

Towards a New Approach for Modeling Volume Datasets Based on Orthogonal Polytopes in Four-Dimensional Color Space

Ricardo Pérez-Aguila

Abstract - This work is devoted to describe some results for a new approach intended to represent volume datasets in a concise way. The main idea is to specify voxelizations as orthogonal polytopes whose fourth dimension corresponds to color. Then, the 4D representation is expressed via the Extreme Vertices Model in the n -Dimensional Space (n D-EVM). There will be presented some examples that show how the proposed methodology leads to a representation whose storing requirements are lesser than those demanded by the original datasets. Moreover, it will be described the way the properties and algorithms behind the n D-EVM can be applied in order to interrogate the 4D-EVM associated to a dataset with the purpose of obtaining, in efficient way and working directly and only with the EVM, useful geometrical and topological information. Also, there will be described EVM-based algorithms for morphological erosion and dilation and the way they can be useful in aspects such as the removal of “Salt and Pepper” noise under the context of our 4D representation.

Index Terms - Representation and Manipulation of Volume Datasets, Polytopes Representation Schemes, Geometrical and Topological Interrogations, Morphological Operations.

I. INTRODUCTION

The representation of a polytope through a scheme of Hyperspatial Occupancy Enumeration is essentially a list of identical hyperspatial cells occupied by the polytope. Specific types of cells, called hypervoxels [5] are hyper-boxes (hypercubes, for example) of a fixed size that lie in a fixed grid in the n D space. By instantiation, it is well known that a 2D hypervoxel is a pixel while a 3D hypervoxel is a voxel; the term *rexel* is suggested for referencing a 4D hypervoxel [5]. The collection of hyperboxes can be codified as an n D array C_{x_1, x_2, \dots, x_n} . The array represents the coloration of each hypervoxel. If $C_{x_1, x_2, \dots, x_n} = 0$, the empty hypervoxel C_{x_1, x_2, \dots, x_n} represents an unoccupied region from the n -Dimensional space. If $C_{x_1, x_2, \dots, x_n} = k \neq 0$, where k is in a given color scale (black & white, grayscale, RGB, etc.), then the occupied hypervoxel C_{x_1, x_2, \dots, x_n} represents an used region from the n -Dimensional space with intensity k . In fact, the set of occupied cells defines an orthogonal polytope p whose vertices coincide with some of the occupied cells' vertices.

By using the representation through an array, the spatial complexity of a hypervoxelization is at least

$$\prod_{i=1}^n m_i$$

where m_i , $1 \leq i \leq n$, is the length of the grid along the X_i -axis. For example, a 3D grid with $m_1 = m_2 = m_3 = 1,000$ requires to store 1 billion (1×10^9) voxels. Moreover, according to the used color scale each voxel will have a storing requirement. For example, if the color space is RGB then each voxel will require three bytes (four, if the alpha channel is considered) for codifying its corresponding intensity.

It is well known that some devices represent natively volume datasets through voxelizations. However, sometimes their storing requirements make difficult their manipulation and the extraction of information and knowledge. In this sense, several efforts have been made in order to reduce the spatial complexity of volume datasets always taking in account the information they contain, due to its importance and relevance, should be compromised as minimum as possible. For example, in [6] is presented an algorithm for compression of datasets by means of quadrees in order to encoding slices of data. Such encodings are used for discovering similarities between consecutive slices. In [15], 3D medical datasets are compressed via a method sustained in the use of octrees. Both works share us evidence of the spatial conciseness provided by considering the use of solid representation schemes. This work is devoted to present an alternative representation for volume datasets. In particular, datasets whose color scales are not binary are the dominion of the proposal (in [9] is presented a methodology designed specifically for 3D datasets with black & white color scale). The main idea is to specify datasets as 4D polytopes where the fourth dimension corresponds to color. Then, the 4D representation is concisely expressed and manipulated through a polytopes' representation scheme: the Extreme Vertices Model.

This work is organized as follows: The Section II will describe the fundamentals behind the Extreme Vertices Model in the n -Dimensional Space (n D-EVM). Section III discusses some algorithms for interrogate an n D-EVM in order to obtain some useful geometrical information about an orthogonal polytope represented via the model. Section IV describes Rodríguez & Ayala's algorithms for performing morphological erosion and dilation over an n D-OPP expressed in the EVM. The proposed methodology for the conversion of a volume dataset to a 4D-EVM, the core contribution of this work, is described in the Section V. The Section VI presents some examples of datasets expressed under the EVM. By comparing the storing

Manuscript received March 19, 2010.

Ricardo Pérez-Aguila is with the Universidad Tecnológica de la Mixteca (UTM), Carretera Huajuapán-Acatlilma Km. 2.5, Huajuapán de León, Oaxaca 69000, México (e-mail: ricardo.perez.aguila@gmail.com).

requirements between the original and the 4D-EVM representations it is established the conciseness of our proposal. The Section VII describes the way the algorithms and properties behind the nD -EVM can be applied in order to interrogate the 4D-EVM associated to a dataset for obtaining useful information. There are described the benefits of the internal organization of the Extreme Vertices in an EVM. Specifically, it will be seen how such organization, together with our proposal, leads to a characterization of the elements in a dataset according to its color intensity. Finally, Section VIII shows the application of erosion and dilation's EVM-based algorithms on the removal of "Salt and Pepper" noise in the context of a 4D-OPP that models a 3D volume dataset and their impact in the spatial complexity of our proposed representation.

II. THE EXTREME VERTICES MODEL IN THE n -DIMENSIONAL SPACE (nD -EVM)

This section and Section III compose a summary of results originally presented in [1, 8]. For the sake of brevity, some propositions are only enunciated. Their corresponding proofs can be found in [1, 8]. In Section II.A there will be introduced some conventions and preliminary background related directly with orthogonal polytopes. In Sections II.B to II.F the foundations of the nD -EVM will be established. The Section II.G presents some basic algorithms under the EVM.

A. The n -Dimensional Orthogonal Pseudo-Polytopes (nD -OPPs)

Definition 1 [16]: A Singular n -Dimensional Hyper-Box in \mathbb{R}^n is given by the continuous function

$$I^n : [0,1]^n \rightarrow [0,1]^n \\ x \sim I^n(x) = x$$

A General Singular k -Dimensional Hyper-Box in the closed set $A \subset \mathbb{R}^n$ is the continuous function $c : [0,1]^k \rightarrow A$

Definition 2 [16]: For all i , $1 \leq i \leq n$, the two singular $(n-1)D$ hyper-boxes $I_{(i,0)}^n$ and $I_{(i,1)}^n$ are defined as follows:

If $x \in [0,1]^{n-1}$ then

$$I_{(i,0)}^n(x) = I^n(x_1, \dots, x_{i-1}, 0, x_i, \dots, x_{n-1}) = (x_1, \dots, x_{i-1}, 0, x_i, \dots, x_{n-1}) \\ I_{(i,1)}^n(x) = I^n(x_1, \dots, x_{i-1}, 1, x_i, \dots, x_{n-1}) = (x_1, \dots, x_{i-1}, 1, x_i, \dots, x_{n-1})$$

Definition 3 [16]: In a general singular nD hyper-box c the (i, α) -cell is defined as $c_{(i,\alpha)} = c \circ I_{(i,\alpha)}^n$, while the orientation of an $(n-1)D$ cell $c \circ I_{(i,\alpha)}^n$ is given by $(-1)^{\alpha+i}$. An $(n-1)D$ oriented cell is given by the scalar-function product $(-1)^{i+\alpha} \cdot c \circ I_{(i,\alpha)}^n$

Definition 4 [16]: A formal linear combination of singular general kD hyper-boxes, $1 \leq k \leq n$, for a closed set A is called a k -chain.

Definition 5 [16]: Given a singular nD hyper-box I^n , the $(n-1)$ -chain, called the boundary of I^n , is defined as

$$\partial(I^n) = \sum_{i=1}^n \left(\sum_{\alpha=0,1} (-1)^{i+\alpha} \cdot I_{(i,\alpha)}^n \right)$$

Moreover, given a singular general nD hyper-box c we define the $(n-1)$ -chain, called the boundary of c , by

$$\partial(c) = \sum_{i=1}^n \left(\sum_{\alpha=0,1} (-1)^{i+\alpha} \cdot c \circ I_{(i,\alpha)}^n \right)$$

Definition 6 [16]: The boundary of an n -chain $\sum c_i$, where each c_i is a singular general nD hyper-box, is given by

$$\partial\left(\sum c_i\right) = \sum \partial(c_i)$$

Definition 7: A collection c_1, c_2, \dots, c_k , $1 \leq k \leq 2^n$, of general singular nD hyper-boxes is a combination of nD hyper-boxes if and only if

$$\left[\bigcap_{\alpha=1}^k c_\alpha([0,1]^n) = \underbrace{(0, \dots, 0)}_n \right] \wedge \\ \left[(\forall i, j, i \neq j, 1 \leq i, j \leq k) (c_i([0,1]^n) \neq c_j([0,1]^n)) \right]$$

In the above definition the first part of the conjunction establishes that the intersection between all the nD general singular hyper-boxes is the origin, while the second part establishes that there are not overlapping nD hyper-boxes.

Definition 8: An n -Dimensional Orthogonal Pseudo-Polytope p , or just an nD -OPP p , will be an n -chain composed by nD hyper-boxes arranged in such way that by selecting a vertex, in any of these hyper-boxes, it describes a combination of nD hyper-boxes (Definition 7) composed up to 2^n hyper-boxes.

B. nD -EVM's Fundamentals

Definition 9: Let c be a combination of hyper-boxes in the n -Dimensional space. An Odd Adjacency Edge of c , or just an Odd Edge, will be an edge with an odd number of incident hyper-boxes of c . Conversely, if an edge has an even number of incident hyper-boxes of c , it will be called Even Adjacency Edge, or just an Even Edge.

Definition 10: A brink or extended edge is the maximal uninterrupted segment, built out of a sequence of collinear and contiguous odd edges of an nD -OPP.

Property 1: Even edges of an nD -OPP do not belong to brinks.

Property 2: Every odd edge belongs to brinks, whereas every brink consists of m edges, $m \geq 1$, and contains $m+1$ vertices. Two of these vertices are at either extreme of the brink and the remaining $m-1$ are interior vertices.

Property 3: Any extreme vertex of an nD -OPP, $n \geq 1$, when is locally described by a set of surrounding nD hyper-boxes, has exactly n incident linearly independent odd edges.

Definition 11: The ending vertices of all the brinks in p will be called Extreme Vertices of an nD -OPP p . $EV(p)$ will denote to the set of Extreme Vertices of p .

The Fig. 1 shows an example of a 3D-OPP and its set of Extreme Vertices. Vertices v_1 , v_2 , and v_3 are non-extreme vertices because, in the case of v_3 , it has six incident odd edges, while vertices v_1 and v_2 have each one four incident coplanar odd edges (see Property 3). In the figure can be also appreciated that exactly three linearly independent odd edges are incident to the remaining vertices, actually, the Extreme Vertices of p .

The brinks in an nD -OPP p can be classified according to the main axis to which they are parallel. Since the extreme vertices mark the end of brinks in the n orthogonal directions, is that any of the n possible sets of brinks parallel to X_i -axis, $1 \leq i \leq n$, produces to the same set $EV(p)$. As an example, the Figs. 2.a, 2.b, and 2.c, show brinks parallel to X_1 , X_2 , and X_3 -axes, respectively, for a 3D-OPP.

Definition 12: Let p be an nD -OPP. The Extreme Vertices Model of p , denoted by $EVM_n(p)$, is defined as the model as only stores to all the extreme vertices of p .

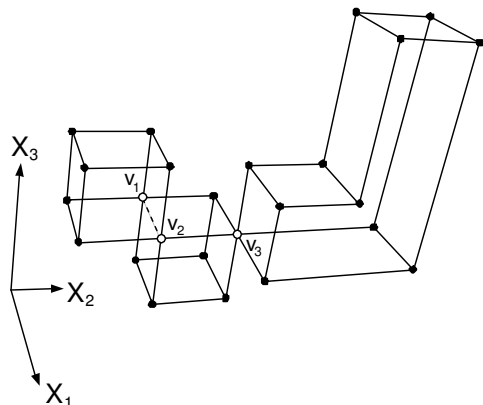


Fig. 1. Example of a 3D-OPP and its set of Extreme Vertices. Vertices v_1 , v_2 , and v_3 are non-extreme vertices. (Continuous lines indicate odd edges while the dotted line indicates an even edge).

Theorem 1: Let p be an nD -OPP. Then $\text{Card}(\text{EV}(p)) = \text{Card}(\text{EVM}_n(p))$ is an even number.

Let Q be a finite set of points in \mathbb{R}^3 . In [1] was defined the ABC-sorted set of Q as the set resulting from sorting Q according to coordinate A, then to coordinate B, and then to coordinate C. For instance, a set Q can be ABC-sorted in six different ways: $X_1X_2X_3$, $X_1X_3X_2$, $X_2X_1X_3$, $X_2X_3X_1$, $X_3X_1X_2$ and $X_3X_2X_1$. Now, let p be a 3D-OPP. According to [1] the Extreme Vertices Model of p , $\text{EVM}_3(p)$, denotes to the ABC-sorted set of the extreme vertices of p . Then $\text{EVM}_3(p) = \text{EV}(p)$ except by the fact that coordinates of points in $\text{EV}(p)$ are not necessarily sorted. In general, it is always assumed that coordinates of extreme vertices in the Extreme Vertices Model of an nD -OPP p , $\text{EVM}_n(p)$, have a fixed coordinates ordering (for example, $X_1X_2 \dots X_{i-1} \dots X_{n-1}X_n$ such that $i-1 < i$, $1 < i \leq n$). Moreover, when an operation requires manipulating two EVMs, it is assumed both sets have the same coordinates ordering.

Definition 13: The Projection Operator for $(n-1)D$ cells, points, and set of points is respectively defined as follows:

- Let $c(I_{(i,\alpha)}^n(x)) = (x_1, \dots, x_n)$ be an $(n-1)D$ cell embedded in the nD space. $\pi_j(c(I_{(i,\alpha)}^n(x)))$ will denote the projection of the cell $c(I_{(i,\alpha)}^n(x))$ onto an $(n-1)D$ space embedded in nD space whose supporting hyperplane is perpendicular to X_j -axis, i.e. $\pi_j(c(I_{(i,\alpha)}^n(x))) = (x_1, \dots, \hat{x}_j, \dots, x_n)$
- Let $v = (x_1, \dots, x_n)$ a point in \mathbb{R}^n . The projection of that point in the $(n-1)D$ space, denoted by $\pi_j(v)$, is given by $\pi_j(v) = (x_1, \dots, \hat{x}_j, \dots, x_n)$
- Let Q be a set of points in \mathbb{R}^n . The projection of the points in Q , denoted by $\pi_j(Q)$, is a set of points in \mathbb{R}^{n-1} such that $\pi_j(Q) = \{p \in \mathbb{R}^{n-1} : p = \pi_j(x), x \in Q \subset \mathbb{R}^n\}$

In all cases \hat{x}_j is the coordinate corresponding to X_j -axis to be suppressed.

Definition 14: Let p be an nD -OPP. A kD extended hypervolume of p , $1 < k < n$, denoted by $\phi(p)$, is the maximal set of kD cells of p that lies in a kD space, such that a kD cell e_0 belongs to a kD extended hypervolume if and only if e_0 belongs to an $(n-1)D$ cell present in $\partial(p)$, i.e.

$$(e_0 \in \phi(p)) \Leftrightarrow$$

$$(\exists c, c \text{ belongs to } \partial(p)) (e_0([0,1]^k) \subseteq c([0,1]^{n-1}))$$

Definition 15: Consider an nD -OPP p :

- Let np_i be the number of distinct coordinates present in the vertices of p along X_i -axis, $1 \leq i \leq n$.
- Let $\Phi_k^i(p)$ be the k -th $(n-1)D$ extended hypervolume, or just a $(n-1)D$ couplet, of p which is perpendicular to X_i -axis, $1 \leq k \leq np_i$.

See in Figs. 2.d, 2.e, and 2.f, a 3D-OPP with its corresponding sets of 2D couplets perpendicular to X_1 , X_2 , and X_3 axes, respectively.

C. Sections and Slices of nD -OPPs

Definition 16: A Slice is the region contained in an nD -OPP p between two consecutive couplets of p . $\text{Slice}_k^i(p)$ will denote to the k -th slice of p which is bounded by $\Phi_k^i(p)$ and $\Phi_{k+1}^i(p)$, $1 \leq k < np_i$.

Definition 17: A Section is the $(n-1)D$ -OPP, $n > 1$, resulting from the intersection between an nD -OPP p and a $(n-1)D$ hyperplane perpendicular to the coordinate axis X_i , $n \geq i \geq 1$, which not coincide with any $(n-1)D$ -couplet of p . A section will be called external or internal section of p if it is empty or not, respectively. $S_k^i(p)$ will refer to the k -th section of p between $\Phi_k^i(p)$ and $\Phi_{k+1}^i(p)$, $1 \leq k < np_i$. Moreover, $S_0^i(p)$ and $S_{np_i}^i(p)$ will refer to the empty sections of p before $\Phi_1^i(p)$ and after of $\Phi_{np_i}^i(p)$, respectively. Finally, $ns_i = np_i + 1$ refers to the number of sections of the nD -OPP p .

See in Figs. 2.g, 2.h, and 2.i, a 3D-OPP with its corresponding sets of internal sections perpendicular to X_1 , X_2 , and X_3 axes, respectively.

D. Computing Couplets from Sections

Theorem 2: The projection of the set of $(n-1)D$ couplets, $\pi_i(\Phi_k^i(p))$, $1 \leq i \leq n$, of an nD -OPP p , can be obtained by computing the regularized XOR (\otimes^*) between the projections of its previous $\pi_i(S_{k-1}^i(p))$ and next $\pi_i(S_k^i(p))$ sections, i.e.,

$$\pi_i(\Phi_k^i(p)) = \pi_i(S_{k-1}^i(p)) \otimes^* \pi_i(S_k^i(p)), \quad \forall k \in [1, np_i]$$

E. Computing Sections from Couplets

Theorem 3: The projection of any section, $\pi_i(S_k^i(p))$, of an nD -OPP p , can be obtained by computing the regularized XOR between the projection of its previous section, $\pi_i(S_{k-1}^i(p))$, and the projection of its previous couplet $\pi_i(\Phi_k^i(p))$. Or, equivalently, by computing the regularized XOR of the projections of all the previous couplets, i.e.

$$\begin{cases} S_0^i(p) = \emptyset \\ \pi_i(S_k^i(p)) = \pi_i(S_{k-1}^i(p)) \otimes^* \pi_i(\Phi_k^i(p)), \quad \forall k \in [1, np_i] \end{cases}$$

that is $\pi_i(S_k^i(p)) = \bigotimes_{j=1}^k \pi_i(\Phi_j^i(p))$

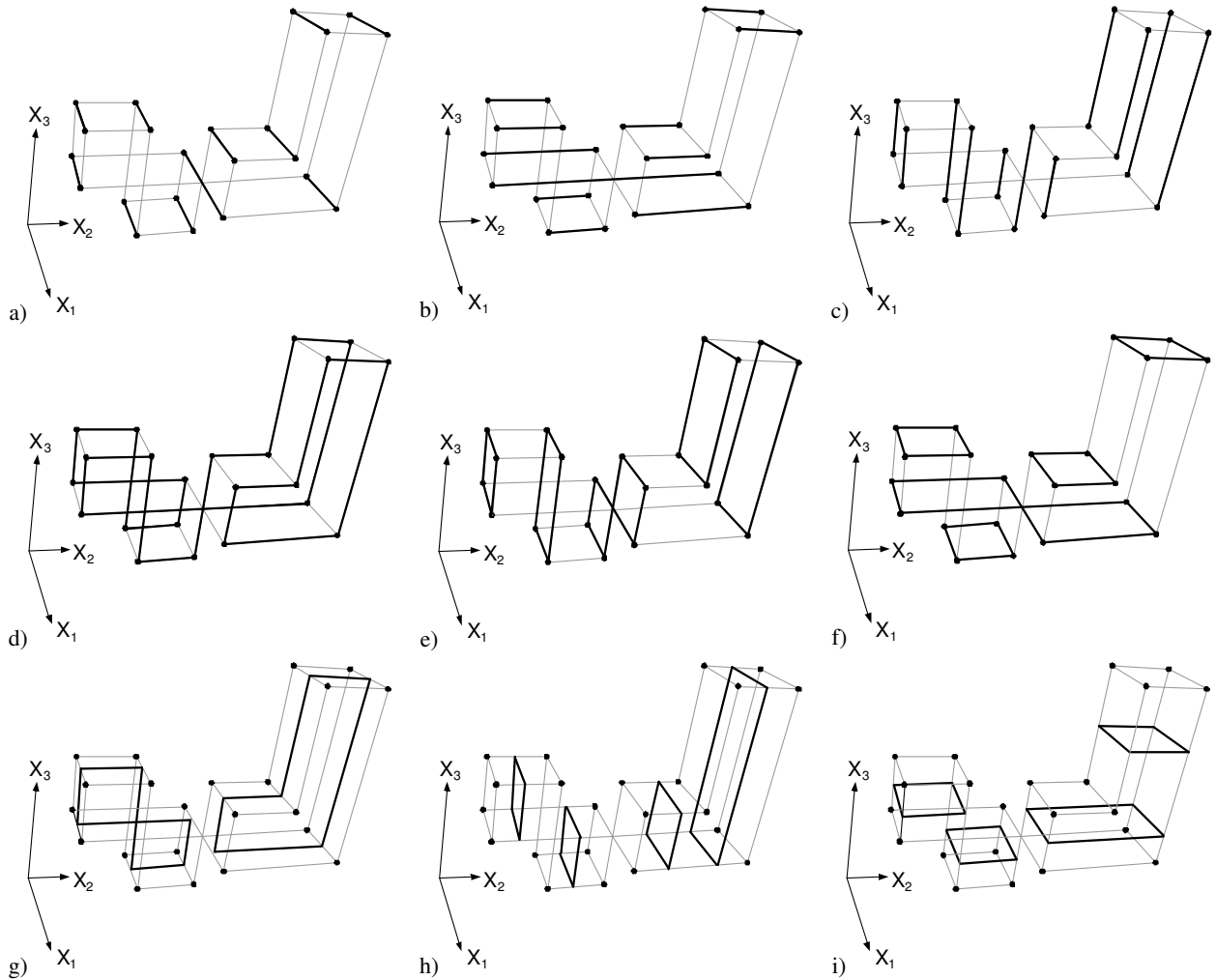


Fig. 2. Brinks (a, b, c), 2D couplets (d, e, f), and 2D sections (g, h, i) in a 3D-OPP (the OPP presented in Fig. 1).

F. The Regularized XOR operation on the nD-EVM

Theorem 4: Let p and q be two nD-OPPs having $EVM_n(p)$ and $EVM_n(q)$ as their respective Extreme Vertices Models in nD space, then

$$EVM_n(p \otimes^* q) = EVM_n(p) \otimes EVM_n(q)$$

This result allows expressing a formula for computing nD-OPPs sections from couplets and vice-versa, by means of their corresponding Extreme Vertices Models. These formulae are obtained by combining Theorem 4 with Theorem 2; and Theorem 4 with Theorem 3, respectively:

Corollary 1:

$$EVM_{n-1}(\pi_i(\Phi_k^i(p))) = EVM_{n-1}(\pi_i(S_{k-1}^i(p))) \otimes EVM_{n-1}(\pi_i(S_k^i(p)))$$

Corollary 2:

$$EVM_{n-1}(\pi_i(S_k^i(p))) = EVM_{n-1}(\pi_i(S_{k-1}^i(p))) \otimes EVM_{n-1}(\pi_i(\Phi_k^i(p)))$$

G. Basic Algorithms for the nD-EVM

It was stated in Section II.B that coordinates of vertices in the Extreme Vertices Model of an nD-OPP p , $EVM_n(p)$, have a fixed coordinates ordering. It was also commented when an operation requires manipulating two EVMs, it is

assumed both sets have the same coordinates ordering. Now, we introduce X_A -axis and X_Z -axis as the nD space's coordinate axes respectively associated to the first and last coordinates present in the vertices of $EVM_n(p)$. For example, given coordinates ordering $X_1X_2X_3X_4$, for a 4D-OPP, then $X_A = X_1$ and $X_Z = X_4$.

The following primitive operations are in fact based in those originally presented in [1]:

Output: An empty nD-EVM.
Procedure InitEVM()
 { Returns the empty set. }

Input: An (n-1)D-EVM hvl embedded in nD space.
Input/Output: An nD-EVM p
Procedure PutHvl(EVM hvl, EVM p)
 { Appends an (n-1)D couplet hvl, which is perpendicular to X_A -axis, to p. }

Input: An nD-EVM p
Output: An (n-1)D-EVM embedded in (n-1)D space.
Procedure ReadHvl(EVM p)
 { Extracts the next available (n-1)D couplet perpendicular to X_A -axis of p. }

Input: An nD -EVM p
Output: A Boolean.
Procedure EndEVM(EVM p)
{ Returns true if the end of p along X_A -axis has been reached. }

Input/Output: An $(n-1)D$ -EVM p embedded in $(n-1)D$ space.
Input: A coordinate $coord$ of type $CoordType$ ($CoordType$ is the chosen type for the vertex coordinates)
Procedure SetCoord(EVM p , $CoordType$ $coord$)
{ Sets the X_A -coordinate to $coord$ on every vertex of the $(n-1)D$ couplet p . For $coord = 0$, it performs the projection $\pi_A(p)$. }

Input: An $(n-1)D$ -EVM p embedded in nD space.
Output: A $CoordType$
Procedure GetCoord(EVM p)
{ Gets the common X_A coordinate of the $(n-1)D$ couplet p . }

Input: An nD -EVM p .
Output: A $CoordType$
Procedure GetCoordNextHvl(EVM p)
{ Gets the common X_A coordinate of the next available $(n-1)D$ couplet of p . }

Input: Two nD -EVMs p and q .
Output: An nD -EVM
Procedure MergeXor(EVM p , EVM q)
{ Applies the Exclusive OR operation to the vertices of p and q and returns the resulting set. }

Function MergeXor performs an XOR between two nD -EVMs, that is, it keeps all vertices belonging to either $EVM_n(p)$ or $EVM_n(q)$ and discards any vertex that belongs to both $EVM_n(p)$ and $EVM_n(q)$. Since the model is sorted, this function consists on a simple merging-like algorithm, and therefore, it runs on linear time [1]. Its complexity is given by $O(\text{Card}(EVM_n(p)) + \text{Card}(EVM_n(q)))$ since each vertex from $EVM_n(p)$ and $EVM_n(q)$ needs to be processed just once. Moreover, according to Theorem 4, the resulting set corresponds to the regularized XOR operation between p and q since

$$EVM_n(p \otimes^* q) = EVM_n(p) \otimes EVM_n(q)$$

From the above primitive operations, the Algorithms 1 and 2 may be easily derived [1, 8].

Algorithm 1. Computing $EVM_{n-1}(\pi_1(S_k^A(p)))$ as

$$EVM_{n-1}(\pi_1(S_{k-1}^A(p))) \otimes EVM_{n-1}(\pi_1(S_k^A(p)))$$

Input: A section S encoded as a $(n-1)D$ -EVM.
A couplet hvl encoded as a $(n-1)D$ -EVM.
Output: An $(n-1)D$ -EVM.
Procedure GetSection(EVM S , EVM hvl)
// Returns the projection of the
// next section of an nD -OPP whose
// previous section is S .
return MergeXor(S , hvl)
end-of-procedure

Algorithm 2. Computing $EVM_{n-1}(\pi_1(\Phi_k^A(p)))$ as

$$EVM_{n-1}(\pi_1(S_{k-1}^A(p))) \otimes EVM_{n-1}(\pi_1(S_k^A(p)))$$

Input: $(n-1)D$ -EVM associated to section S_i .
 $(n-1)D$ -EVM associated to section S_j .
Output: An $(n-1)D$ -EVM.
Procedure GetHvl(EVM S_i , EVM S_j)
// Returns the projection of the
// couplet between consecutive
// sections S_i and S_j .
return MergeXor(S_i , S_j)
end-of-procedure

The Algorithm 3 computes the sequence of sections of an nD -OPP p from its nD -EVM using the previous functions [1, 8]. It sequentially reads the projections of the $(n-1)D$ couplets hvl of the polytope p . Then it computes the sequence of sections using function *GetSection*. Each pair of sections S_i and S_j (the previous and next sections about the current hvl) is processed by a generic processing procedure (called *Process*), which performs the desired actions upon S_i and S_j (Note that some processes may only need one of such sections).

Algorithm 3 Computing the sequence of sections from an nD -OPP p represented through the nD -EVM.

Input: An nD -EVM p .
Procedure EVM_to_SequenceSequence(EVM p)
EVM hvl // Current couplet.
EVM S_i , S_j // Previous and next
// sections about hvl .
 $hvl = \text{InitEVM}()$
 $S_i = \text{InitEVM}()$
 $S_j = \text{InitEVM}()$
 $hvl = \text{ReadHvl}(p)$
while (Not (EndEVM(p)))
 $S_j = \text{GetSection}(S_i, hvl)$
Process(S_i , S_j)
 $S_i = S_j$
 $hvl = \text{ReadHvl}(p)$ // Read next couplet.
end-of-while
end-of-procedure

III. INTERROGATIONS ON nD -OPPS REPRESENTED VIA THE nD -EVM

The following two sections present a pair of algorithms that show the applicability of the nD -EVM. In particular, the algorithms have the objective of determining two geometrical properties of nD -OPPs expressed in the EVM: computation of the content enclosed by an nD -OPP (Section III.A), and computation of the $(n-1)D$ content of a nD -OPP's boundary (Section III.B).

A. Computing the Content of an nD -OPP

The 1D content of a segment is its perimeter; the 2D content of a polygon is its area; the 3D content of a polyhedron is its volume, and so on. In this section, it is described a procedure for computing the nD content enclosed by an nD -OPP. In this case we will consider the partition induced by its Slices (Definition 16). A Slice can be seen as a set of one or more disjoint nD hyperprisms whose $(n-1)D$ base is the slice's section. As pointed out in [1] the volume of a 3D-OPP p can be computed as the sum of the volumes of its 3D slices, where the volume of a $Slice_k^i(p)$ is given by the product between the area of its respective section $S_k^i(p)$ (the 2D base of $Slice_k^i(p)$) and the

distance between $\Phi_k^i(p)$ and $\Phi_{k+1}^i(p)$ (the height of the 3D prism $Slice_k^i(p)$). Now let $q = S_k^i(p)$. The area of the 2D-OPP q (see Fig. 3 for an example) can be computed as the sum of the areas of its 2D slices, where the area of a $Slice_k^i(q)$ is given by the product between the length of its respective section $S_k^i(q)$ (the 1D base of $Slice_k^i(q)$) and the distance between $\Phi_k^i(q)$ and $\Phi_{k+1}^i(q)$ (the height of the “2D prism” $Slice_k^i(p)$). Finally let $r = S_k^i(q)$. In the base case the length of the 1D-OPP r is computed as the sum of the lengths of its brinks.

Let p be an n D-OPP. The n D space enclosed by p , denoted by $Content_{(n)}(p)$, can be computed as the sum of the contents of its n D slices [8]:

$Content_{(n)}(p)$

$$= \begin{cases} Length(p) & n = 1 \\ \sum_{k=1}^{np_i-1} Content_{(n-1)}(S_k^i(p)) \cdot dist(\Phi_k^i(p), \Phi_{k+1}^i(p)) & n > 1 \end{cases}$$

Where np_i is the number of couplets of p perpendicular to X_i -axis; $S_k^i(p)$ is the k -th section of the n D-OPP p which is perpendicular to X_i -axis and it is between couplets $\Phi_k^i(p)$ and $\Phi_{k+1}^i(p)$. Algorithm 4 implements the above recursive function in order to compute the content of n D space enclosed by an n D-OPP p expressed in the EVM [1, 8].

Algorithm 4. Computing the content of n D space enclosed by an n D-OPP expressed in the n D-EVM.

```

Input: An  $n$ D-EVM  $p$ .
          The number  $n$  of dimensions.
Output: Content of  $n$ D space enclosed by  $p$ .
Procedure Content(EVM  $p$ , int  $n$ )
  real cont=0.0 //  $n$ D space enclosed by  $p$ .
  EVM hv11, hv12 // Couplets between
                  // a slice of  $p$ .
  EVM s // Current section of  $p$ .
  if( $n = 1$ ) then
    // Base case:  $p$  is a 1D-OPP.
    return Length( $p$ )
  else
    hv11 = InitEVM( )
    hv12 = InitEVM( )
    s = InitEVM( )
    hv11 = ReadHvl( $p$ )
    while(Not(EndEVM( $p$ )))
      hv12 = ReadHvl( $p$ )
      s = GetSection( $s$ , hv11)
      // Recursive Call.
      cont = cont +
        Content( $s$ ,  $n-1$ ) * dist(hv11, hv12)
      hv11 = hv12
    end-of-while
    return cont
  end-of-else
end-of-procedure

```

B. Computing the Content of the Boundary of an n D-OPP

In [1] is pointed out that the surface of a 3D-OPP p (see Fig. 4 for an example) can be computed as the sum of the areas of its 2D couplets perpendicular to X_i -axis, where the area of a $\Phi_k^i(p)$ is given by $Content_2(\Phi_k^i(p))$ (see previous section). To this sum must be added the sum of the areas of the faces between $\Phi_k^i(p)$ and $\Phi_{k+1}^i(p)$. These areas are found by the product between the perimeter of the section $S_k^i(p)$ and the distance between $\Phi_k^i(p)$ and

$\Phi_{k+1}^i(p)$ (the height of the 3D prism $Slice_k^i(p)$). Now let $q = S_k^i(p)$, we have reached the base case. The perimeter of the 2D-OPP q can be computed as [1]:

$$Perimeter(q) = x_1Sum(q) + x_2Sum(q)$$

where $x_1Sum(q)$ and $x_2Sum(q)$ denote the sum of the lengths of all brinks parallel to X_1 -axis and X_2 -axis, respectively.

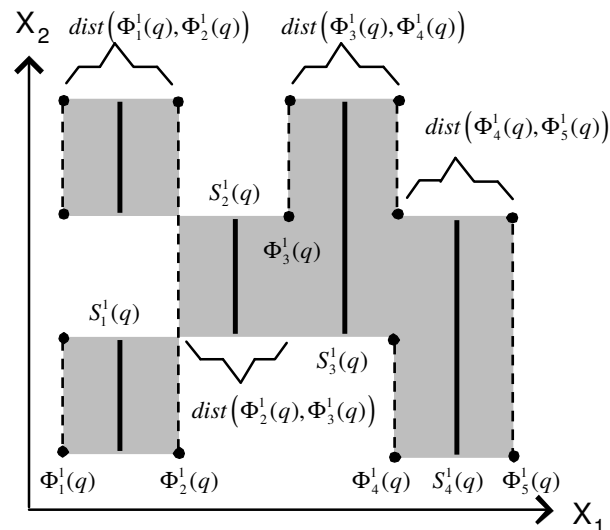


Fig. 3. A 2D-OPP q whose area is being computed: The total area of q is the sum of the areas of its slices.

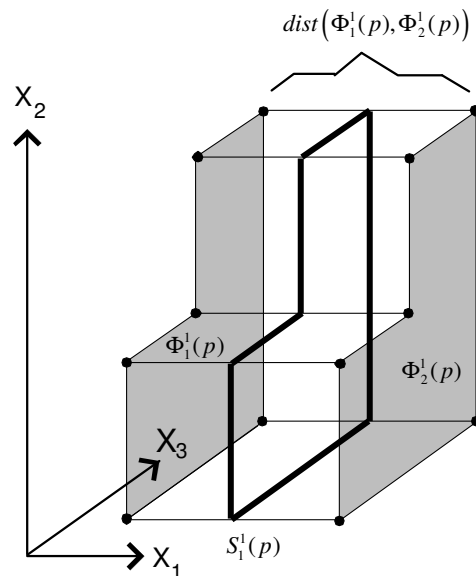


Fig. 4. Computing the content of the boundary in a 3D prism.

Let p be an n D-OPP. The $(n-1)$ D space enclosed by p 's boundary, denoted by $BoundaryContent_{(n)}(p)$, can be computed as follows [8]:

$BoundaryContent_{(n)}(p)$

$$= \begin{cases} x_1Sum(p) + x_2Sum(p) & n = 2 \\ \sum_{k=1}^{np_i} Content_{(n-1)}(\Phi_k^i(p)) + \sum_{k=1}^{np_i-1} \left[BoundaryContent_{(n-1)}(S_k^i(p)) \cdot dist(\Phi_k^i(p), \Phi_{k+1}^i(p)) \right] & n > 2 \end{cases}$$

Algorithm 5 implements the above recursive expression in order to compute the content of $(n-1)$ D space enclosed by the boundary of p expressed through the n D-EVM [1, 8].

Algorithm 5. Computing the content of $(n-1)$ D space enclosed by the boundary of a n D-OPP expressed in the n D-EVM.

```

Input: An  $n$ D-EVM  $p$ .
        The number  $n$  of dimensions.
Output: The content of  $(n-1)$ D space enclosed
        by the boundary of  $p$ .
Procedure BoundaryContent(EVM  $p$ , int  $n$ )
    // The content of  $(n-1)$ D space enclosed
    // by  $p$ 's boundary.
    real cont = 0.0
    // Couplets between a slice of  $p$ .
    EVM hv11, hv12
    EVM s // Current section of  $p$ .
    hv11 = InitEVM( )
    hv12 = InitEVM( )
    s = InitEVM( )
    if ( $n = 2$ ) then
        // Base case:  $p$  is a 2D-OPP, its
        // perimeter is computed.
        return  $x_1$ Sum( $p$ ) +  $x_2$ Sum( $p$ )
    else
        hv11 = ReadHvl( $p$ )
        while (Not(EndEVM( $p$ )))
            hv12 = ReadHvl( $p$ )
            s = GetSection(s, hv11)
            // Call to algorithm Content and
            // recursive call.
            cont = cont + Content(hv11,  $n-1$ ) +
                BoundaryContent(s,  $n-1$ ) *
                dist(hv11, hv12)
            hv11 = hv12
        end-of-while
        // hv11 contains  $p$ 's last couplet.
        cont = cont + Content(hv11,  $n-1$ )
        return cont
    end-of-else
end-of-procedure
    
```

IV. BASIC MORPHOLOGICAL OPERATIONS IN THE n D-EVM

Let C_{x_1, x_2, \dots, x_n} be a hypervoxelization (not necessarily binary) with length along the X_i -axis m_i , $i = 1, 2, \dots, n$. Let S be a subset of \mathbb{Z}^n . S is called the structuring element. The dilation of hypervoxelization C_{x_1, x_2, \dots, x_n} by S , denoted by $C \oplus S$, is another hypervoxelization defined as:

$$(C \oplus S)_{x_1, x_2, \dots, x_n} = \sup \left\{ \begin{array}{l} C_{x_1 - \bar{x}_1, x_2 - \bar{x}_2, \dots, x_n - \bar{x}_n} : \\ (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) \in S, 1 \leq x_i - \bar{x}_i \leq m_i, i = 1, 2, \dots, n \end{array} \right\}$$

In a similar fashion, the erosion of hypervoxelization C_{x_1, x_2, \dots, x_n} by S , $C \ominus S$, is given by:

$$(C \ominus S)_{x_1, x_2, \dots, x_n} = \inf \left\{ \begin{array}{l} C_{x_1 - \bar{x}_1, x_2 - \bar{x}_2, \dots, x_n - \bar{x}_n} : \\ (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n) \in S, 1 \leq x_i - \bar{x}_i \leq m_i, i = 1, 2, \dots, n \end{array} \right\}$$

The well known morphological erosion and dilation constitute the fundamentals behind other useful operators. Burgeth et al [2] list some of them:

- Opening: $C \circ S = (C \ominus S) \oplus S$
- Closing: $C \bullet S = (C \oplus S) \ominus S$
- White top-hat: $WTH(C) = C \setminus (C \circ S)$
- Black top-hat: $BTH(C) = (C \bullet S) \setminus C$

- Beucher gradient: $\nabla_s(C) = (C \oplus S) \setminus (C \ominus S)$
- Internal gradient: $\nabla_s^-(C) = C \setminus (C \ominus S)$
- External gradient: $\nabla_s^+(C) = (C \oplus S) \setminus C$

Rodríguez & Ayala [11] point out that dilation and erosion over binary hypervoxelizations can be understood as operations that only add or remove black hypervoxels in terms of the size and shape of the structuring element. Furthermore, they comment that erosion and dilation can be seen as processes where the object's interior is shrunk or elongated respect to the size of the structuring element. This reasoning constitutes the basis of their proposed algorithms for performing erosion and dilation over an n D-OPP expressed in the n D-EVM. Their idea is in fact very simple [11]:

- Given an input n D-OPP p , its $(n-1)$ D sections, perpendicular to X_A -axis are obtained.
- Current section S_j is an $(n-1)$ D-OPP that describes the interior of the current n D Slice of p . S_j is sent as input to the algorithm via a recursive call.
- The output of the recursive call is a $(n-1)$ D-OPP, actually a new section, that corresponds to the erosion (dilation) of section S_j . The eroded (dilated) section is used for determine the new $(n-1)$ D couplet before it.
- The obtained couplets define a new n D-OPP which corresponds to the erosion (dilation) of the input n D-OPP p .

As specified before, the procedure descends recursively over the number of dimensions. The base case is reached when $n = 1$. In this situation, it is obtained a set of segments embedded in X_Z -axis. In this phase, the algorithm directly shrinks (elongates) those segments according to an erosion (dilation) ratio. Hence, a box-shaped structuring element is used [11]. It is possible some segments can be merged or disappear depending which operation is applied. Algorithm 6 implements the procedure, originally presented in [11], for computing the erosion of an n D-OPP expressed in the EVM. Implementing dilation operation is made in similar way and taking in account the corresponding base case.

It is convenient to comment a property. An EVM is a set of points with a fixed coordinates ordering. Algorithm 6, in the main call, computes $(n-1)$ D sections perpendicular to X_A -axis, then, recursively, computes $(n-2)$ D sections perpendicular to the axis following X_A , and so on, until the base case is reached and the segments in the input 1D-OPP are embedded in X_Z -axis. As previously stated, is at this point where the erosion (dilation) takes place, defining new segments and for instance defining new lower dimensional sections until there are obtained the final $(n-1)$ D sections corresponding to the erosion (dilation) of the input n D-OPP. This implies Algorithm 6 erodes (dilates) the input OPP only respect to X_Z -axis [11]. For full erosion, along all the n main axes of the n D space, it is required to sort n times the coordinates in the input EVM in such way each time one different axis appears in the last place [11]. The Algorithm 7 computes the full erosion of an n D-OPP p . It receives as input an EVM, the number n of dimensions, and an array that contains the erosion ratios to be applied respect to each coordinates axis. It is assumed the i -th position in the array indicates the erosion ratio to apply along the X_i -axis. When such erosion is going to be performed, via Algorithm 6, then we have such coordinate must appear in the last position of the current coordinates ordering, i.e., $X_Z = X_i$.

The coordinates sorting is performed by calling a procedure *SortEVM*. Given a permutation $X_{a_1} X_{a_2} \dots X_{a_n}$, the function sorts the extreme vertices of p first according to coordinate X_{a_1} , after according to coordinate X_{a_2} , and so on until p is sorted according to coordinate X_{a_n} . An algorithm for full dilation is implemented in a similar fashion.

Algorithm 6. Eroding, in direction of X_z -axis, an nD -OPP expressed in the nD -EVM.

Input: An nD -EVM p .
The number n of dimensions.
The erosion ratio.

Output: An nD -EVM corresponding to the erosion of p along X_z -axis.

Procedure Erosion
(EVM p , int n , CoordType ratio)

```

if( $n = 1$ ) then
    // Base case:  $p$  is a 1D-OPP.
    return Erosion1D( $p$ , ratio)
else
    EVM hvl // Current couplet of  $p$ .
    EVM Si // Previous section to hvl.
    EVM Sj // Next section to hvl.
    EVM erodedP //  $p$ 's erosion.
    EVM erodedHvl // Couplets to append to
    //  $p$ 's erosion.
    EVM erodedSi // Previous section to
    // erodedHvl.
    EVM erodedSj // Next section to
    // erodedHvl.
    CoordType coord // Common  $X_a$ -coordinate
    // of couplet hvl.
    erodedP = InitEVM( )
    Si = InitEVM( )
    erodedSi = InitEVM( )
    while(Not (EndEVM( $p$ )))
        coord = GetCoordNextHvl( $p$ )
        hvl = ReadHvl( $p$ )
        Sj = GetSection(Si, hvl)
        // Recursive Call.
        erodedSj = Erosion(Sj,  $n-1$ , ratio)
        erodedHvl =
            GetHvl(erodedSi, erodedSj)
        SetCoord(erodedHvl, coord)
        PutHvl(erodedP, erodedHvl)
        Si = Sj
        erodedSi = erodedSj
    end-of-while
    return erodedP
end-of-else
end-of-procedure

```

Algorithm 7. Full erosion of an nD -OPP expressed in the nD -EVM.

Input: An nD -EVM p .
The number n of dimensions.
An array with erosion ratios.

Output: An nD -EVM corresponding to the full erosion of p .

Procedure FullErosion
(EVM p , int n , CoordType ratios[1, ..., n])

```

EVM erosionP =  $p$ 
int  $i = 0$ 
for each sorting in
    { $X_1 X_2 \dots X_{n-1} X_n$ ,  $X_n X_1 \dots X_{n-2} X_{n-1}$ ,  $X_{n-1} X_n X_1 \dots X_{n-3} X_{n-2}$ ,
    ...,  $X_2 X_3 \dots X_n X_1$ }
    SortEVM(erosionP,  $n$ , sorting)
    // Erosion along  $X_z$ -axis
    erosionP=Erosion(erosionP,  $n$ , ratios[ $n-i$ ])
     $i = i + 1$ 
end-of-for
return erosionP
end-of-procedure

```

Given algorithms for full erosion and full dilation under the nD -EVM, it is possible to implement, in straight way, morphological operators such as Opening (\circ) or Closing (\bullet). See Algorithms 8 and 9.

Algorithm 8. Computing the morphological opening of an nD -OPP.

Input: An nD -EVM p .
The number n of dimensions.
An array with erosion ratios.
An array with dilation ratios.

Output: An nD -EVM corresponding to the full dilation of the full erosion of p .

Procedure Opening
(EVM p , int n ,
CoordType erosionRatios[1, ..., n],
CoordType dilationRatios[1, ..., n])

```

EVM openingP =  $p$ 
openingP =
    fullErosion(openingP,  $n$ , erosionRatios)
openingP =
    fullDilation(openingP,  $n$ , dilationRatios)
return openingP
end-of-procedure

```

Algorithm 9. Computing the morphological closure of an nD -OPP.

Input: An nD -EVM p .
The number n of dimensions.
An array with erosion ratios.
An array with dilation ratios.

Output: An nD -EVM corresponding to the full erosion of the full dilation of p .

Procedure Closure
(EVM p , int n ,
CoordType erosionRatios[1, ..., n],
CoordType dilationRatios[1, ..., n])

```

EVM closureP =  $p$ 
closureP =
    fullDilation(closureP,  $n$ , dilationRatios)
closureP =
    fullErosion(closureP,  $n$ , erosionRatios)
return closureP
end-of-procedure

```

V. REPRESENTING VOLUME DATASETS THROUGH 4D ORTHOGONAL POLYTOPES

Now, we have the elements to proceed to present the contribution of this work. The conversion of a voxelization to a 4D polytope, and therefore to a 4D-EVM, is in fact a straight procedure. As commented in Section I, the proposal is oriented to color spaces with three or more values. If it is the case the datasets have binary intensity values then refer to [9] where a more appropriate representation is described. It is assumed the intensities of each voxel are expressed as a single number. Consider the following

Corollary 3 [1]: Let p and q be two disjoint or quasi disjoint nD -OPPs having $EVM_n(p)$ and $EVM_n(q)$ as their respective Extreme Vertices Models, then

$$EVM_n(p \cup q) = EVM_n(p) \otimes EVM_n(q)$$

An nD hyperprism is a polytope generated by the parallel motion of an $(n-1)D$ polytope; it is bounded by the $(n-1)D$ polytope in its initial and final positions and by several $(n-1)D$ hyperprisms [3, 14]. In this sense, each voxel in a 3D dataset is extruded towards the fourth dimension, that is, it is converted in a 4D hyperprism whose bases are precisely the original voxel and its height is given by its corresponding intensity-plus-one value in the dataset. The vertices' coordinates X_1 , X_2 , and X_3 in the hyperprism's bases correspond to the original voxel's coordinates. The inferior base's points have their X_4 coordinate equal to zero, while in the remaining vertices the X_4 coordinate is equal to the intensity-plus-one value (see Fig. 5). Let us call xf to the set composed by the 4D hyperprisms (the extruded voxels) of the extruded 3D dataset.

Let pr_i be a 4D hyperprism in xf and npr the number of prisms in that set (this number is in fact equal to the number

of voxels in the original dataset). Since all the hyperprisms in xf are quasi disjoint 4D-OPPs, the extreme vertices, of the whole 4D extruded dataset, can be easily obtained by computing the regularized union of all the hyperprisms in xf . Hence, Corollary 3 can be applied in the following way:

$$EVM_4(F) = \bigotimes_{i=1}^{npr} EVM_4(pr_i \in xf)$$

Where F is the 4D-OPP that represents the union of all the hyperprisms in xf . By this way, it is obtained a representation for a 3D Dataset through a 4D-OPP and the EVM.

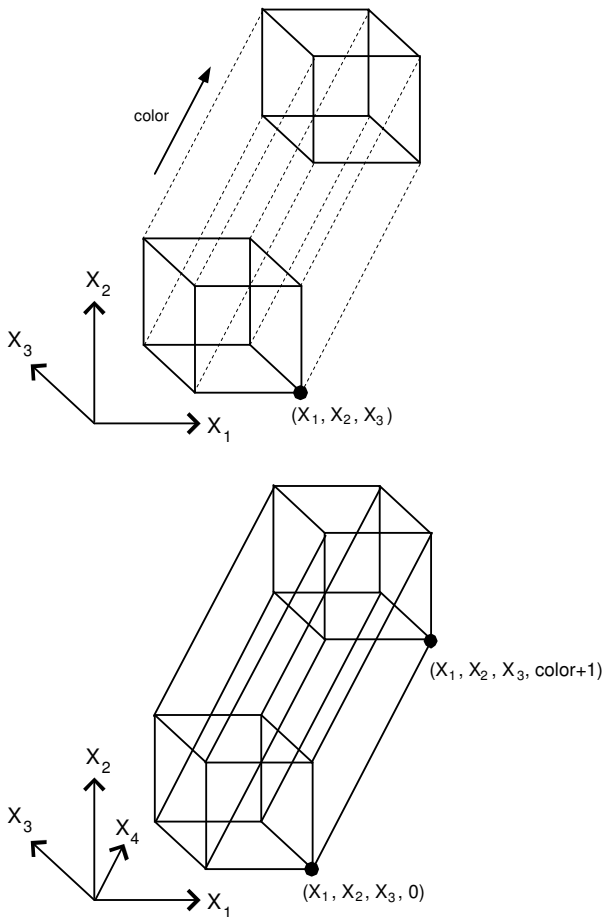


Fig. 5. Extruding a voxel towards the fourth dimension. The result is a 4D hyperprism whose height is determined by the voxel's intensity color.

VI. MODELING VOLUME DATASETS WITH 4D-OPPS AND THE EVM: CONCISENESS

In this section there are described some results related to the conversion from voxelizations to the proposed representation. Such datasets were taken from *The Volume Library* [12], and the University of Iowa's Department of Radiology [4] (datasets' visualizations shown in Table I and Fig. 6 were obtained via a volume rendering software available at [17]). Prior to the conversion, a multilevel thresholding was applied to the original datasets in order to reduce noise. The Table I shows the measures obtained when the considered voxelizations were converted to 4D-OPPs and the EVM.

Now, let p be a dataset expressed under a voxelization with size $(x_1Size \times x_2Size \times x_3Size)$ and with $EVM_4(p)$ as its corresponding EVM. Consider the ratio

$$\frac{x_1Size \cdot x_2Size \cdot x_3Size}{Card(EVM_4(p))}$$

For example, consider the model *CT-Hand* (Table I). Its source voxelization has size $(492 \times 240 \times 155)$ which implies that it is required to store 18,302,400 voxels. The 4D-EVM associated to *CT-Hand* has 7,191,726 extreme vertices. Hence, the proposed ratio gives the value 2.54 which implies that the number of stored voxels that belong to the original representation of the object is precisely 2.54 times greater than the number of obtained extreme vertices. The Table II shows the ratio Number-of-voxels/Number-of-Extreme-Vertices for the models described in Table I. The value shared by the ratio depends on the topology and geometry of the objects being modeled, but it shows the conciseness, related to storing requirements, when they are represented through the EVM.

VII. QUERYING VOLUME DATASETS EXPRESSED IN THE 4D-EVM

The importance behind a volume dataset is the information can be obtained about it. If the datasets are represented through the 4D-EVM then the extraction of its couplets will provide a classification of the elements in the original model according to their intensities. Given a dataset based on a scale of K intensities, then it is possible to obtain at most, from its corresponding 4D-OPP, $1+K$ 3D couplets perpendicular to the axis associated to color. The "extra" couplet, in fact the first couplet, is the result of the union of all the inferior bases of the 4D hyperprisms in xf (the set composed by the dataset's extruded voxels): the points in such bases have value zero for the coordinate associated to color (see Fig. 5).

It is common that intensities correspond to physical properties. For example, in a medical dataset intensities could refer to certain tissues. Hence, by extracting 3D couplets perpendicular to color axis we obtain, for a given couplet, only those parts of the dataset with the same type of tissue. The Table III shows some 3D couplets obtained from the dataset *CT-Hand* (3D-EVM visualizations from Tables III, V, and VI were achieved by means of software developed in [13]). In Table III, it can be observed that the first three couplets describe parts of the dataset whose voxels correspond to soft tissue like skin, cartilage, and muscles. The remaining couplets, also shown in the same table, show different types of bone tissue present in the voxelization. As commented previously, the type of material is, in this case, assumed to be defined by the color intensity of the voxels in the original dataset.

The projection of each couplet, perpendicular to the axis associated to color, in a 4D-EVM, is in fact a 3D-EVM which in time corresponds to a binary dataset because it only contains material of the same type. Hence, it is possible to apply Algorithm 3 to one of these 3D couplets in order to obtain their associated 2D sections. These sections will share a description of the interior of the homogeneous object with the objective of performing the appropriate analyses according to the application. The Table IV shows some 2D sections corresponding to one of the 3D couplets from the model *Vismale* (see Table I). There are shown 25 of these 2D sections which allow understand the internal organization of the selected 3D couplet. It is possible to visualize some internal organs.

The Algorithms 4 and 5 (Sections III.A and III.B, respectively) can be used for interrogate a nD -EVM in order to determine, for its corresponding nD -OPP, its nD content and the $(n-1)D$ content of its boundary. These algorithms

can be applied to the 4D-EVM associated to a dataset; in this sense its corresponding 4D content and the 3D content of its boundary are obtained. Furthermore, it is clear that the projection of each couplet perpendicular to color axis is a 3D-OPP which contains only elements of the same type of material. Hence, it is possible to apply Algorithms 4 and 5 in order to determine its volume and its boundary's area. By this way, it is obtained measure information about specific parts of the original dataset.

For example, Table V shows 3 couplets taken from the dataset *CT-Chest*. In 3D couplet 4 it is possible to observe soft tissue and the lungs. The 3D couplets 32 and 50 correspond to bone tissue where some ribs and the vertebral column can be observed (the scapulae can be visualized in 3D couplet 32; they are not present in couplet 50). Table V also presents the volume and the boundary's area of these 3D couplets.

TABLE I
VOLUME DATASETS USED FOR CONVERSION TO 4D-OPPS AND FINALLY EXPRESSED THROUGH THE 4D-EVM.


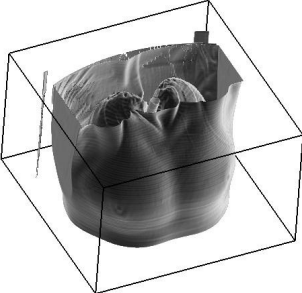
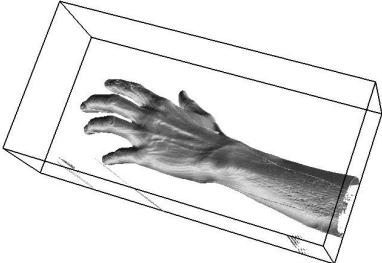
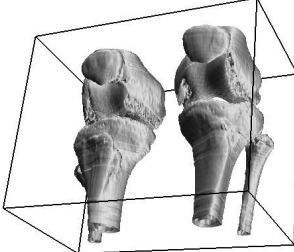
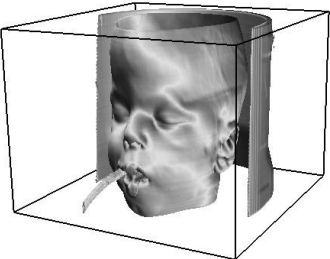
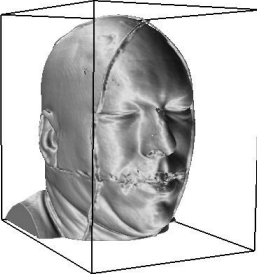
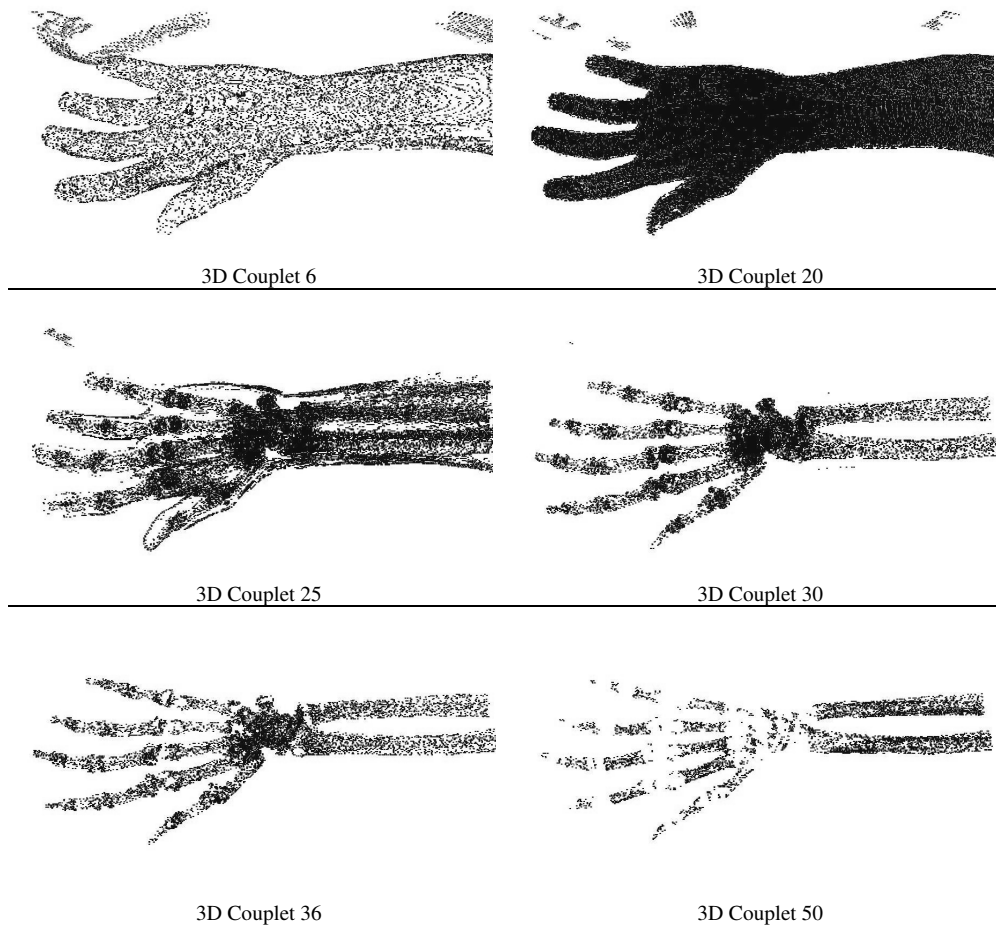
	
<p>Blunt Fin Description: Airflow over a flat plate with a blunt fin rising from the plate. Voxelization size: $(256 \times 128 \times 64) \cong 2,097,152$ 4D-EVM Size: 347,956</p>	<p>CT-Chest Description: Computed Tomography of a normal chest. Voxelization size: $(384 \times 384 \times 240) \cong 35,389,440$ 4D-EVM Size: 19,674,862</p>
	
<p>CT-Hand Description: Computed Tomography of a normal hand. Voxelization size: $(492 \times 240 \times 155) \cong 18,302,400$ 4D-EVM Size: 7,191,726</p>	<p>CT-Knee Description: Computed Tomography of knee, anterior tibial osteotomy. Voxelization size: $(379 \times 229 \times 305) \cong 26,471,255$ 4D-EVM Size: 17,938,182</p>
	
<p>Baby Description: baby head. Voxelization size: $(256 \times 256 \times 98) \cong 6,422,528$ 4D-EVM Size: 2,975,618</p>	<p>Vismale Description: Visible male (head). Voxelization size: $(128 \times 256 \times 256) \cong 8,388,608$ 4D-EVM Size: 6,300,952</p>

TABLE II
THE RATIO NUMBER-OF-VOXELS/NUMBER-OF-EXTREME-VERTICES FOR DATASETS SHOWN IN TABLE I.

Object p	Voxelization Size (Number of voxels)	Card($EVM_4(p)$) (Number of extreme vertices)	$\frac{x_1 \text{Size} \cdot x_2 \text{Size} \cdot x_3 \text{Size}}{\text{Card}(EVM_4(p))}$
<i>Blunt Fin</i>	2,097,152	347,956	6.02
<i>CT-Hand</i>	18,302,400	7,191,726	2.54
<i>Baby</i>	6,422,528	2,975,618	2.15
<i>CT-Chest</i>	35,389,440	19,674,862	1.79
<i>CT-Knee</i>	26,471,255	17,938,182	1.47
<i>Vismale</i>	8,388,608	6,300,952	1.33

TABLE III
SOME 3D COUPLETS, PERPENDICULAR TO COLOR AXIS, EXTRACTED FROM THE 4D-EVM ASSOCIATED TO *CT-HAND*.



VIII. STORAGE AND NOISE REDUCTION VIA 4D MORPHOLOGICAL EROSION AND DILATION

It is well known the “Salt and Pepper” noise in a binary dataset refers to the random presence, due to the acquisition process, of white (the salt) or black (the pepper) components [7]. The literature mentions some methodologies for reduction of this type of noise such as Median Filtering, Chain Code Processing, Instantiation to the Shortest Path Problem, etc.

In particular, we will concentrate now in the reduction of “Salt and Pepper” noise via morphological erosion and dilation. Section IV dealt with Rodríguez & Ayala’s EVM-based algorithms for performing erosion and dilation over an n D-OPP. As commented in past sections, each 3D couplet perpendicular to color axis, in our generated 4D-OPPs, contains material of the same type because our procedure groups the elements of a dataset relating them by their color intensity. In this sense, we said each 3D couplet corresponds to a binary voxelization. We can establish then our 4D-OPPs can be seen as a “binary expression” of 3D volume datasets because they partition the 4D space in two sets: the black (occupied) regions and the white (empty) regions.

The algorithms presented in Section IV were designed specifically under the context of eroding (dilating) binary images: the black regions defined the OPP to erode (dilate) [11]. According to our previous discussion, our generated 4D-OPPs can be used as input for purposes of erosion (dilation). The idea is the reduction of “Salt and Pepper”

noise of the corresponding volume dataset via a 4D context obtaining as benefit the deletion of undesired black components, the filling of undesired white components, and contour smoothing. We will see how the application of such operators has repercussions over geometry and topology of the elements that compose a represented volume dataset. Moreover, another important benefit that can be obtained via the application of erosion (dilation) is the reduction of the 4D representation’s spatial complexity, i.e. the EVM’s size.

Consider the dataset *CT-Pelvi* [4] (Fig. 6). It corresponds to a computed tomography of a human’s pelvis. The dataset has size $(358 \times 358 \times 151) \equiv 20,812,330$ voxels. Its representation via 4D-EVM required 17,985,162 extreme vertices hence its ratio Number-of-voxels/Number-of-Extreme-Vertices is 1.15.

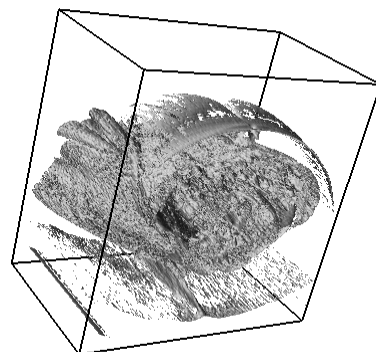
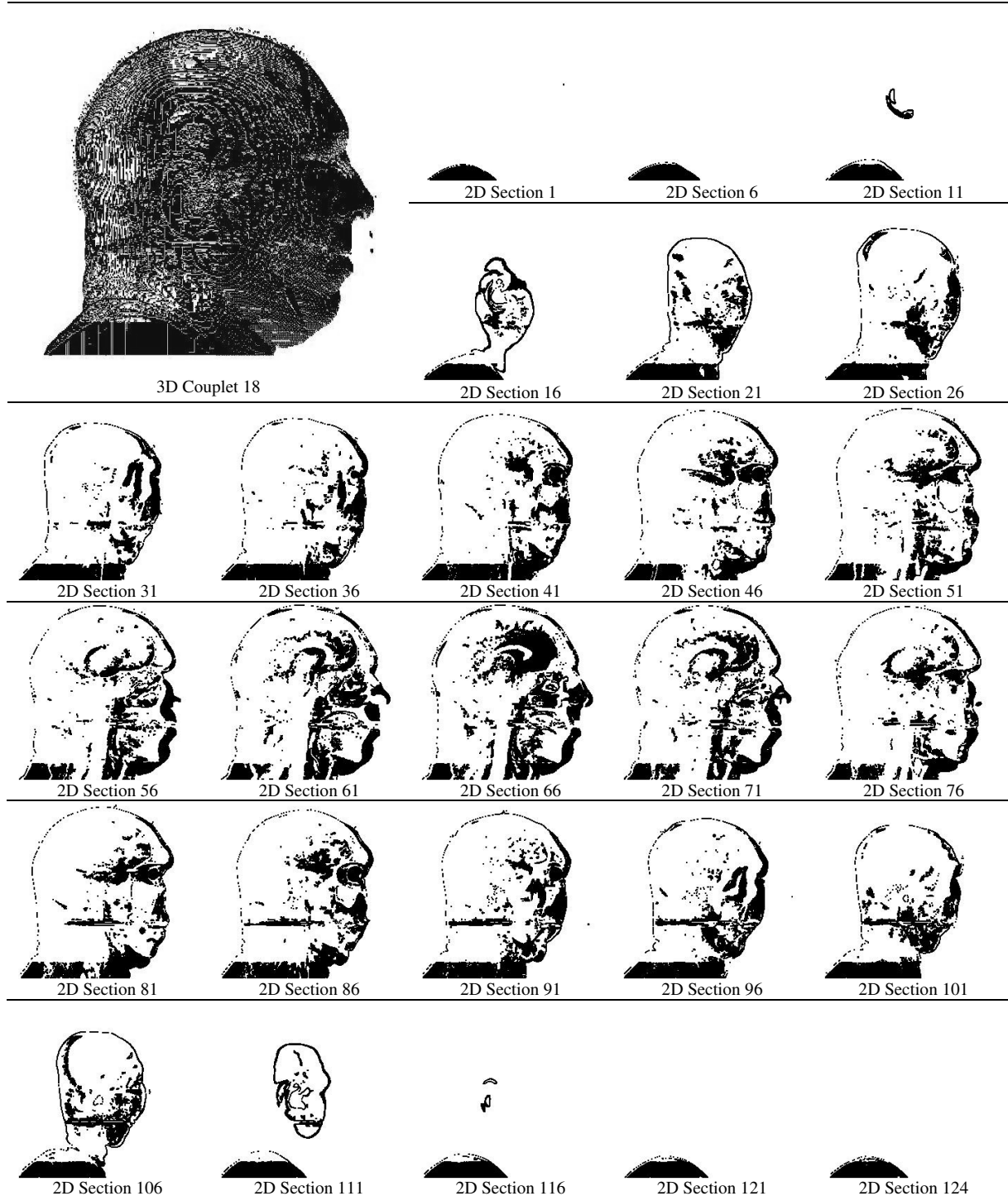


Fig. 6. Dataset *CT-Pelvi* (see text for details).

TABLE IV
 VISUALIZING SOME 2D SECTIONS PERPENDICULAR TO X_7 -AXIS WHICH WERE PROCESSED FROM ONE 3D COUPLET
 OF THE 4D-OPP ASSOCIATED TO THE DATASET *VISMALE*.



Erosion and dilation were applied separately to *CT-Pelvi*'s 4D-EVM representation. In both cases were used the ratios [1.0, 1.0, 1.0, 0.0]. This implies erosion/dilation were applied by shorting/elongating a unit in all directions, except the one related to color axis because these operations could modify the values of such coordinate and therefore the correspondence with the original color scale could be altered. In the erosion's case it was obtained a new 4D representation that required 7,531,384 extreme vertices. The dilation's 4D representation required 9,355,148 extreme vertices. The new ratios Number-of-voxels/Number-of-Extreme-Vertices were 2.76 and 1.92, respectively.

The Table VI shows some 3D couplets extracted from the 4D-EVMs modeling the erosion and dilation of the original 4D representation of *CT-Pelvi* dataset. Corresponding 3D couplets from the original 4D-OPP are also included in the table for visual comparison. Couplet 7 shows some parts of skin tissue while couplet 17 shares the visualization of soft tissue and some internal organs. In both cases, erosion and dilation produced couplets that required a number of extreme vertices minor than the size of the couplet associated to the original 4D representation. Couplets 24 and 34 correspond to bone tissue: there can be appreciated the coccyx, sacrum, iliac crest, femurs, and hip

bones, among others. The erosion of couplet 24 has more extreme vertices than its dilation. However, both sizes are minor than the 184,354 extreme vertices in the original couplet. Because no erosion/dilation was applied respect to color axis then each couplet, in the new representations, contains material of a type described appropriately by the original color scale.

The observations related to the obtained 3D couplets are sustained in the fact erosion/dilation act, from the EVM-based algorithms' point of view, by modifying the interior of their corresponding 4D-OPPs, specifically operating over their 3D sections. The Rodríguez & Ayala algorithms, once the sections have been modified, compute new 3D couplets which by instance contain changes in their topology and geometry. The presented example leads us to establish the way erosion/dilation operations can be useful tools, applied in a 4D context, on one hand, for reducing noise present in the original dataset, while on the other hand, it is possible to reduce the spatial complexity of the representation.

IX. CONCLUDING REMARKS AND FUTURE WORK

The Extreme Vertices Model (3D-EVM) was originally presented, and widely described, in [1] for representing Orthogonal Pseudo-Polyhedra (3D-OPPs). The model has

enabled the development of simple and robust algorithms for performing the most usual and demanding tasks on solid modeling, such as closed and regularized Boolean operations, solid splitting, set membership classification operations, and measure operations on 3D-OPPs. In [8] was formally proved that the EVM in fact is a complete scheme for the representation of n -Dimensional Orthogonal Pseudo-Polytopes (n D-OPPs). The meaning of complete scheme was based in Requicha's set of formal criterions that every scheme must have rigorously defined: Domain, Completeness, Uniqueness and Validity. Although the EVM of an n D-OPP has been defined as a subset of the n D-OPPs vertices, which in principle defines the model's conciseness, there is much more information about the polytope hidden within this subset of vertices. Moreover, several operations can be performed efficiently and working directly and only with the EVM representation. In Section II.G was presented a specific algorithm for computing XOR in linear time: the MergeXor algorithm. The algorithm's linear complexity is a valuable property when it is observed the n D-EVM's intensive use of Regularized XOR in several fundamental operations such as the computation of sections of an n D-OPP.

TABLE V
SOME 3D COUPLETS EXTRACTED FROM THE 4D-EVM ASSOCIATED TO DATASET CT-CHEST
AND INTERROGATIONS ABOUT THEIR MEASURES: VOLUME AND BOUNDARY'S AREA.

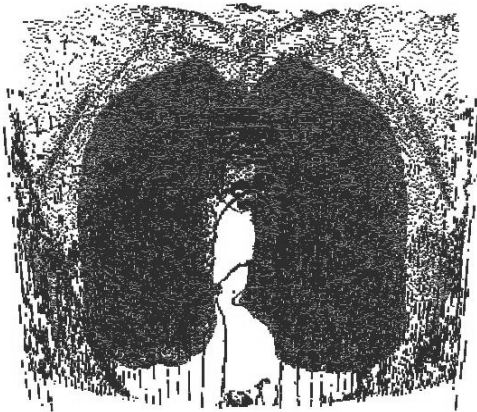
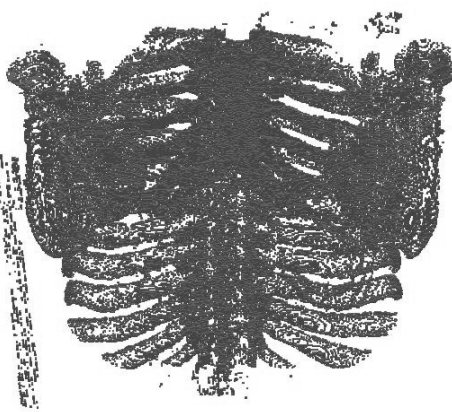
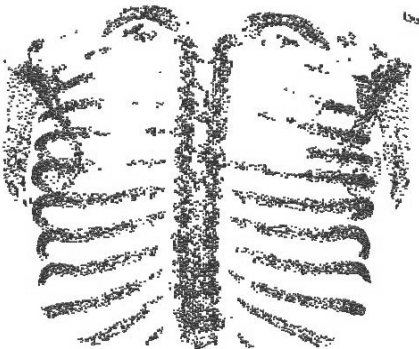

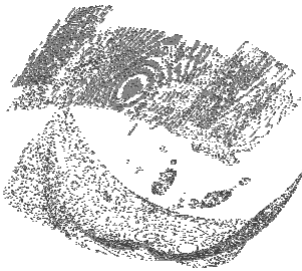

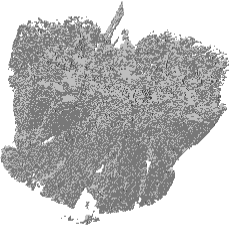
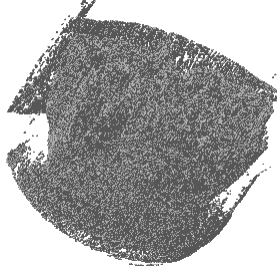
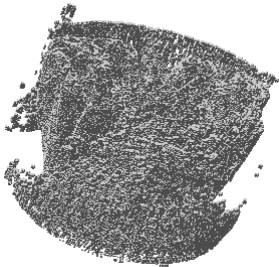






	
<p>3D Couplet 4 3D Content (Volume): $264,101 u^3$ Boundary's Area: $1,111,298 u^2$</p>	<p>3D Couplet 32 3D Content (Volume): $156,630 u^3$ Boundary's Area: $578,954 u^2$</p>
	
<p>3D Couplet 50 3D Content (Volume): $14,904 u^3$ Boundary's Area: $79,736 u^2$</p>	

TABLE VI

SOME 3D COUPLETS EXTRACTED FROM THE 4D-EVMS ASSOCIATED TO EROSION AND DILATION OF THE 4D REPRESENTATION OF DATASET *CT-PELVI*

3D Couplet	Erosion	Original	Dilation
7			
	54,826 Extreme Vertices	73,768 Extreme Vertices	64,368 Extreme Vertices
17			
	631,242 Extreme Vertices	1,759,508 Extreme Vertices	675,054 Extreme Vertices
24			
	125,906 Extreme Vertices	184,354 Extreme Vertices	124,252 Extreme Vertices
34			
	21,754 Extreme Vertices	94,042 Extreme Vertices	88,858 Extreme Vertices

In this work we have focused about a proposal for the representation of volume datasets through four-dimensional orthogonal polytopes which in time are expressed in the EVM. The evaluations presented in Section VI lead to establish that an important level of conciseness is obtained when such voxelizations are modeled according to our four-dimensional context. Moreover, in Sections VII and VIII we have described how the basic procedures under the Extreme Vertices Model share the extraction of useful information related to the original datasets: 1) the classification of their elements according to intensities via 3D couplets perpendicular to color axis; 2) analysis of the geometry and topology of those parts of a dataset which are composed by the same type of material via the computation

of 2D sections; 3) measure interrogations in order to know volumes or boundaries' areas; and, 4) direct manipulation of the 4D representation for reducing noise and spatial complexity. It is possible to compute other geometrical and topological interrogations over an EVM. By this way it could be obtained much more information and properties about the datasets. Furthermore, there are well specified procedures under the nD -EVM which allow performing Regularized Boolean Operations, Polytopes Splitting, Discrete Compactness Computation, among others. In [1, 8, 10, 11] there are described with enough detail algorithms based in the nD -EVM, besides the ones described in this work, which are useful and efficient for performing these interrogations and/or manipulations.

According to Section VII, if the volume datasets are represented through the 4D-EVM then the extraction of its couplets will provide a classification of the elements in the original model according to their intensities. The projection of each couplet, perpendicular to the axis associated to color is in fact a 3D-EVM which in time corresponds to a binary dataset because it only contains material of the same type. This observation opens a new line of future research because the next logical step is given by the consideration of additional approaches in such way the points in a volume dataset could be characterized by taking in account not only their color. Due to the presence of scanning noise and artifacts, a classification based only in intensities is sometimes not enough because points that should belong to the same class could be characterized as distinct points and therefore assigned to distinct classes (hence, our 4D representation will integrate each one in distinct 3D couplets). An intelligent approach, such as an Artificial Neural Network, could automatically identify the classes of points present in a volume dataset by taking in account their geometry, topology, neighborhood, etc. Once the classification is achieved, then the corresponding conversion to our proposed four-dimensional representation could be performed. It could be possible to assign to each class a coordinate in the 4D space in such way 3D couplets perpendicular to the fourth axis will correspond to sets of points, in the original dataset, that belong to the same class. By observing the results from Table II it is clear expressing volume datasets via the 4D-EVM has advantages in terms of storing requirements. It is expected, that by considering more appropriate points classification, a much better conciseness should be obtained.

REFERENCES

- [1] A. Aguilera, *Orthogonal Polyhedra: Study and Application*, PhD Thesis, Universitat Politècnica de Catalunya, 1998.
- [2] B. Burgeth, N. Papenberg, A. Bruhn, M. Welk, C. Feddern, J. Weickert, J. "Morphology for Higher-Dimensional Tensor Data via Loewner Ordering", *Computational Image and Vision*, 30:407-418, 2004.
- [3] H.S.M. Coxeter, *Regular Polytopes*, Dover Publications, Inc., New York, 1963.
- [4] Department of Radiology, University of Iowa. Web Site (visited in September, 2009): <http://radiology.uiowa.edu/downloads/>
- [5] A. Jonas, N. Kiryati, "Digital Representation Schemes for 3-D Curves", *Technical Report CC PUB #114*, The Technion - Israel Institute of Technology, Haifa, Israel, 1995.
- [6] G. Klajnsek, B. Rupnik, D. Spelic, "An Improved Quadtree-based Algorithm for Lossless Compression of Volumetric Datasets", *6th WSEAS Int. Conference on Computational Intelligence, Man-Machine Systems and Cybernetics*, 1:264-270, Spain, December 2007.
- [7] S. Marchand-Maillet, Y.M. Sharaiha, *Binary Digital Image Processing: A Discrete Approach*, Academic Press, 1999.
- [8] R. Pérez-Aguila, *Orthogonal Polytopes: Study and Application*, PhD Thesis, Universidad de las Américas - Puebla (UDLAP), 2006. Available at: http://catarina.udlap.mx/u_dl_a/tales/documentos/dsc/perez_a_r/
- [9] R. Pérez-Aguila, "Modeling and Manipulating 3D Datasets through the Extreme Vertices Model in the n-Dimensional Space (nD-EVM)", *Research in Computer Science, Special Issue: Industrial Informatics*, México, 31:15-24, 2007.
- [10] R. Pérez-Aguila, "Computing the Discrete Compactness of Orthogonal Pseudo-Polytopes via Their nD-EVM Representation," *Mathematical Problems in Engineering*, vol. 2010, Article ID 598910, 28 pages, 2010. doi:10.1155/2010/598910
- [11] J. Rodríguez, D. Ayala, "Erosion and Dilatation on 2D and 3D Digital Images: A new size-independent approach", *Vision Modeling and Visualization 2001*, Germany, 1:143-150, 2001.
- [12] S. Roettger, The Volume Library. Web Site (visited in September, 2009): <http://www9.informatik.uni-erlangen.de/External/vollib/>

- [13] R. Ruiz-Rodríguez, *Implementación del EVM (Extreme Vertices Model) en Java*, M.Sc Thesis, Universidad de las Américas – Puebla (UDLAP), 2002. Available at: http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/ruiz_r_r/
- [14] D.M.Y. Sommerville, *An Introduction to the Geometry of N Dimensions*, Dover Publications Inc., 1958.
- [15] W. Song, S. Hua, Z. ou, H. An, K. Song, "Octree Based Representation and Volume Rendering of Three-Dimensional Medical Data Sets", *International Conference on BioMedical Engineering and Informatics 2008*, 1:316-320, 2008.
- [16] M. Spivak, *Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus*, HarperCollins Publishers, 1965.
- [17] University of Tübingen. The Official Volren and Volvis Homepage. Web Site (visited in September, 2009): <http://www.gris.uni-tuebingen.de/edu/areas/scivis/volren/software/software.html>

Ricardo Pérez-Aguila received his BSc (2001), MSc (2003) and PhD (2006) degrees in Computer Science from the Universidad de las Américas-Puebla (UDLAP). In 2003-2006 he worked in the Actuarial Sciences, Physics and Mathematics Department at the same institution. In 2007 he incorporated as a full time faculty member at the Universidad Tecnológica de la Mixteca (UTM). He is Candidate to National Researcher of México's Researchers National System (SNI-Conacyt). His interests consider the study of n -Dimensional Polytopes by analyzing their Visualization, Geometry, Topology, Representation, and Applications. In the Neural Networks field he has been particularly interested in the Neural Network Architectures based in Non-Supervised Training.