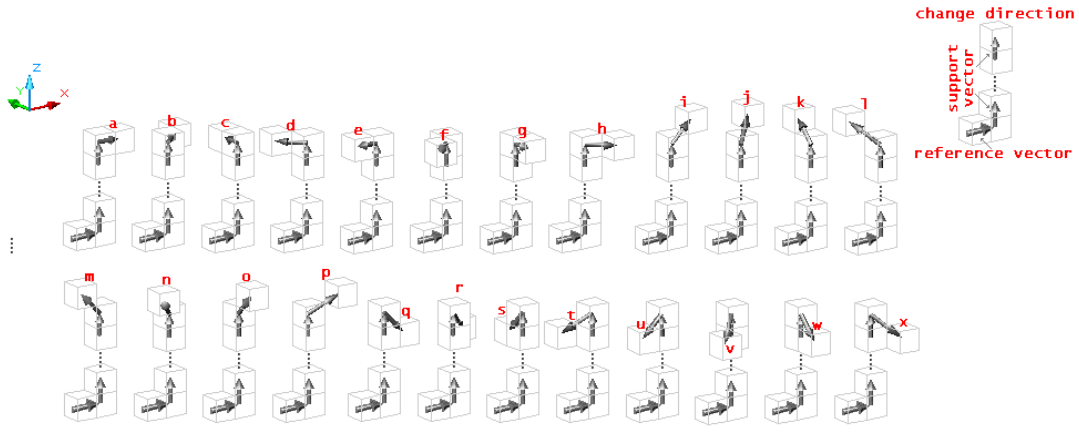
$$P_{\mathcal{B}} = \{\langle 1,0,1\rangle, \langle 1,0,0\rangle, \langle 1,1,0\rangle, \langle 1,0,1\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,1\rangle, \langle 0,-1,1\rangle, \langle 0,-1,1\rangle, \langle 1,0,0\rangle\}$$

its $F26$ coding is $P_{F26} = \{i, a, b, i, a, a, a, i, o, o, a\}$ or, simplifying the notation, the last equality can be rewritten in a familiar string notation: $S_{F26} = iabiaaaiooa$.

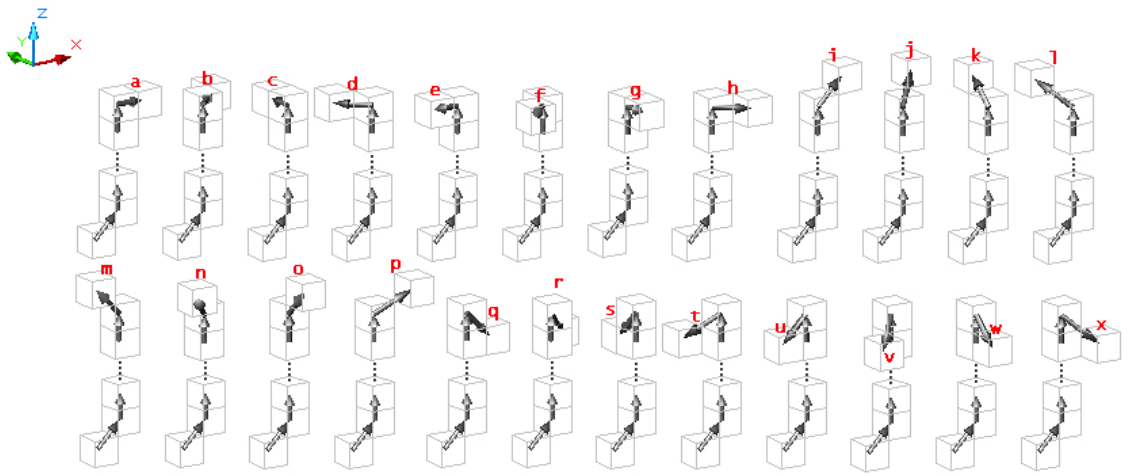
3.2 Proposing a three-dimensional relative chain code (3DRC)

In this section, we propose a new method to cover a 3D discrete curve that is independent of the Cartesian plane. What we are going to do now is to propose a relative coding, which, among other properties, like mirroring invariance and high compression levels that we explain later, is also invariant under translation and rotation transformations.

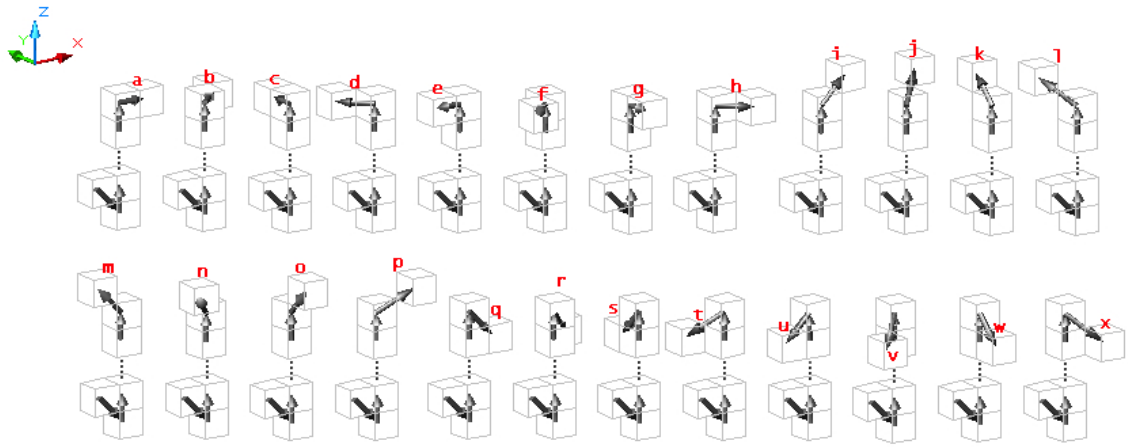
Because of **Definition 9**, a simple path is composed of GEPs, so to encode a simple path, we encode each of its GEPs. We are going to prove that any GEP is equivalent to one element of the groups in Fig. 8.



Group 1: *ref* and *supp* make an angle of 90



Group 2: ref and $supp$ make an angle of less than 90



Group 3: ref and $supp$ make an angle of more than 90

Figure 8: There are 72 possible change directions for any 3D discrete curve, divided in three groups, depending of the angle between reference and support vector.

Note that an arbitrary GEP is not always equal to one of the elements of the groups, but performing three vox rotations and a usual rotation, at most, we obtain that any arbitrary GEP is equivalent to one of the elements of the groups. The following result tells us how to obtain the element of the group which is equivalent to an arbitrary GEP; it also shows what symbol corresponds to a GEP.

Something very important to note at this moment, is that to assign a symbol to a configuration of vectors $ref, supp$ and $chng$, one should only use the norms $\|ref\|, \|supp\|$ and $\|ref - supp\|$. This fact, insignificant at first glance, is the cause for our code to be invariant under rotations.

Theorem 1. Any GEP is equivalent to an element of the set of 72 different voxel arrays in Fig. 8.

Proof.

The main difference between an elemental path and a GEP is mainly that the GEP could be longer, however they have the same shape and are equivalent elemental paths. So, let us first see how to encode an elemental path. The generalized case is a consequence. Let $P = \{v_1, v_2, v_3, v_4\}$ be an elemental path, and let $P_{\mathcal{B}} = \{ref, supp, chng\}$ be its basis. Without loss of generality, the demonstration consists of making a usual rotation and vox rotations to the vector *supp* to obtain the position $\langle 0, 0, 1\rangle$ and thus we assign a symbol. The following algorithm tells us what to do for leaving this vector at the desired position.

Let us assume that $ref = \langle x_1, y_1, z_1\rangle$ and $supp = \langle x_2, y_2, z_2\rangle$,

if $\|supp\| = 1$, let us apply to $ref, supp$ and $chng$ a **usual rotation**, /* Of course, there are six options */

- if $supp = \langle 1, 0, 0\rangle$, around the y axis of -90 degrees.
- if $supp = \langle -1, 0, 0\rangle$, around the y axis 90 degrees.
- if $supp = \langle 0, 1, 0\rangle$, around the x axis 90 degrees.
- if $supp = \langle 0, -1, 0\rangle$, around the x axis -90 degrees .
- if $supp = \langle 0, 0, 1\rangle$, is already in the desired position and do not apply more rotations.
- if $supp = \langle 0, 0, -1\rangle$, around $thex$ axis 180 degrees.

if $\|supp\| = \sqrt{2}$, let us apply to $ref, supp$ and $chng$ a **vox rotation** of 45, /* including signs, rotation is around one of the six perpendicular directions: */

- if $x_2 = 0$, around $\langle y_2, 0, 0\rangle$,
- if $y_2 = 0$, around $\langle 0, z_2, 0\rangle$,
- if $z_2 = 0$, around $\langle 0, 0, x_2\rangle$,

and we return to the case when $\|supp\| = 1$.

if $\|supp\| = \sqrt{3}$, apply to $ref, supp$ and $chng$ a **vox rotation** of 45,/* there are three options */

- if $\|ref\| = 1$, around ref,
- if $\|ref\| = \sqrt{2}$,
  - if $\|ref - supp\| = 1$, around $ref - supp$,
  - if $\|ref - supp\| = \sqrt{5}$,
    - if $x_1 = x_2$, around $\langle x_1, 0, 0\rangle$,
    - if $y_1 = y_2$, around $\langle 0, y_1, 0\rangle$,
    - if $z_1 = z_2$, around $\langle 0, 0, z_1\rangle$,
- if $\|ref\| = \sqrt{3}$,
  - if $\|ref - supp\| = 2$, around $(ref - supp)/\|ref - supp\|$,
  - if $\|ref - supp\| = \sqrt{8}$,
    - if $x_1 = x_2$, around $\langle x_1, 0, 0\rangle$,
    - if $y_1 = y_2$, around $\langle 0, y_1, 0\rangle$,
    - if $z_1 = z_2$, around $\langle 0, 0, z_1\rangle$,

and we return to the case when $\|supp\| = \sqrt{2}$.

A time that $supp = \langle 0, 0, 1\rangle$ proof is almost ready. So, if θ is the angle that goes from $\langle x_1, y_1, 0\rangle$ to the projection $\langle 1, 0, 0\rangle$, applying to $ref, supp$ and $chng$ a vox rotation of $-\theta$ around z, we obtain $ref = \langle 1, 0, z_1\rangle$, $supp = \langle 0, 0, 1\rangle$, and thus $ref, supp$ and $chng$ have the form of some element of the basis. The symbol that accompanies such an element is the symbol corresponding to the elemental path P. We know that $z_1 \in \{-1, 0, 1\}$ (three elements) and $chng \in \mathcal{B} \setminus \{\langle 0, 0, 1\rangle, \langle 0, 0, -1\rangle\}$ (24 vectors); for this reason, $3 \times 24 = 72$, plus y symbol, there are 73 symbols generated.

if $z_1 = 0$, ref and $supp$ make an angle of 90 degrees, and **Group 1** is obtained

if $z_1 = -1$, ref and $supp$ make an angle of less than 90 degrees, and **Group 2** is obtained

if $z_1 = 1$, ref y $supp$ make an angle of more than 90 degrees, and **Group 3** is obtained

Everything is now in the behavior of $chng$. Depending on this vector, we chose the corresponding symbol. The symbol associated to $chng$ in **Group 1**, **Group 2** or **Group 3**, is the corresponding symbol to the elemental path P.

We introduce the symbol y to be used to label a path that has no changes of direction. Thus, if $P = \{v_1, v_2, \cdots, v_m\}$ is a GEP, with $m > 4$, its GEP is of the form $P_{\mathcal{B}} = \left\{b_1, b_2^{m-3}, b_3\right\}$, so, the code to be associated to the GEP P is

$$\gamma = \underbrace{yy\cdots y}_{m-4}\alpha, \tag{2}$$

where α is the symbol associated to the vectors $ref, supp, chng$ of P.

□

Depending on the configuration reached by $ref, supp$ and $chng$, we assign a symbol, and **Theorem 1** tells us that we have 72 possible direction changes to represent any 3D discrete curve.

The example given in Fig. 9 shows how to assign a symbol to an elemental path. As can be seen, the symbol path is p.

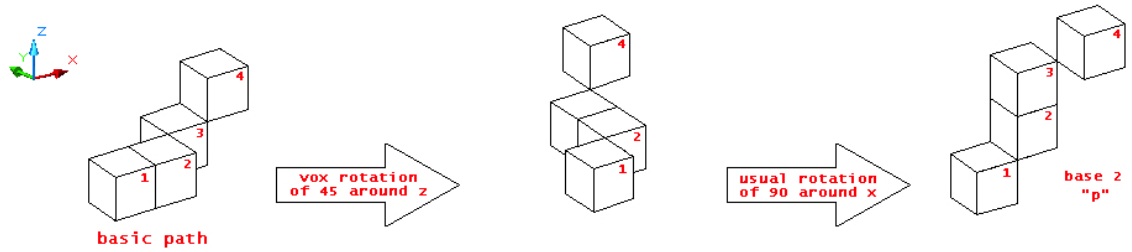


Figure 9: A set of discrete rotations has to be performed to obtain a symbol in a four-voxel array.

4 Coding and decoding

In this section, we consider paths in graphs. However, by observing their voxel reconstruction through the paths with no loops, we particularly consider trees. We then explain how to decode the 3DRC chain codes.

4.1 Coding a simple path

Once we know how to assign a symbol to an elemental path or to a GEP, we now encode a simple path.

Theorem 2 The code of a simple path $P = \{v_1, v_2, \ldots, v_n\}$ is of the form:

$$\underbrace{yy\cdots y}_{m_1-1}\gamma_1^*\gamma_2\cdots\gamma_k\underbrace{yy\cdots y}_{m_{k+2}-1}, \tag{3}$$

where, γ_1^* is the coding of the first GEP P^1, and γ_i is the symbol associated to the GEP P^i, $i = 2, 3, \cdots, k$.

Proof.

By **Definition 9** we know that a basis of a simple path P is of the form

$$P_{\mathcal{B}} = \left\{b_1^{m_1}, b_2^{m_2}, \ldots, b_{k+1}^{m_{k+1}}, b_{k+2}^{m_{k+2}}\right\}, \tag{4}$$

where the sets

$$P_{\mathcal{B}}^1 = \{b_1, b_2^{m_2}, b_3\}, \qquad P_{\mathcal{B}}^2 = \{b_2, b_3^{m_3}, b_4\}, \qquad \cdots \qquad , P_{\mathcal{B}}^k = \left\{b_k, b_{k+1}^{m_{k+1}}, b_{k+2}\right\} \tag{5}$$

are GEP bases, Theorem 1 tells us how to associate a symbol to a GEP.

To reduce the 72 symbols, the first GEP that appears deserves a special treatment. Moreover, decoding plays a very important role, so it is necessary to obtain exactly the same form of this path, and to achieve this, we use the **Lemma 1**.

After obtaining the necessary information of P^1, as indicated by the **Lemma 1**, the code associated with the simple path P is

$$\underbrace{yy\cdots y}_{m_1-1}\gamma_1^*\gamma_2\cdots\gamma_k\underbrace{yy\cdots y}_{m_{k+2}-1} \tag{6}$$

where

γ_1^* is the coding of the first GEP P^1 (**Lemma 1**), and

γ_i is the symbol associated to the GEP P^i, $i = 2, 3, \cdots, k$ (**Theorem 1**)

□

Lemma 1. The coding of the first GEP of a simple path is of the form $\gamma_1^* = N\underbrace{yy\cdots y}_{n-4}\alpha N_1N_2N_3N_4$, where N and N_i are integers, y is the code to go straight ahead, and α is one of the symbols of Fig. 8.

Proof. Let us assume that $P_1 = \{v_1, v_2, \ldots, v_m\}$, is the first GEP to be codified, and let $P_{\mathcal{B}}^1 = \left\{b_1, b_2^{m-3}, b_3\right\}$ be the GEP basis of P_1.

If $\|supp\| < \sqrt{3}$ we write $N_4 = 0$,

otherwise it means that $\|supp\| = \sqrt{3}$. After using part of the algorithm of **Theorem 1** when $\|supp\| = \sqrt{3}$, following our method we should do a vox rotation of 45, so, $\|supp\| = \sqrt{2}$,.

if vox rotation was around x, we write $N_4 = 1$.

if vox rotation was around $-x$, we write $N_4 = 2$.

if vox rotation was around y, we write $N_4 = 3$.

if vox rotation was around $-y$, we write $N_4 = 4$.

if vox rotation was around z, we write $N_4 = 5$.

if vox rotation was around $-z$, we write $N_4 = 6$.

Once $\|supp\| = \sqrt{2}$, if we assume $supp = \langle x_2, y_2, z_2\rangle$, taking θ_1 as the angle that goes from the projection $\langle x_2, y_2, 0\rangle$ to $\langle 1, 0, 0\rangle$ with $0 \leq \theta_1 < 360$, we have:

$$\mathbf{Rot_z}\left(supp, \theta_1\right) = \langle 1, 0, z_2\rangle, \tag{7}$$

and taking θ_2 as the angle that goes from the projection $\langle 1, 0, z_2\rangle$ to $\langle 0, 0, 1\rangle$, with $0 \leq \theta_2 < 360$, we have:

$$\mathbf{Rot_y}\left(\mathbf{Rot_z}\left(supp, \theta_1\right), \theta_2\right) = \mathbf{Rot_y}\left(\langle 1, 0, z_2\rangle, \theta_2\right) = \langle 0, 0, 1\rangle. \tag{8}$$

So, we write:

$$\begin{aligned} ref &:= \mathbf{Rot_y}\left(\mathbf{Rot_z}\left(ref,\theta_1\right),\theta_2\right), \\ supp &:= \mathbf{Rot_y}\left(\mathbf{Rot_z}\left(supp,\theta_1\right),\theta_2\right) = \langle 0,0,1\rangle, \\ chng &:= \mathbf{Rot_y}\left(\mathbf{Rot_z}\left(chng,\theta_1\right),\theta_2\right). \end{aligned}$$

Finally we make again a vox rotation around z to obtain some element of the basis. If now we assume that $ref = \langle u_1, u_2, u_3\rangle$, taking θ_3 like the angle that goes from the projection $\langle u_1, u_2, 0\rangle$ to $\langle 1,0,0\rangle$, with $0 \leq \theta_3 < 360$, we have that

$$\mathbf{Rot_z}\left(ref,\theta_3\right) = \langle 1,0,u_3\rangle, \tag{9}$$

so, we make

$$\begin{aligned} ref &:= \mathbf{Rot_z}\left(ref,\theta_3\right) = \langle 1,0,z_1\rangle \\ supp &:= \mathbf{Rot_z}\left(supp,\theta_3\right) = \langle 0,0,1\rangle \\ chng &:= \mathbf{Rot_z}\left(chng,\theta_3\right) \end{aligned}$$

From this procedure, we conclude that vectors $ref, supp, chng$ belong to some vector basis N, with $N = 1$, $N = 2$ or $N = 3$. So, the code associated to the first GEP P^1 (and thus to the basis of the GEP $P^1_{\mathcal{B}} = \left\{b_1, b_2^{m-3}, b_3\right\}$) of the simple path is

$$\gamma_1^* = N\underbrace{yy\cdots y}_{n-4}\alpha N_1 N_2 N_3 N_4 \tag{10}$$

where

α is the symbol associated to the new (after rotations) $ref, supp, chng$ vectors,

$N_1 = \theta_1/45$,

$N_2 = \theta_2/45$,

$N_3 = \theta_3/45$.

□

4.2 Decoding a simple path

In the previous sections, we found how to encode any simple path. Now, we will focus on the inverse problem, *i.e.*, given a code, how do we recover the sequence of voxels that compose a simple path?

It should be noted that given a chain code

$$\underbrace{yy\cdots y}_{m_1}\gamma_1^*\gamma_2\cdots\gamma_k\underbrace{yy\cdots y}_{m_{k+2}} \tag{11}$$

we obtain the basis path again, and the simple path should be recovered.

As we mentioned in the previous section, the first elemental path of the simple path plays a very important role in the encoding, because we start from it.

Theorem 3. Decoding γ_1^*, we obtain the first GEP.

Proof. We take vectors $ref, supp$ and $chng$ depending on the symbol γ of the N group of Fig. 8.

As numbers N_1, N_2 and N_3 arise in the coding, we now simply do

$$\begin{aligned} ref &:= \mathbf{Rot_z}\left(\mathbf{Rot_y}\left(\mathbf{Rot_z}\left(ref,-\theta_3\right),-\theta_2\right),-\theta 1\right) \\ supp &:= \mathbf{Rot_z}\left(\mathbf{Rot_y}\left(\mathbf{Rot_z}\left(supp,-\theta_3\right),-\theta_2\right),-\theta 1\right) \\ chng &:= \mathbf{Rot_z}\left(\mathbf{Rot_y}\left(\mathbf{Rot_z}\left(chng,-\theta_3\right),-\theta_2\right),-\theta 1\right) \end{aligned}$$

where

$$\theta_1 = N_1 * 45, \theta_2 = N_2 * 45, \theta_3 = N_3 * 45, \tag{12}$$

and if $N_4 \neq 0$, we do not rotate vectors $ref, supp$ and $chng$ anymore. On the contrary, we make a vox rotation of -45 to the redefined $ref, supp$ and $chng$,

if $N_4 = 1$, we make a rotation around x

if $N_4 = 2$, we make a rotation around $-x$

if $N_4 = 3$, we make a rotation around y

if $N_4 = 4$, we make a rotation around $-y$

if $N_4 = 5$, we make a rotation around z

if $N_4 = 6$, we make a rotation around $-z$

Thus, we obtain three vectors of the basis path: $b_1 := ref, b_2 := supp$ and $b_3 := chng$. However, since $\gamma_1^* = N\underbrace{yy\cdots y}_{m}\alpha N_1N_2N_3N_4$, (Eq. 10), our basis of the GEP is

$$P = \left\{b_1, b_2^{m+1}, b_3\right\}. \tag{13}$$

Taking $m_2 = m + 1$, P becomes

$$P = \left\{b_1, b_2^{m_2}, b_3\right\}. \tag{14}$$

□

Now that we have the decoding of the first GEP, the rest is easy.

Theorem 4. A chain code of the form $\underbrace{yy\cdots y}_{m_1}\gamma_1^*\gamma_2\cdots\gamma_k\underbrace{yy\cdots y}_{m_{k+2}}$ can be decoded to obtain the original simple path.

Proof. Since we know how to decode γ_1^*, we start to form our simple basis path. By **Theorem 2**, γ_1^* has been associated with the following simple basis path:

$$P = \left\{b_1, b_2^{m_2}, b_3\right\}. \tag{15}$$

The coding starts with $\underbrace{yy\cdots y}_{m_1-1}$, this means that the start of the path is a straight line; thus, P becomes

$$P = \left\{b_1^{m_1}, b_2^{m_2}, b_3\right\}. \tag{16}$$

The rest of the $\gamma_i's$ tell us the path that the simple basis path must follow. By (2), γ_2 is of the form $\gamma_2 = \underbrace{yy\cdots y}_{h}\alpha$ Taking $m_3 = h + 1$, and by Eq. (2), P becomes $P = \left\{b_1^{m_1}, b_2^{m_2}, b_3^{m_3}\right\}$. Let $ref = b_2$ and $supp = b_3$. By the algorithm of **Theorem 1**, there are rotations r_1, r_2, r_3, r_4 such that

$$r_4\left(r_3\left(r_2\left(r_1\left(ref\right)\right)\right)\right) = \langle 1, 0, z_1\rangle \tag{17}$$

$$r_4\left(r_3\left(r_2\left(r_1\left(supp\right)\right)\right)\right) = \langle 0, 0, 1\rangle \tag{18}$$

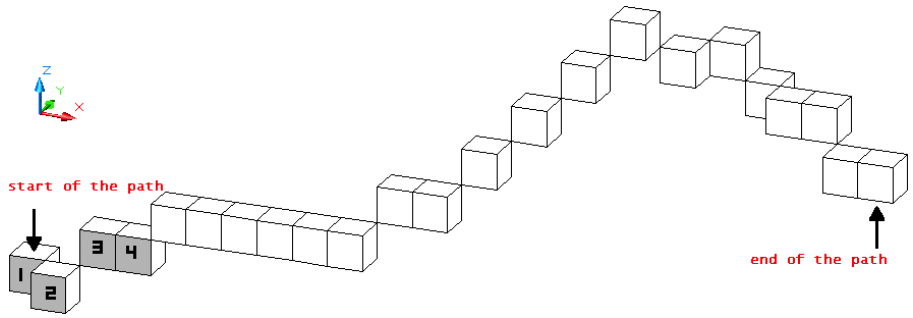


Figure 10: Simple path. Gray face voxels: first GEP.

$$P_{\mathcal{B}} = \{\langle 1,-1,0\rangle, \langle 1,1,1\rangle, \langle 1,0,0\rangle, \langle 1,0,1\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,0,0\rangle, \langle 1,1,1\rangle, \langle 1,0,0\rangle, \langle 1,1,1\rangle, \langle 1,1,1\rangle, \langle 1,1,1\rangle, \langle 1,1,1\rangle, \langle 1,1,-1\rangle, \langle 1,1,0\rangle, \langle 1,0,-1\rangle, \langle 1,-1,0\rangle, \langle 1,0,0\rangle, \langle 1,-1,-1\rangle, \langle 1,0,0\rangle\}.$$

if $z_1 = 0$, let *chng* be the vector of change associated to the symbol α in basis 1

if $z_1 = 1$, let *chng* be the vector of change associated to the symbol α in basis 2

if $z_1 = -1$, let *chng* be the vector of change associated to the symbol α in basis 3

The new element of the simple basis path is

$$b_4 = -r_1\left(-r_2\left(-r_3\left(-r_4\left(ref\right)\right)\right)\right), \tag{19}$$

and thus P becomes

$$P = \{b_1^{m_1}, b_2^{m_2}, b_3^{m_3}, b_4\}. \tag{20}$$

Taking $ref = b_3$ and $supp = b_4$, we obtain b_5, and continuing this way we arrive to the basis of the elemental path:

$$P = \left\{b_1^{m_1}, b_2^{m_2}, b_3^{m_3}, \cdots, b_{k+1}^{m_{k+1}}, b_{k+2}\right\}. \tag{21}$$

Finally, since the coding ends with $\underbrace{yy\cdots y}_{m_{k+2}-1}$, once again, by Eq. (2), the basis of the elemental path that we look for is

$$P = \left\{b_1^{m_1}, b_2^{m_2}, b_3^{m_3}, \cdots, b_{k+1}^{m_{k+1}}, b_{k+2}^{m_{k+2}}\right\}. \tag{22}$$

□

We already know how to encode and decode simple paths. Let us see the next example:

Example. What is the code of the simple path given in Fig. 10?

To answer this question, we have to identify the GEPs of the path and then use **Theorem 1**. The first is in gray in Fig. 10. To encode the first GEP, let us use **Lemma 1**; in addition, we verify that we perform, at most, 4 rotations to encode a GEP. Visiting the GEPs of the curve we can find the code associated to Figure 10 is: **1j8711jiyyyypiiyyyjipkjji**.

To decode, we have to identify the first GEP and then use **Theorem 4**, which builds the basis P of the simple path we are decoding. The first GEP is **1j8711**, and the information that it contains as follows,

$$\underbrace{\mathbf{1}}_{N}\ \underbrace{\mathbf{j}}_{symbol}\ \underbrace{\mathbf{8}}_{N_1}\ \underbrace{\mathbf{7}}_{N_2}\ \underbrace{\mathbf{1}}_{N_3}\ \underbrace{\mathbf{1}}_{N_4} \tag{23}$$

So, we start with GEP **j** of group **1** of Fig. 8.

4.3 Avoiding loops

Despite curve-skeletons could have loops, we can try them as trees.

Let us start by defining a loop in trajectories of voxels.

Definition 15 A *loop* is a path $P = \{v_1, v_2, \cdots, v_n\}$ in which v_1 is adjacent to v_n.

We do not only consider loops, but also paths which contain loops.

Definition 16 A *path-1-loop* is a set of connected voxels P in which a subset, say P^1, forms a loop. Also the set $P \setminus P^1$ has no another loop. In this case we say P *has 1 hole*.

Example given in Fig. 11 is a path-1-loop.

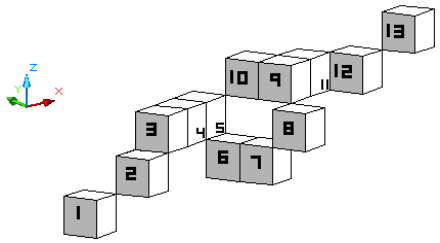


Figure 11: A path-1-loop. Note that in $P^1 = \{v_5, v_6, v_7, v_8, v_9, v_{10}\}$ and in $P \setminus P^1 = \{v_1, v_2, v_3, v_4, v_{11}, v_{12}, v_{13}\}$. There are no more loops.

In general, a path-s-loop is a path with s different holes:

Definition 17 A *path-s-loop* is a connected set of voxels P in which s subsets of P, say $P^1, \cdots, P^s$, different between them (do not close the same region in $\mathbb{R}^3$), form a loop (each of them). In addition, in the set $P \setminus \bigcup P^i$ there is no other loop. In this case we say P *has s holes*.

Example. Fig. 12 presents a path-2-loop.

In this case

$P^1 = \{v_5, v_6, v_7, v_8, v_9, v_{10}\}$ is a loop

$P^2 = \{v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{19}, v_{20}, v_{21}\}$ is a loop

In $P \setminus \left(P^1 \bigcup P^2\right) = \{v_1, v_2, v_3, v_4, v_9, v_{11}, v_{12}, v_{13}, v_{22}, v_{23}, v_{24}, v_{25}\}$ there are no more loops.

Although visually a loop appears, an advantage to work with voxel representation is that a path can be tried it as a tree. For example the shape of Fig. 13 can be seen as a tree in Fig. 12(a).

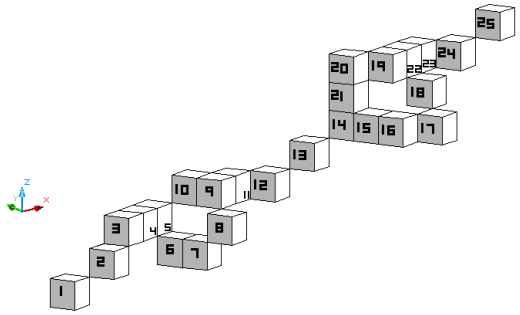


Figure 12: Example of path-2-loops

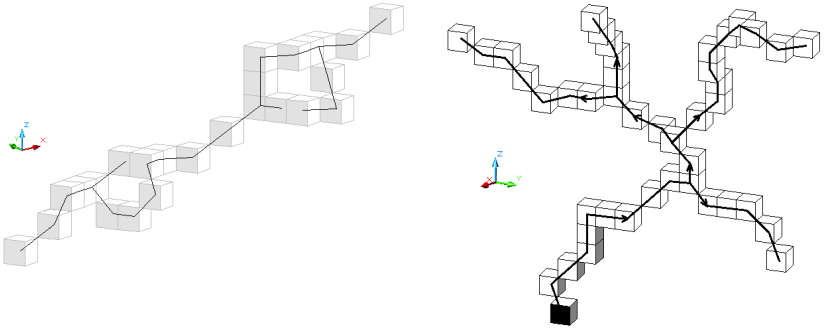


Figure 13: 3D trees: (a) A tree associated to a set of the connected voxels of Fig. 12, (b) a directed tree. The black face voxel indicates the starting node, whereas gray face voxels its first GEP.

4.4 Coding trees

We encoded 3D tree objects. In his work, Bribiesca [13] considers trees whose branches are formed solely by voxels that are connected by faces. As a natural generalization to his work, the branches of the trees which we consider now consist of connected voxel faces, edges and vertices. Just as in theory of graphs [30], we have that the number of edges of a graph is equal to the number of nodes less one.

For *starting position*, consider the number of directed paths in a graph with n_l leaves. Of course, there are $n_l(n_l - 1)$ different directed paths. Choose the largest directed path. Finally chose the starting position of the leaf corresponding to the first letter in the lexicographical order.

For the codification of this type of structures, let us use two more symbols: "(", and ")", we also provide the following definitions, and the algorithm that comes later.

The first definition of this section helps us to know what path to take when we find ourselves in a node; the lexicographical order.

Definition 18 *Lexicographical ordering* ($<_{lex}$). Let Y_1, Y_2 be two characters.

$$Y_1 <_{lex} Y_2 \qquad \text{iff} \qquad 0 < \text{Ascii}(Y_2) - \text{Ascii}(Y_1).$$

Ascii(Y) represents the Ascii code of the character Y.

For next definitions, to slightly shorten the notation, let us write "v is face" instead "v is face connected". Analogously for edge and vertex.

Definition 19 Let us denote as $Ady(v)$ the number of voxels adjacent to v that have not been visited.

In the following definition, we see when a voxel is representing a node of a tree. One would think that if a tree is been visiting, as soon as we reach a voxel whose $Ady(v) \geq 2$, it would mean that we are at a node, but this is not the case, while a voxel has $Ady(v) = 2$, sometimes we will not consider v a node. The reason for these conditions is simple. If a voxel is adjacent to two others, in such a way that one is face and the other is edge (or vertex), it is because we cannot traverse the face first and then the edge (vertex).

Definition 20 Let us assume $Ady\,(v) = 2$ and v_1 y v_2 are non visited voxels adjacent to v. If v_1 is adjacent to v_2, and also

1. v is face with v_1 and v is edge with v_2, or
2. v is edge with v_1 and v is vertex with v_2, or
3. v is face with v_1 and v is vertex with v_2,

therefore v does not represent a node of the tree, and the next voxel to visit is v_1. In any other case, if $Ady\,(v) \geq 2$ and does not hold any of the above conditions, then we say that v is a *node*.

The algorithm for encoding trees shall be as follows:

step 1) Choose a voxel with $Adj\,(v) = 1$ (starting node). Use the voxels to follow the starting node to form the first GEP and encode it using **Lemma 1.** Save *supp* and *chng* and delete the first GEP of the tree, except the last voxel, and call it voxel v_1. If v_1 is a node, go to step 3; otherwise go step 2.

step 2) Go to the next voxel and name it v_2, re-define $ref := supp$, $supp := chng$ and $chng := c_2 - c_1$, where c_1 and c_2 are the center of v_1 and v_2, respectively. Delete v_1 and re-name v_2 as v_1. Encode using $ref, supp$ and $chng$. If v_1 is a node, go to step 3; if there is not a next voxel (call v_1 *final node*) go to step 4; otherwise go to step 2.

step 3) Type an open parenthesis in the code "(" and save v_1, ref and *sup* in a list $\mathcal{L}$. The rest of voxels should have a code in the sense of **Definition 18**. Go to step 2.

step 4) If there are no more voxels, finish. Otherwise type in the code a closed parenthesis ")" and take v_1, ref and *supp* as the last in the list $\mathcal{L}$. Delete this information in $\mathcal{L}$. If the current v_1 remains as a node, go to step 3. Otherwise go to step 2.

4.5 Decoding trees

Once we know how to encode a tree, the decoding is almost immediate. The algorithm to decode trees is as following.

step 1) Start decoding as if it were a simple path.

step 2) Continue decoding as if it were a simple path, if there is an open parenthesis, go to step 3. If there is a closed parenthesis, go to step 4. If there are no more characters in the code, decoding is finished.

step 3) Save the coordinates of the last created voxel, and of the current vectors ref and *supp* in a list $\mathcal{L}$. Go to step 2.

step 4) Return to the last voxel in the list $\mathcal{L}$, and consider that the current vectors ref and *supp* are also the last introduced in list $\mathcal{L}$. Delete this information from $\mathcal{L}$. Go to step 2.

5 Invariant under transformations

The proposed 3DRC code has several interesting properties. In this section, we demonstrate that the proposed code is invariant under translation, rotation and mirror transformations.

5.1 Invariant under translation transformations

When any curve is rigidly moved from one place S to another S', the coordinates of the voxels change in the following way, $v' = v + \alpha$, where α represents the displacement of the coordinates.

However, the 3DRC code remains invariant, since, despite making such displacement, the basis path remains unchanged, *i.e.*, $P'_{\mathcal{B}} = P_{\mathcal{B}}$. Because of this, exactly the same code is built before or after carrying out a translation of the discrete curve.

5.2 Invariant under rotation transformations

Corollary 1 3DRC code is invariant under rotation transformations.

Proof. Demonstration of this corollary follows from Theorem 1 and Definition 8, and if we apply the rotations given by Definition 11 in which a change of direction by a GEP in the 3D Euclidean space is given, not by Cartesian coordinates, but by *reference* and *support* vectors, independently whether they are oriented in the discrete 3D space. □

As an example, consider Fig. 14.

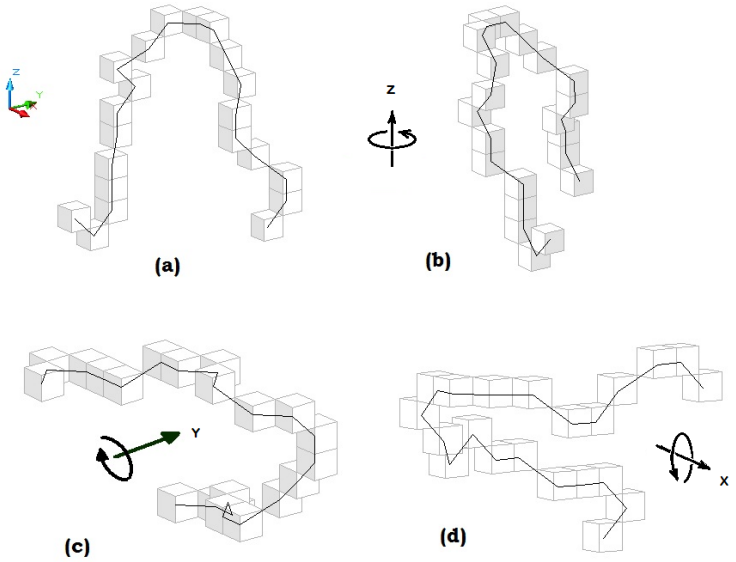


Figure 14: A Simple path rotated around the axis: (a) Original path, (b) rotated around axis Z, (c) rotated around axis Y, and (d) rotated around axis X.

The original code of the path is:

S = 1**m**6780**yjipahnjloykpnopl**,

whereas the codes of the path rotated around X, Y and Z axis are: S_x = 1**m**2740**yjipahnjloykpnopl**, S_y = 1**m**7660**yjipahnjloykpnopl**, and S_z = 1**m**4780**yjipahnjloykpnopl**, respectively.

By Lemma 1, the first GEP has to be chosen from the groups of elements in Fig. 8. As can be seen, code symbols are the same, except possibly the four numbers after the first symbol. As explained in Lemma 1, there is a group of rotations to go from an arbitrary orientation of the first GEP to a single element of Fig. 8. So, the code is determined in a unique way, independently of orientation.

5.3 Invariant under mirror transformations

Another property of the proposed code is its invariance under mirror transformations.

Symbol	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}
A	a	b	c	d	e	f	g	h	i	j	k	l
A^M	a	h	g	f	e	d	c	b	i	p	o	n
Symbol	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}	s_{21}	s_{22}	s_{23}	s_{24}
A	m	n	o	p	q	r	s	t	u	v	w	x
A^M	m	l	k	j	q	x	w	v	u	t	s	r

Table 1. Interchangeable symbols when mirror transformation occurs.

A mirror transformation is carried out when the coordinates of the curve mirror in the system S' relate to the system S in the following way: $v'(x', y', z') = v(x, y, -z)$. In this case, the mirror is the X-Y plane. On the other hand, if $v'(x', y', z') = v(x, -y, z)$, the mirror is the X-Z plane. And, if $v'(x', y', z') = v(-x, y, z)$, the mirror is the Y-Z plane. The following theorem arises.

Theorem 5. The 3DRC code is invariant under mirror transformations through X-Y, Z-Y and Z-X planes.

Proof.

When a curve is reflected on any of planes X-Y, Y-Z or Z-X, a mirror transformation given by the coordinates of the voxels comes out. In the case of transformation of mirror under the plane X-Y, coordinates x, y remain the same, but $z' = -z$. Under X-Z, we have that x and z are equal, but $y' = -y$; and in the case of mirror under Y-Z, we have $x' = -x$. It is easy to see that when any element of the Fig. 8 is taken, some other element is its reflection. Thus, for example, when the element labeled by the symbol 'b' is reflected on any of the three mentioned planes, the reflection corresponds to the element labeled by the symbol 'h'. This analysis can be done with each of the elements of Fig. 8. The outcome can be seen in Table 1, *i.e.*, the invariant set to make 3DRC the same under mirror transformation can be obtained by the assignments given in Table 1.

Note that only six of the 24 symbols remain the same in the mirror transformation. The symbols are: a,e,i,m,q, and u. The rest have to be interchangeable as Table 1 suggests.

This transformation table is valid when the object is reflected on the X-Y, Y-Z or X-Z planes. □

See Fig. 15 for an example of mirror invariance.

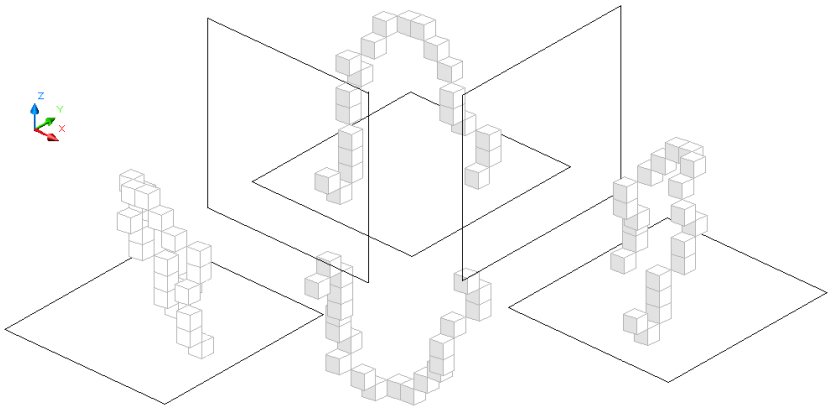


Figure 15: An irregular curve reflected on planes X-Y, Y-Z and X-Z.

The chain code of the curve in Fig. 15 is: S =1m6780yjipahnjloykpnopl. When reflected on the X-Y plane, the chain code becomes: S_z= 1m6540ypijablpnkyojlkjn. When reflected on the X-Z plane, the chain code becomes: S_y = 1m2780ypijablpnkyojlkjn. Finally, if the original curve is reflected on the Y-Z plane, the chain code is: S_x = 1m6780ypijablpnkyojlkjn. By taking into account the assignments of Table 1, we can easily verify that the original S code is invariant under these mirror transformations.

6 Applications and comparison with other codes

In this section we present two applications that can be carried out to solve problems of the real world: representations of curve-skeletons and, also, representations of Digital Elevation Models (DEMs). On the other hand we discuss the advantage of our proposed 3DRC code regarding the most utilized in literature: 5OT and F26.

6.1 Curve-skeletons

As can be observed from the nature of 3D chain codes, these are one-dimensional (1D) structures suitable to represent tridimensional curves. On the other hand, curve-skeletons are, also, 1D representations of 3D objects. Research in 3D curve-skeleton tackle two general problems: find robust algorithms to obtain curve-skeleton representations and propose similarity measures for object based recovery.

The extraction, analysis and use of curve-skeletons is a very active field in computer vision [33]. A vast number of papers have been written about curve-skeleton representations for real applications to different kind of problems, including, among others: virtual navigation, registration, morphing, scientific analysis, classification, inverse kinematics [46], or for applications in virtual colonoscopies [34] and virtual endoscopies [35]. In animation, there is a vast number of articles, to mention a few: [36], [37], [38] and [39].

In fact, there is no a unique curve-skeleton algorithm for general 3D object applications, but to solve specific problems by using particular data sets.

Also, curve-skeletons have been used to establish measures of similarity for 3D shapes [41, 42, 43, 51].

Many papers have presented algorithms to obtain, as far as possible, the best curve-skeletons that work as appropriate descriptors of voxelized objects in order to preserve the original topology by thinning, deleting and pruning algorithms [47, 48], or by using the mesh data and obtaining point clouds [49, 50]. There are other efforts to preserve original shape [52] as much as possible.

An application of our chain code in 3D curve-skeleton representation can be achieved. Most of the papers above-mentioned, utilize voxel data to represent the curve-skeletons, and the different algorithms are applied over such information. An alternative is to utilize our proposed 3D chain code. Whereas the other representations employ complicated relationships between neighbor voxels, our propose method consider symbolic representation which handles local data and takes care of the curve-skeleton shape. Every discrete movement along the curve is encoded by one of the 25 symbols of the 3DRC code. This information is an alternative to voxel representation. Therefore, knowing symbol distribution, either in the whole or in the different parts of the curve-skeleton, its shape can be interpreted.

Fig. 16 shows a sample of human and animal-like models that can be used to make shape analysis or to find similarity measures, whereas Fig. 17 shows other kind of 3D models, representing manufactured objects, that can be also classified in terms of their shapes and topology. There are many websites to download 3D models, see for example the Stanford Computer Graphics Laboratory site: http://graphics.stanford.edu/data/.

To test our method, we used the curve-skeletons of the 3D above-mentioned objects. In our work, we used a free access voxelizer and skeletonizer [40]. The voxelizer is called *binvox*, whereas the skeletonizer is called *thinvox*. Binvox uses the count method and parity of Nooruddin and Turk ray [44], who presented a way to manipulate polygonal models and convert them to volumetric representations. Once volumetric objects are obtained, the thinning algorithm is applied. Thinvox supports the directional thinning method described by Klmn Palgyi and Attila Kuba [45].

Fig. 18 presents an example of a Lion curve-skeleton object and the other representation proposed here: its chain code. On the other hand, Fig. 19 shows both, an example of an animal and a manufactured-like curve-skeletons. We applied our method on 10 different 3D models to obtain their curve-skeletons, as shown in Table 2, which contains the sizes, in terms of the number of rows, columns and slices the binary files require, and number of voxels obtained in the skeletonization.

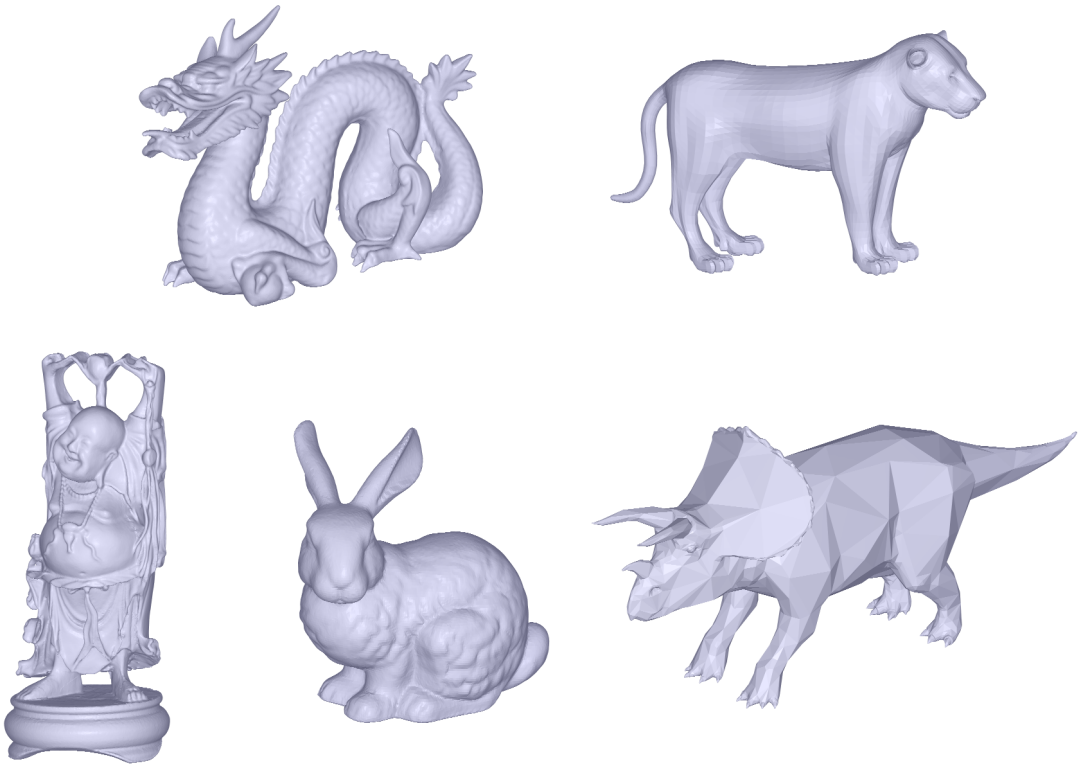


Figure 16: A sample of 3D models.

Our proposal code is well sounded as a new alternative to represent curve-skeletons, because the information given represents faithfully the shape, we consider the code as a descriptor or as a "word". From example, it can be read from the code, all the digital straight lines contained, also the nodes (articulated parts) by reading the parenthesis symbols.

Tables 3 and 4, rich in information about object shape, present symbol frequencies. From Table 3 and 4 can be obtained, also, symbol probabilities, the number of nodes of the curve-skeletons and histograms to be compared in order to look for similarity measures. A further work could exploit this kind of data by considering parts of the objects, for example those between nodes, matching symbols, look for discrete straight lines, and so forth. Next section gives an example of shape analysis of this kind of codes.

Of course, in literature there are other ways to work with 3D objects, particularly if they are so irregular that skeleton is not suggested to be used. This is the case for Digital Elevation Models (DEM).

6.2 Digital Elevation Model Data (DEM)

Some papers analyse and study elevation models by considering triangulated meshes [54, 55]. DEM is an ordered array which represents the spatial distribution elevations over some datum in a landscape [53].

From a DEM, Fig. 20(a) presents a 3DMesh from a DEM data, of the volcano called *Iztaccíhuatl* which is located in the Mexico Valley area. A voxelization process was carried out. It consists of filling the elevation data of DEMs by voxels. We employed 523 972 voxels to obtain a solid object (see Fig. 20 (b)). On the other hand, in Fig. 21 we can appreciate both, the 5OT and 3DRC chain implementations.

Visually speaking, representing DEMs is better with either F26 or 3DRC chains than with 5OT, as can be appreciated in Fig. 21. The former give a more soften relief than the given by 5OT. Fig. 22

Object	*size*	*voxels*
Bed	33 ×113 × 104	341
Bench	44 ×29 × 122	996
Boss	19 ×38 × 40	99
Bunny	72 ×116 × 64	439
Dragon	51 ×84 × 121	566
Happy	36 ×121 × 43	547
Lion	121 ×103 × 31	365
Schaap	39 ×73 × 89	231
Trice	47 ×122 × 24	293
Wheel	108 ×42 × 106	653

Table 2. Sizes and number of voxels of the curve-skeletons

symbol \ *Object*	Bed	Bench	Boss	Bunny	Dragon	Happy	Lion	Schaap	Trice	Wheel
a	70	7	2	10	11	12	26	16	2	62
b	2	7	5	25	18	9	71	10	0	19
c	5	19	0	16	37	35	42	4	0	50
d	14	5	12	34	29	19	71	2	12	17
e	17	7	0	36	10	9	16	0	25	14
f	1	11	7	20	12	18	2	0	41	20
g	3	55	0	4	18	49	0	0	69	30
h	5	7	1	13	21	26	15	5	31	13
i	39	6	4	18	19	30	8	35	6	28
j	14	3	4	4	15	13	9	6	2	8
k	14	8	12	9	61	28	8	6	0	40
l	15	2	8	14	23	18	18	5	7	11
m	20	6	6	32	25	13	11	27	19	18
n	8	6	8	10	12	15	0	8	9	3
o	17	4	8	16	32	35	1	8	5	46
p	10	3	3	24	14	14	4	12	8	7
q	31	4	0	6	22	21	5	14	1	12
r	6	1	0	3	8	7	6	7	0	14
s	2	10	7	20	16	28	10	24	2	66
t	7	4	6	9	11	19	18	3	8	8
u	2	8	2	15	20	14	9	0	12	10
v	4	3	1	12	22	11	1	0	14	9
w	1	5	1	5	29	29	1	2	11	60
x	6	0	0	4	17	14	6	5	2	8
y	25	586	1	29	46	37	2	30	4	62
z	2	188	0	50	17	22	3	1	2	17
(	10	26	4	15	31	34	10	4	15	25
)	10	26	4	15	31	34	10	4	15	25
l_{F26}	360	1017	106	468	627	613	383	238	322	702

Table 3. Frequency of F26 symbols.

symbol \ *Object*	Bed	Bench	Boss	Bunny	Dragon	Happy	Lion	Schaap	Trice	Wheel
a	13	7	1	31	22	21	7	20	4	20
b	5	6	0	5	14	15	1	1	3	13
c	0	2	0	1	5	5	1	0	1	3
d	0	1	1	1	1	2	1	0	1	0
e	2	1	2	1	0	2	1	0	1	0
f	0	1	0	1	3	1	0	0	0	1
g	2	2	0	2	2	1	2	0	0	0
h	1	4	1	4	8	8	3	4	4	4
i	83	34	20	81	120	105	85	49	72	151
j	15	17	12	37	61	49	32	15	24	43
k	16	19	16	31	41	43	28	7	18	41
l	7	6	2	8	15	15	3	5	9	15
m	13	14	2	13	18	21	10	4	12	22
n	2	4	1	8	11	16	3	4	7	12
o	17	11	6	30	37	40	24	11	14	36
p	25	11	8	25	75	50	19	16	23	40
q	2	0	0	0	0	0	0	0	0	2
r	1	3	0	1	2	1	2	0	1	0
s	0	0	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0	0	0
u	0	0	0	0	1	0	0	0	0	0
v	1	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	0	0	0
x	1	1	0	0	1	0	0	1	0	2
y	132	819	24	156	127	148	139	91	96	245
(	10	26	4	15	30	34	10	4	15	25
)	10	26	4	15	30	34	10	4	15	25
l_{3DRC}	358	1015	104	466	624	611	381	236	320	700

Table 4. Frequency of 3DRC symbols.

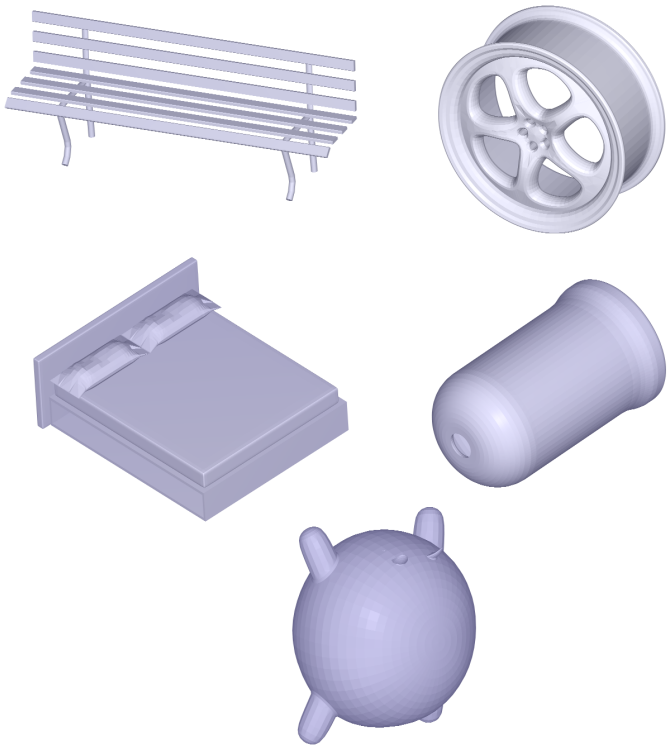
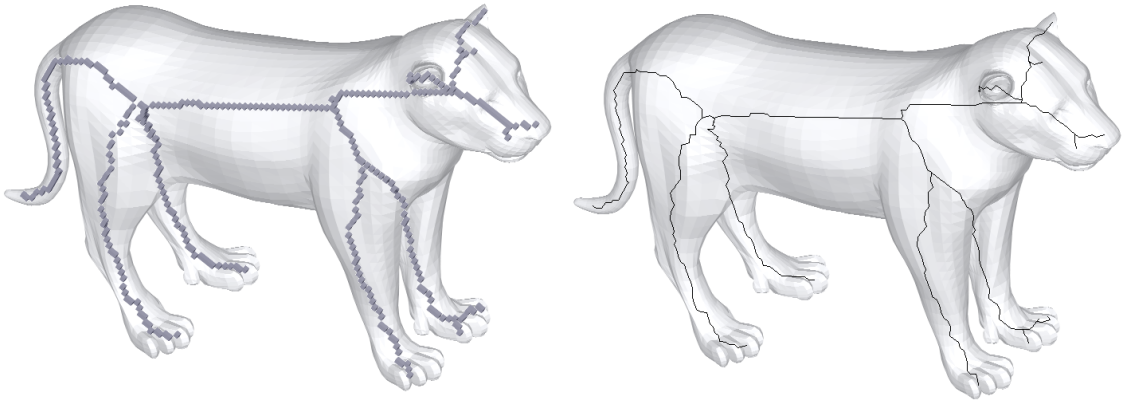


Figure 17: Example of 3D models corresponding to manufactured objets.



(a) (b)

2i8770jiiyyyyyojjiiiiiyiiyiiyyiiiiiyyyyymyipoyykoiyoiiaiyyyl(ai(gkiiiyyiohyjkjiikjoiyyjyiyyyyymiyoiyykippyyiypypi)ioyyyy
kiyykiyyyyyyyyyyyyyyyyyyyyyyyy(kmyyiyiyyyi(jyyyymiopyijimikiyyoiyyyyijkkmjykon(inoi(je)r)oji)pojjoppkjyjkyyykiyyykiyy
iimjkkkopjjnpy)omaiyiyyyyyyyyyyyoiyliyyyy(gyjii(jjipyliy)pck)pa(roiyyyjyiiibiy)yiiyiiyyyyyyy(dp)ikji)mophaiajikiooij
yjyjkykoppiyymiyyykiyokjkjiiiahjkp

(c)

Figure 18: Example of Lion curve-skeleton: (a) voxelized, (b) its tree vector skeleton, and c) its chain code.

presents two different viewpoints of the 3DRC chain. Observe in Fig. 23 that straight parts of the terrain is better represented by 3DRC than 5OT code. Whereas only one symbol, y, is needed in 3DRC, 5OT uses twice the number of symbols if a digital straight line is inclined 45 degrees.

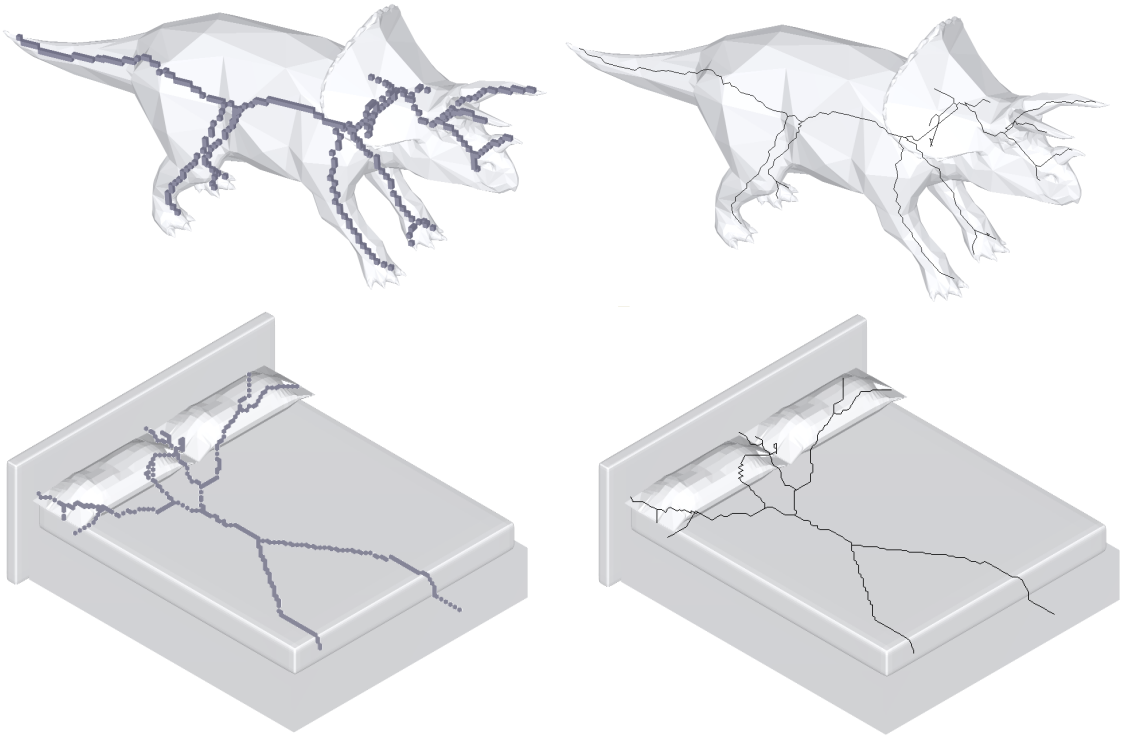


Figure 19: Curve-skeletons: (a) voxelized, (b) chain tree.

Chain code can be used to codify the shape of terrains, for example the relief of mountains can be quantified with symbol representation. Chain codes can be used in the analysis of the terrain to obtain a better quantification of its shape. They can be utilized like an object signature, i.e., when specialists read this code, for example, geologists, civil engineers, agriculturists, and so forth, depending the region of interest, a piece of "word" can tell how the shape of the region is. Fig. 24 shows the 3DRC code for the first three slices of the volcano "Iztaccihuatl". See the first contour slice from Fig. 22 (b). For example, flat regions, or those with no many changes, have a high probability of y's. See Fig. 25, where the first slice (right front of Fig. 22(b)) is coded by 3DRC. As the code suggests, the straight lines are coded by the 'y' symbol. Every change in slice is represented by the consecutive symbols oc. When an analysis is wanted to be made, a set of symbols can be inspected to know the nature of the shape. The regions with high changes of slops, are determined with symbols a,e and q.

Of course, detailed analysis of the relief should be studied when a combination of such symbols is made with others, like i's and m's.

6.3 Redundancy in information

Another interesting property of chain codes is their redundancy in information. As can be seen from previous results, the probability of each symbol to appear in chain codes is different. For example, symbol 'y' is highly probable to appear in the sample objects.

We compared the redundancy of information in F26 and 3DRC codes, and to do so, we calculate the frequency of occurrence of the symbols. For the case of curve-skeletons Table 3 shows the frequency of appearance of the symbols of F26 code, while Table 4 shows the frequency of the symbols of 3DRC. For Bench curve-skeleton note that symbol 'y' (vector $\langle 0, 0, 1 \rangle$ of F26) appears with high frequency, regarding the others; this is because this object has a lot of straight line segments. According to the theory of information [31], the entropy of a signal, in this case resulting of a string of symbols, can be obtained

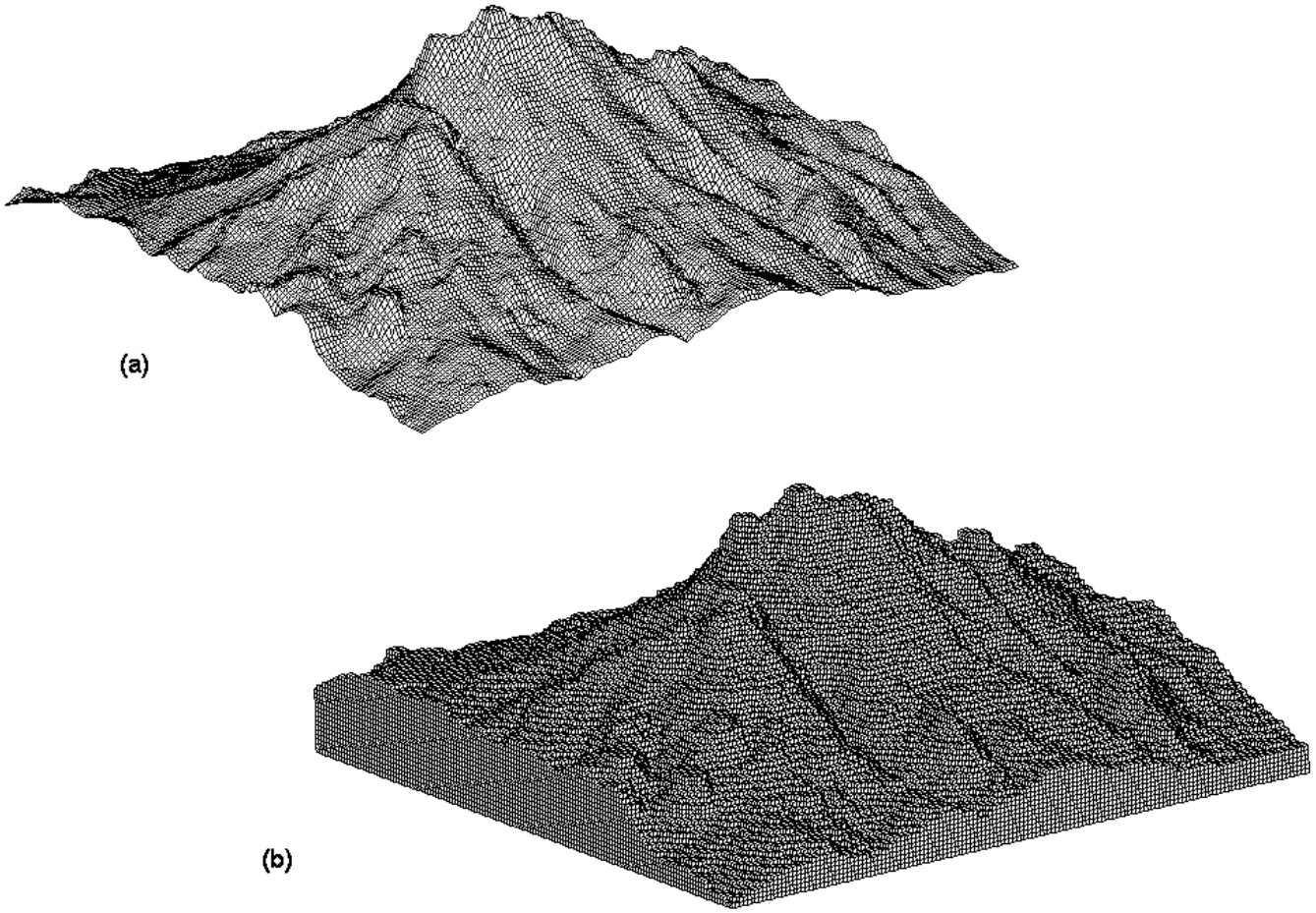


Figure 20: A 3D object: (a) “Iztaccihuatl” volcano modeled by a 3D mesh, (b) its voxelized version.

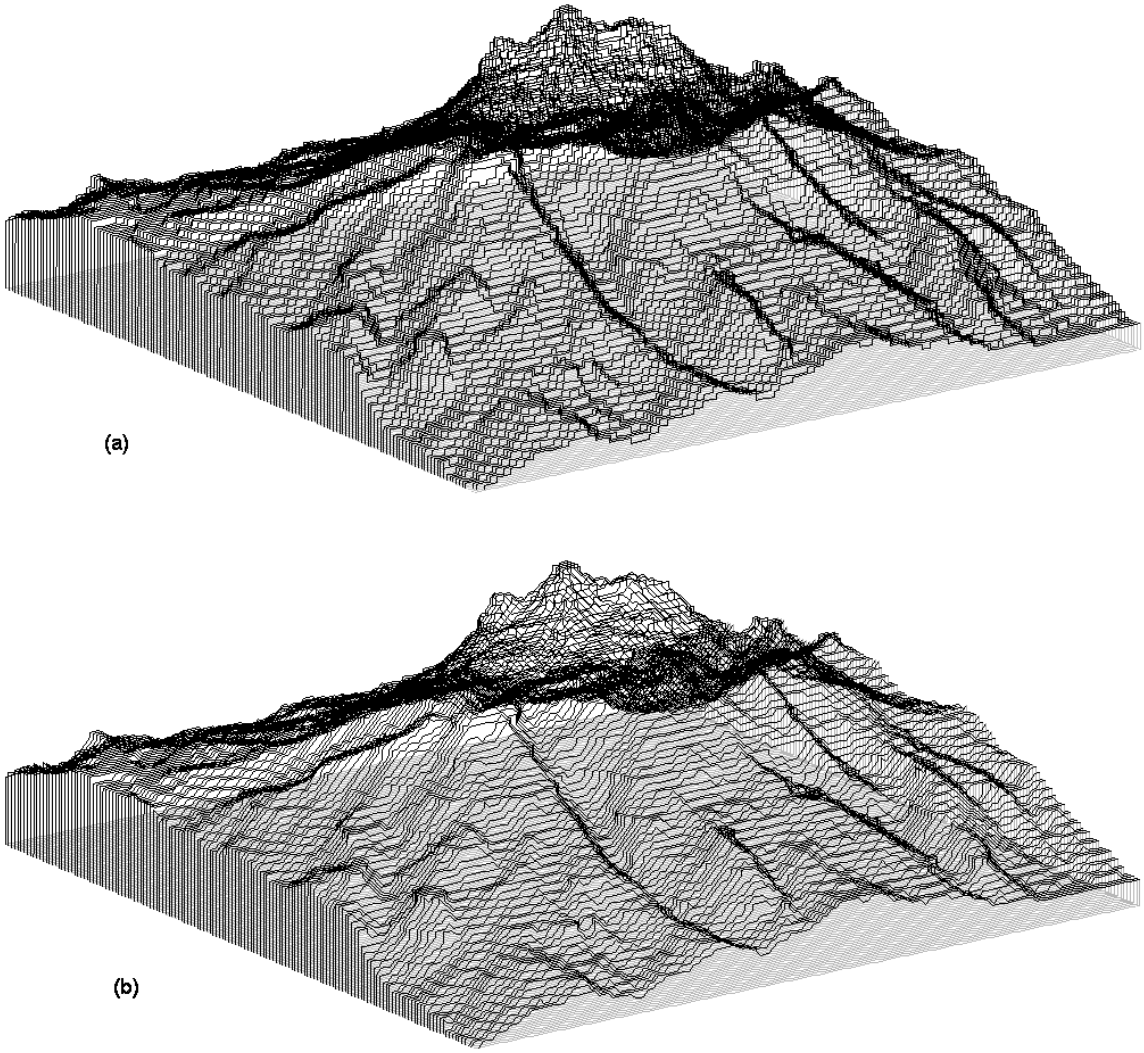


Figure 21: "Iztaccihuatl" volcano represented by chain codes: (a) 5OT chain code, (b)3DRC chain code.

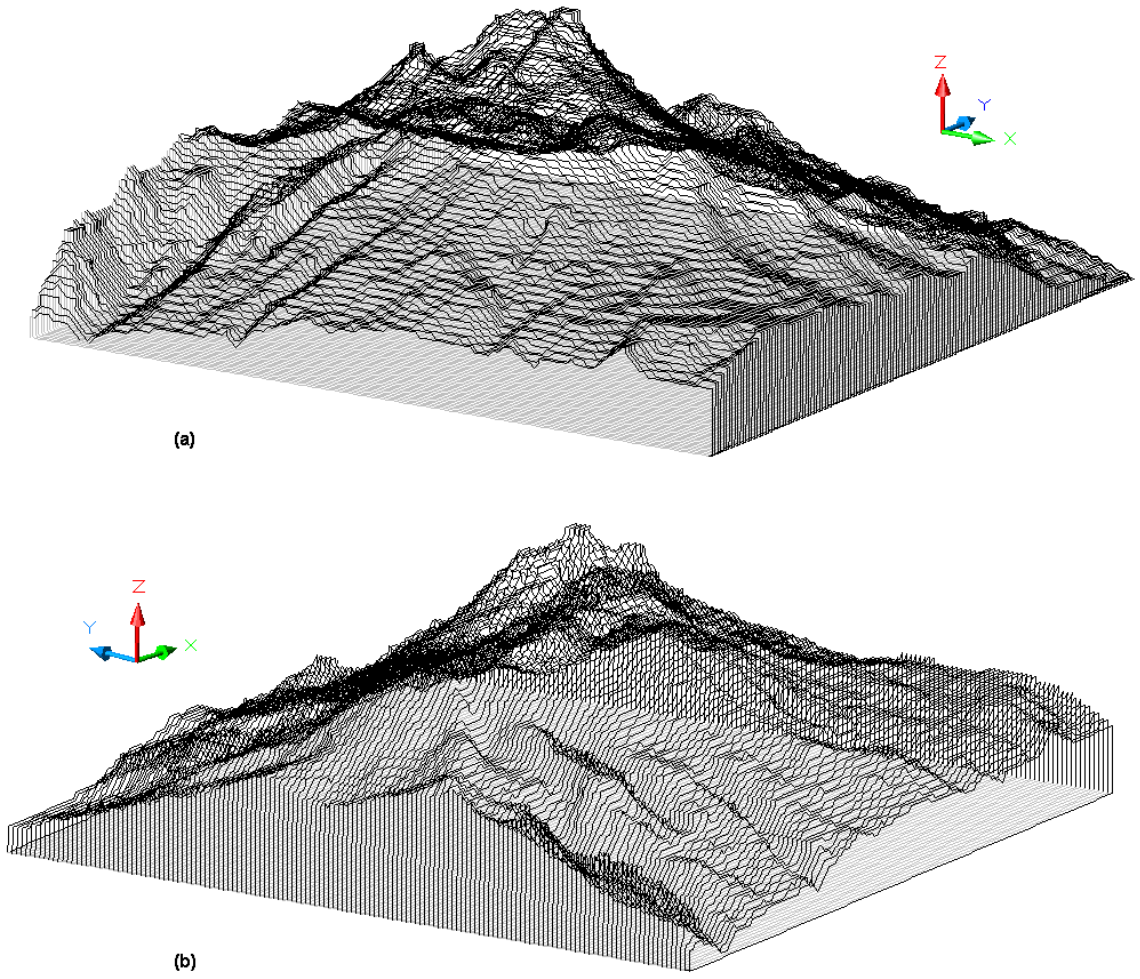


Figure 22: Two different views (a) and (b) of the "Iztaccihuatl" volcano coded by our proposed 3DRC chain.

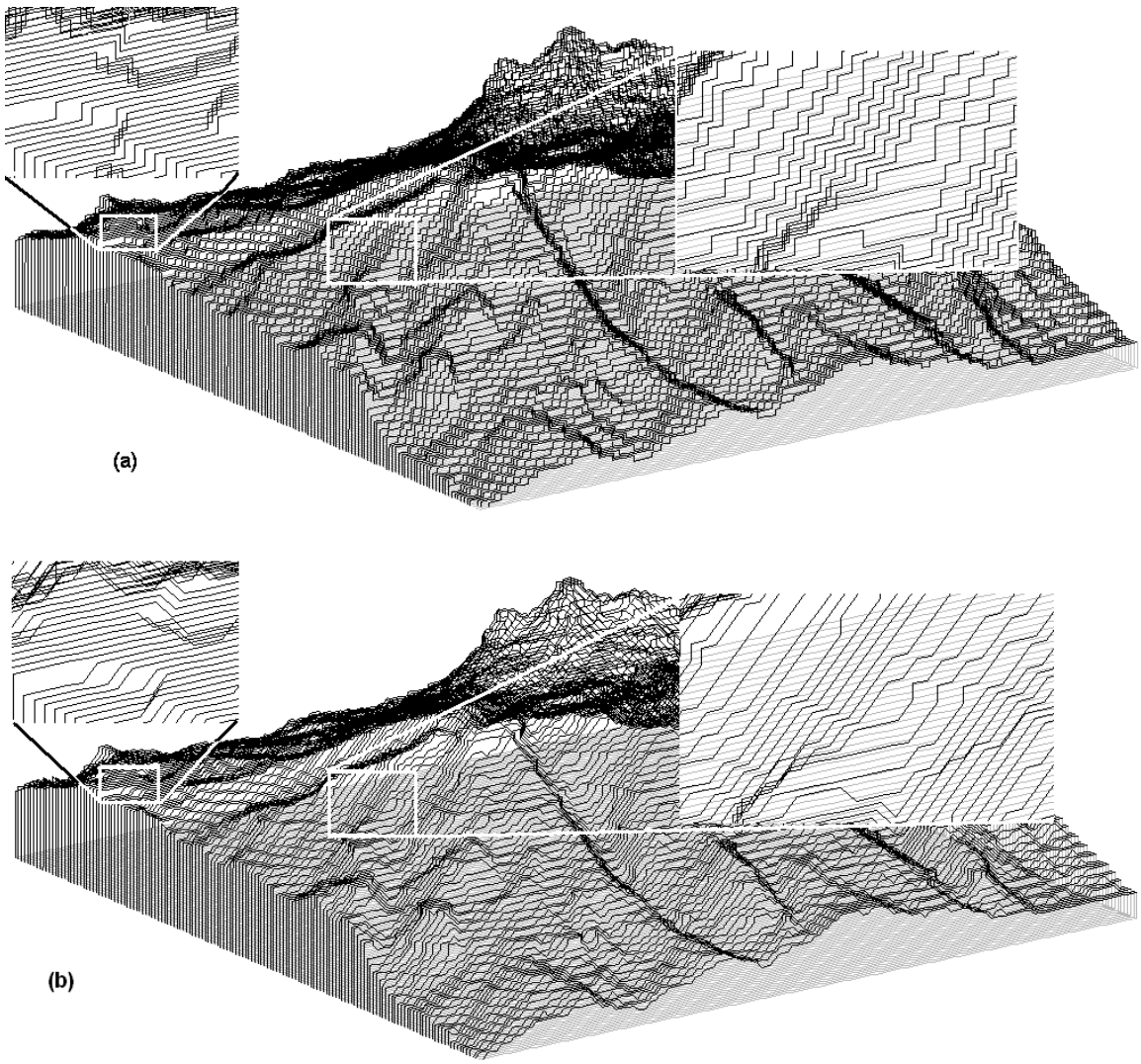


Figure 23: A DEM data: "Iztaccihuatl" volcano, (a) 5OT chain representation; (b)3DRC representation.

yyy1i6620iymiyyiyyiymyyiyyyyiiyyyiiyyyyyyyyyyiiyyiqiyeyyyiyiiiiyiiiiymiyiimiyyyiiyyyyiiyyyyymiyy
yyyyyyyyiiyymiiiymiiiiamayyyyyiaiyyymyyyyiyeiyymiiyyyyyyyyyyyyyayyyyyyyyyyyyeyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyiiyyyiiyeimmyyyiyyyiiyiiiiymiyyyyymaaiyyyiiyyyyyyyyyymiyyymaiyiiymiyiiymiiiymiyyyyyiiymiym
yyyyiyeiyymyiiyyyyyyyymiyyeyyyyyyyyyyyyyeyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyocyyyyniyayyyeimiyyyiyymyyiyyyyiiyyyyiaiymiyyyyiaaiymyamiyymyymaiiiyyiiy
miyyyymiyyyiiyyyyyyyiimiyyyyyyyyiiyiimiiiiayiyyyyymiyyyyyyyyyiiiyyiyeiymiiyyyiiyyyymiyyyeyyyy
yyyyyyyyyeyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyo...

Figure 24: The 3DRC code of the first three slices of "Iztaccihuatl" volcano .

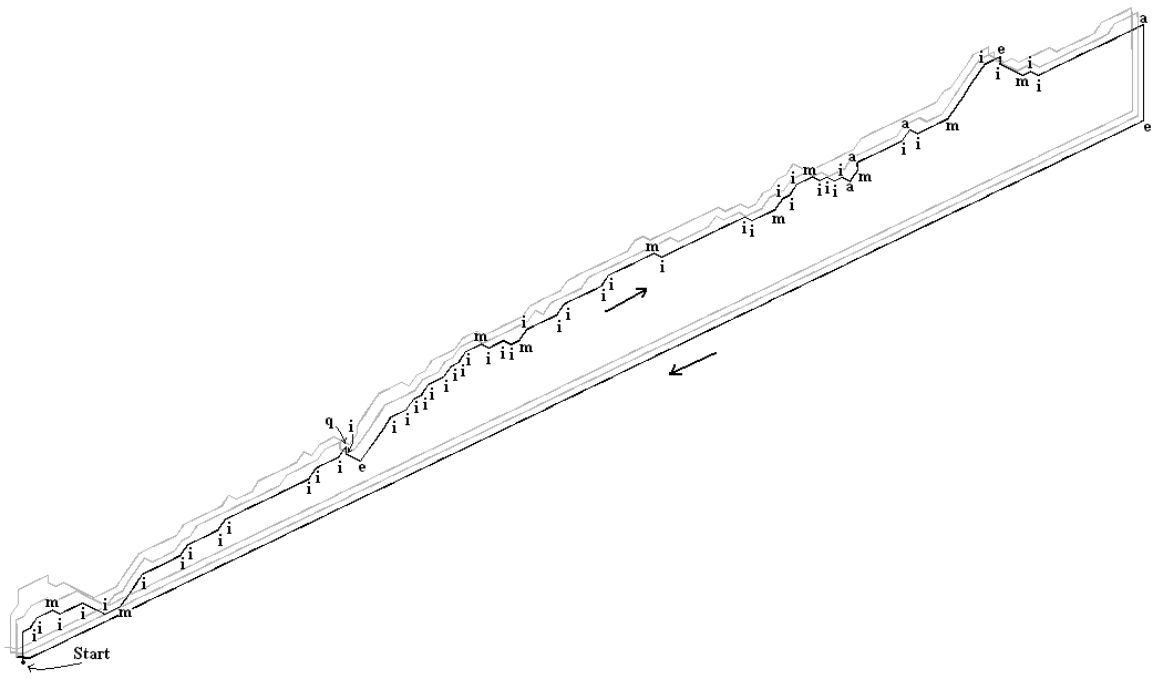


Figure 25: The first slice of "Iztaccihuatl" volcano coded by 3DRC.

from Eq.(24).

$$L = -\sum_{i=1}^{l_{code}} (p_i) log(p_i), \tag{24}$$

where L is the entropy of a chain, p_i the probability of its symbols, and l_{code} its length.

Fig. 26 shows the distribution of probabilities for each code. As can be observed, the greater part of the distribution of 3DRC frequencies is taken by five symbols: 'y', 'i', 'j', 'k', and 'p', which represent 75% of the total, while F26 frequencies are more evenly distributed by most of its symbols: 19 of them represent 75% of the total number of frequencies: 'y, 'd', 'g', 'a', 'c', 'f', 'i','m ','s ', 'k', 'e', 'b', 'h', 'l', 'q','n ','t ', 'u' and 'v'. A simple calculation shows the entropy of each code. So, given the above equation, $L_{3DRC} = 3.03$ bits/symbol, whereas $L_{F26} = 4.35$ bits/symbol.

Therefore, F26 spends more bits per symbol in memory storage than 3DRC. One of the compression algorithms, without loss of information, frequently used in the literature is the Arithmetic [32]. By applying such an algorithm to the resulted chains, to save the information of each one of the compared codes, we have obtained the amount of memory, in bits. Table 4 shows the length (number of symbols) of the chain codes, and the bits required to store the chains, after applying arithmetic algorithm. Note that the average efficiency of 3DRC, regarding F26, is 23%.

On the other hand, using information of the irregular sample given by the volcano, coding with different algorithms, also result in different memory storage efficiency. In Table 5 it is shown some important parameters due to the compared methods. F26 uses the shortest length to codify the object, almost the same like 3DRC. In fact, 3DRC needs one five more symbols in the start of the code, because of rotation operations required and one less symbol due to the first change a relative code is not required. Also, it can be seen that entropies of 5OT and 3DRC are similar, however, the number of bits required for 3DRC is 10.7% fewer than 5OT, and 36.1% fewer than F26.

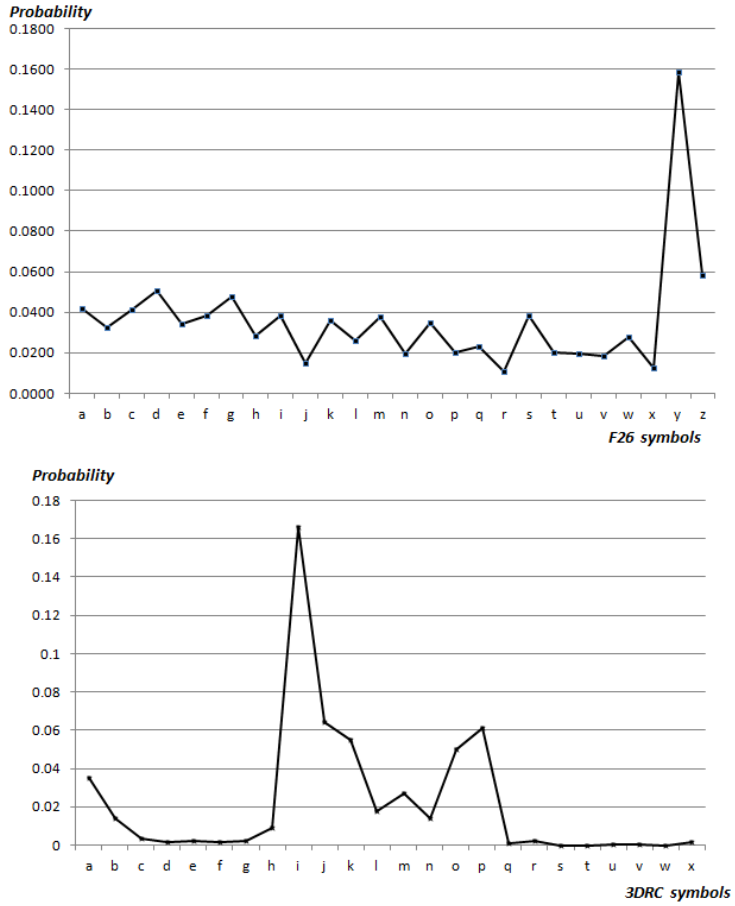


Figure 26: Symbol probabilities for the different codes, F26 and 3DRC, respectively.

6.4 Discussion

The 5OT code introduced by Bribiesca in 2000 [13] is used to represent face connected discrete curves. In fact, we have already compared 5OT with F26 [29]. In that paper, we compared 5OT with three types of curves: cube-filling Hilbert curves, lattice knots and arbitrary curves. For Hilbert and lattice knots, we found better compression of 5OT over the F26 code. The route followed by F26 was given by its six orthogonal directions. So, a subset of six symbols from the 26 was utilized. On the contrary, in the case of arbitrary curves, the 26 symbols from F26 were statistically utilized. To compare it with 5OT, a set of emerging voxels was implemented, *i.e.* a set of 0-voxels were turned on to cause a face connectivity compare with 5OT. Of course, this constitutes more processing time. Even more, if a set of voxels being traversed represent a straight line segment, repeated assignments of reference vectors have to be done to compute the change vector. On the contrary, fewer assignments are required by the 3DRC for reference and support vector controls.

However, with our method, it is not necessary to turn on the 0-voxels; with our proposed code, we naturally follow the face, edge and vertex connected curve.

This fact is very important, because when 0-voxels are turned on to obtain a face connectivity and encode using 5OT, and to decode, it is not known which voxels were turned on, thus recover same shape is difficult, unless keep the information of what 0-voxels were turned on. It takes more consuming time processing. This important issue does not arises with our method.

As can be seen from the 24 relative symbols, there is a subset given by $\{a, g, c, e\}_{group1} \cup \{y\}$ that

Object	length	F26(bits)	3DRC(bits)	Efficiency
Bed	360	1713	1310	0.2353
Bench	1017	2620	1696	0.3527
Boss	106	592	514	0.1318
Bunny	468	2342	1724	0.2639
Dragon	627	3152	2440	0.2259
Happy	613	3092	2384	0.229
Lion	383	1700	1339	0.2124
Schaap	238	1138	878	0.2285
Trice	322	1451	1238	0.1468
Wheel	702	3350	2389	0.2869
Average	471.3636	2066.0909	1571.909	0.2313

Table 5. Compressing chain codes using Arithmetic coding.

Code	*Length*	*Entropy*	*Bits*
F26	50444	2.1323	110425
5OT	58728	1.0861	78937
3DRC	50448	1.175	70474

Table 6. Comparison of 3D chain codes.

corresponds to the known code: $5\text{OT} = \{1, 2, 3, 4\} \cup \{0\}$. So, as a corollary of our method, we have that $5\text{OT} \subset 3\text{DRC}$.

Another advantage of the 3DRC code is that its shape is closer to the actual voxelized curve than that obtained from turning on voxels and represented by 5OT.

About sensitive to noise, any chain code is formulated to represent one-dimensional curves. These may code the noise depending on the original data obtained and the algorithms that were used on the original source data. In the case of the skeletons, it was mentioned that there is a vast literature for the best descriptors, including smooth curve-skeletons, with the lowest possible noise. For example, graph-based representations of point clouds tackle the problem of noise [49] or the geometry contraction process based on an iterative implicit smoothing operation can be used [56].

On the other hand, noise that may appear in the digital elevation models comes from the process of scanning the terrain elevations. However, the chain code represent most faithfully the information of the shape of the land on the basis of the database that have been utilized.

7 Conclusions and further work

We have found a new code for three-dimensional paths. This code is invariant under translation, rotation and mirror transformations. Also, we have codified skeletons, even if there are loops, we have found a method to codify them as trees. In addition, the chains that are generated to represent trees in three dimensions, contain less redundancy than F26. Since the entropy generated by an absolute code is greater, we found that 3DRC is more efficient than existing codes in 3D.

More interesting properties of the proposed code can be found, such as the performance of parentheses, which indicate the existence of nodes in the tree associated with the skeleton. The nodes here can be seen as important information about the shape of the object; therefore, they can be seen also as descriptors of the object. Future work should analyze this case in detail and consider it as a feature of objects to establish measures of similarity.

We have used curve-skeletonization as part of the process to reach chain coding. A future work should find pattern of symbols to identify noise, in order to help the curve-skeleton smoothing.

To give quantitative measures of the DEMs shapes, a further work most consider the relationships between symbols and circularities, slops and corners, than help "read" the code, like a signature of the object, that permit take decisions about the region of the terrain studied by specialists. Further work should try with matrix symbols if attending neighbor slice vicinities to analyze the regions of terrains.

8 Acknowledgement

DEM data used in applications were obtained from INEGI (Instituto Nacional de Estadística Geografía e Informática).

References

[1] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Transactions on Electronic Computers EC-10 (1961) 260 - 268.

[2] C.E. Kim, Three-dimensional digital segments, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5 (1983) 231-234.

[3] Guzman, A. (1987). Canonical Shape Description for 3D stick bodies. MCC Technical Report Number:ACA. 254-87.

[4] E. Bribiesca, A new chain code, Pattern Recognition 32 (1999) 235-251.

[5] H. Sanchez-Cruz, R.M. Rodriguez-Dagnino, Compressing bi-level images by means of a 3-bit chain code. Optical Engineering SPIE. 44 (9) 097004 (2005) 1-8.

[6] L. Echavarri-Aguinaga, R.A. Neri-Calderon, R.M. Rodriguez-Dagnino, Compression rates comparison of entropy coding for three-bit chain codes of bilevel images, Optical Engineering 46 (8) (2007). 087007.

[7] Y.K. Liu, B. Zalik, An efficient chain code with Huffman coding, Pattern Recognition 38 (4) (2005) 553-557.

[8] Y.K. Liu, W. Wei, P.J. Wanga, B. Zalik, Compressed vertex chain codes, Pattern Recognition 40 (2007) 2908-2913.

[9] S.M. Aghito, S. Forchhammer, Context-based coding of bilevel images enhanced by digital straight line analysis, IEEE Transactions on Image Processing 15 (8) (2006) 2120-2130.

[10] H. Sanchez-Cruz; M. Lopez-Cruces, H. Puga, A proposal Modification of the 3OT Chain Code, in: Proceedings of the Computer Graphics and Imaging, Innsbruck, Austria, February 13-15, 2008, 6-11.

[11] H. Sanchez-Cruz, E. Bribiesca, R.M. Rodrguez-Dagnino, Efficiency of chain codes to represent binary objects, Pattern Recognition 40 (2007) 1660-1674.

[12] H. Sanchez-Cruz, A proposal method for corner detection with an orthogonal three-direction chain code, in: Lecture Notes in Computer Science, vol. 4179, Springer, Berlin, 2006, 161-172.

[13] E. Bribiesca, A chain code for representing 3d curves, Pattern Recognition 33 (2000) 755-765.

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

[15] H. Sanchez-Cruz, Proposing a new code by considering pieces of discrete straight lines in contour shapes, J. Vis. Commun. Image. R., 21 (2010) 311-324.

[16] A. Akimov, A. Kolesnikov and P. Franti, Lossless compression of map contours by context tree modelling of chain codes, Pattern Recognition 40 (2007) 944 - 952.

[17] S. Junding and X. Heli, Contour-shape recognition and retrieval based on chain code, International Conference on Computational Intelligence and Security (2009).

[18] T. Kaneko and M. Okudaira. Encoding of arbitrary curves based on chain code representation. IEEE Trans. Commun. 33 (1985) 697-707.

[19] Y. Kato, T. Hirano and O. Nakamura, Fast template matching algorithm for contour images based on its chain coded description applied for human face identification. Pattern Recognition 40 (2007) 1646 - 1659.

[20] Z. Shi and V. Govindaraju, A chain code based scheme for fingerprint feature extraction, Pattern Recognition Letters 27 (2006) 462-468.

[21] H. Sanchez-Cruz and E. Bribiesca, Polygonal Approximation of Contour Shapes Using Corner Detectors. Journal of Applied Research and Technology. 7(3) (2009) 275-291.

[22] H. Freeman, Computer processing of line drawing images, ACM Computing Surveys 6 (1974) 57-97.

[23] A. Jonas, N. Kiryati, Digital representation schemes for 3D curves, Pattern Recognition 30 (1997) 1803-1816.

[24] A. Safonova and J. Rossignac. Compressed piecewise-circular approximations of 3D curves. Comput.-Aided Des. 35 (6) (2003) 533-547.

[25] C.S. Zhao, Epipolar parameterization for reconstructing 3D rigid curve. Pattern Recognition 30 (1997). 1817-1827.

[26] R. Klette, A. Rosenfeld, Digital Geometry. Geometric Methods for Digital Picture Analysis, Morgan Kaufmann, San Francisco, 2004, ISBN: 155860 8613, 672 pp.

[27] T.Y. Kong and A. Rosenfeld, Digital Topology: Introduction and Survey, Computer Vision, Graphics, and Image Processing, 48 (3) (1989) 357-393.

[28] A. Cayley (1889). A theorem on trees. Quart. J. Math. 23 (1889) 376378.

[29] H. Sanchez-Cruz and E. Bribiesca. Study of compression efficiency for three-dimensional discrete curves. Optical Engineering 47(7), 077206 (July 2008)

[30] J.A. Bondy and U.S. R. Murty. Graph theory with applications. Elsevier Science Publishing Co., Inc. 1982.

[31] C.E. Shannon, A mathematical theory of communication, Bell System Technical Journal, 27, pp. 379-423 and 623-656, July and October, 1948.

[32] Roberto Togneri, Christopher J. S. DeSilva. Fundamentals of information theory and coding design. Chapman Hall/CRC, 2002. 385 pp.

[33] Y.-S. Wang and T.-Y. Lee. Curve-Skeleton Extraction Using Iterative Least Squares Optimization. IEEE Trans. on Vis. and Comput. Graph. 14 (4) (2008) 926-936.

[34] T. He, L. Hong, D. Chen, and Z. Liang, Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs, IEEE Trans. Visualization and Computer Graphics, 7 (4) (Oct.-Dec. 2001) 333-342.

[35] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, Virtual Voyage: Interactive Navigation in the Human Colon, Proc. SIGGRAPH 97, (1997) 27-34

[36] N. Gagvani and D. Silver, Animating volumetric models, Graphical Models, vol. 63, no. 6, 443-458, 2001.

[37] C.-H. Lin and T.-Y. Lee, Metamorphosis of 3D polyhedral models using progressive connectivity transformations, IEEE Trans. Visualization and Computer Graphics, 11(1) (2005) 2-12.

[38] L. Wade and R.E. Parent, Automated generation of control skeletons for use in animation, The Visual Computer, 18 (2) (2002) 97-110.

[39] F.-C. Wu, W.-C. Ma, R.-H. Liang, B.-Y. Chen, and M. Ouhyoung, domain connected graph: the skeleton of a closed 3D shape for Animation, The Visual Computer, 22(2) (2006) 117-135.

[40] http://www.cs.princeton.edu/∼min/binvox/

[41] A. Brennecke and T. Isenberg, 3D Shape matching using skeleton graphs, Proc. Simulation and Visualization Conf. (SimVis 04) (2004) 299-310.

[42] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, Skeleton Based Shape Matching and Retrieval, Proc. Shape Modeling Intl (SMI 03), p. 130, 2003.

[43] T. Wang, I. Cheng, V. Lopez, E. Bribiesca, A. Basu, Valence normalized spatial median for skeletonization and matching. Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. 4 (2009) 55-62.

[44] F. S. Nooruddin and G. Turk, Simplification and repair of polygonal models using volumetric techniques, IEEE Trans. on Visual and Comp. Graph. 9(2) (2003) 191-205.

[45] K. Palgyi and A. Kuba, Directional 3D thinning using 8 subiterations, Springer-Verlag. Lecture Notes in Computer Science 1568 (1999) 325-336.

[46] N.D. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms. Visualization and Computer Graphics, IEEE Transactions on, 13(3) (2007) 530 - 548.

[47] T. Ju, M. L. Baker, W. Chiu, Computing a family of skeletons of volumetric models for shape description, Computer-Aided Design 39 (2007) 352-360.

[48] C. Lohou, G. Bertrand, A 3D 6-subiteration curve thinning algorithm based on P-simple points, Discrete Applied Mathematics 151 (2005) 198 - 228.

[49] M. Natali, S. Biasotti, G. Patané, B. Falcidieno, Graph-based representations of point clouds, Graphical Models 73 (2011) 151-164.

[50] S. Wang, J. Wu, M. Wei, X. Ma, Robust curve skeleton extraction for vascular structures, Graphical Models 74 (2012) 109-120.

[51] S. Biasotti, D. Giorgi, M. Spagnuolo and B. Falcidieno, Size functions for comparing 3D models, Pattern Recognition 41 (2008) 2855-2873.

[52] L. Serino, C. Arcelli, G. Sanniti di Baja. On the computation of the $\langle 3, 4, 5 \rangle$ curve skeleton of 3D objects. Pattern Recognition Letters 32 (2011) 1406-1414.

[53] Digital Terrain Modelling: Development and Applications in a Policy Support Environment, Lecture Notes in Geoinformation and Cartography, R.J. Peckham, G. Jordan (Eds). Springer. 2007.

[54] M. de Berg, O. Cheong, H. Haverkort, J-G Lim, L. Toma, The complexity of flow on fat terrains and its i/o-efficient computation, Computational Geometry 43 (2010) 331-356.

[55] T. de Kok, M. van Kreveld, M. Löffler, Generating realistic terrains with higher-order Delaunay triangulations, Computational Geometry 36 (2007) 52-65.

[56] O. K-C. Au, C-L Tai, H-K Chu, D. Cohen-Or, T-Y Lee, Skeleton extraction by mesh contraction. ACM SIGGRAPH 2008, article 44.