

Research Article

Computing the Discrete Compactness of Orthogonal Pseudo-Polytopes via Their nD -EVM Representation

Ricardo Pérez-Aguila

*Computer Engineering Institute, The Technological University of the Mixteca (UTM),
Carretera Huajuapán-Acatlilma Km. 2.5, Huajuapán de León, 69000 Oaxaca, Mexico*

Correspondence should be addressed to Ricardo Pérez-Aguila, ricardo.perez.aguila@gmail.com

Received 17 February 2010; Accepted 1 July 2010

Academic Editor: Mohammad Younis

Copyright © 2010 Ricardo Pérez-Aguila. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This work is devoted to present a methodology for the computation of Discrete Compactness in n -dimensional orthogonal pseudo-polytopes. The proposed procedures take in account compactness' definitions originally presented for the 2D and 3D cases and extend them directly for considering the nD case. There are introduced efficient algorithms for computing discrete compactness which are based on an orthogonal polytopes representation scheme known as the Extreme Vertices Model in the n -Dimensional Space (nD -EVM). It will be shown the potential of the application of Discrete Compactness in higher-dimensional contexts by applying it, through EVM-based algorithms, in the classification of video sequences, associated to the monitoring of a volcano's activity, which are expressed as 4D orthogonal polytopes in the space-color-time geometry.

1. Introduction

In areas such as image processing, pattern recognition, and computer vision, there is required to characterize for a given object its topological and geometrical factors. They have a paramount role in more elaborated tasks such as those related to classification, indexing, or comparison. Some of these factors describe the shape of an object. One of them, and one of the most used, is the shape compactness [1]. The shape compactness of an object refers to a measure between the object and an ideal object [2]. In the 2D euclidean space, shape compactness is usually computed via the well-known ratio

$$C = \frac{P^2}{4\pi A}, \quad (1.1)$$

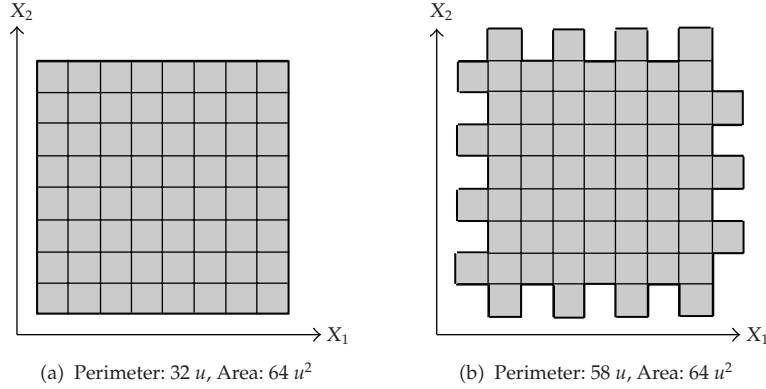


Figure 1: Polygons defined by the union of 64 unitary boxes.

where P is the perimeter of an object and A its area. Such ratio has its origins in the isoperimetric inequality:

$$P^2 \geq 4\pi A. \quad (1.2)$$

It is actually the solution to the isoperimetric problem which states the question related to find the simple closed curve that maximizes the area of its enclosed region [3]. The equality is obtained when the considered curve is a circle. Hence, as pointed out in [1], the ratio for shape compactness is in effect comparing an object with a circle. In the 3D space the isoperimetric inequality is given by

$$A^3 \geq 36\pi V^2, \quad (1.3)$$

where A is the area of the boundary of a 3D object while V is its volume. Hence, the ratio

$$C = \frac{A^3}{36\pi V^2} \quad (1.4)$$

denotes shape compactness of a 3D object, and it effectively is comparing such object with a sphere.

As [1, 4] point out, these classical ratios are very sensitive to variations in the shape of an object. Moreover, they point out, when the above definitions are applied to objects defined via pixelizations (in the 2D case) or voxelizations (3D case), that small changes in the final object's boundary produce more important variations in the computed values. Consider, for example, the sets of boxes presented in Figure 1. The square described by the union of the boxes shown in Figure 1(a) has a perimeter of $32u$ while its area is $64u^2$. Figure 1(b) shows a polygon that can be seen as a modified version (because of noise, artifacts, digitalization scheme, etc.) of the previous one. Its perimeter is given by $58u$. Both polygons have the same area, but their shapes have some slight differences. Shape compactness for the first polygon is given by 1.2732 while for the second is 4.1827. These values are significantly distinct, and

by considering shape compactness as a rule for classification, this could imply that they are very different objects.

In order to provide a solution to the above problem, Bribiesca, in [1, 5], defined the Discrete Compactness. It has its foundation on the notion of counting the number of edges (in the 2D case) and faces (in the 3D case) which are shared between pixels or voxels, according to the case, that define an object. Discrete Compactness is given by the following expression [1, 5]:

$$C_D(p) = \frac{L_C(p) - L_{C_{\min}}}{L_{C_{\max}} - L_{C_{\min}}}, \quad (1.5)$$

where

- (i) $L_C(p)$ is the number of shared edges (faces) within an object p of m pixels (voxels)
- (ii) $L_{C_{\max}}$ is the the maximum number of shared edges (faces) achieved with an object consisting of m pixels (voxels)
- (iii) $L_{C_{\min}}$ is the minimum number of shared edges (faces) achieved with an object consisting of m pixels (voxels)
- (iv) $C_D(p) \in [0, 1]$.

In [1] there are used, for the 2D case, $L_{C_{\max}} = 2(m - \sqrt{m})$ and $L_{C_{\min}} = m - 1$, which, respectively, describe the maximum and minimum number of internal contacts (shared edges) between the m pixels forming a squared object. It is clear, in this case, when $C_D(p) = 1$ the object p corresponds to a square of sides \sqrt{m} , and when $C_D(p) = 0$ it corresponds to a rectangle with base of length 1 and height m . For example, considering again the polygons presented in Figure 1, we have $L_C = 112$ for that shown in Figure 1(a), while $L_C = 99$ for the polygon in Figure 1(b). In both cases $m = 64$, hence, $L_{C_{\max}} = 112$ and $L_{C_{\min}} = 63$. Then, discrete compactness for the polygon in Figure 1(a) is given by $C_D = 1$ while the other has $C_D = 0.7346$. In [4] it is established that $L_{C_{\min}} = 0$. Hence, if $C_D(p) = 0$, then the object corresponds to a chain of pixels such that no edges, and only vertices, are shared. By considering $L_{C_{\min}} = 0$ then discrete compactness for the polygons in Figures 1(a) and 1(b) are 1 and 0.8839, respectively. It both cases, it is clear that discrete compactness provides us a more robust criterion for objects' comparison/classification/description of shapes under the advantage it is much less sensitive to variations in their shape. For the 3D case, in [1] it is used $L_{C_{\max}} = 3(m - m^{2/3})$. If m is a power of 3, then the given $L_{C_{\max}}$ provides the number of shared faces in an array of voxels that correspond to a cube of edges of length $\sqrt[3]{m}$. By using $L_{C_{\min}} = m - 1$ then it is defined a stack of m voxels [1].

2. Problem Statement

An n -dimensional Euclidean polytope Π_n is defined as a finite region of n -dimensional Euclidean space enclosed by a finite number of $(n - 1)$ -dimensional hyperplanes [6]. The finiteness of the region implies that the number N_{n-1} of bounding hyperplanes satisfies the inequality $N_{n-1} > n$. The part of the polytope that lies on one of these hyperplanes is called a cell. Each cell of a Π_n is an $(n - 1)$ -Dimensional polytope, Π_{n-1} . The cells of a Π_{n-1} are Π_{n-2} 's, and so on; thus it is obtained a descending sequence of elements $\Pi_{n-3}, \Pi_{n-4}, \dots, \Pi_3$ (a volume), Π_2 (a polygon), Π_1 (an edge), Π_0 (a vertex).

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 [7], are hyper-boxes (hypercubes, for example) of a fixed size that lie in a fixed grid in the n -dimensional 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 [7]. The collection of hyperboxes can be codified as an n -Dimensional array C_{x_1, x_2, \dots, x_n} . The array will represent the coloration of each hypervoxel. If $C_{x_1, x_2, \dots, x_n} = 0$, the white hypervoxel C_{x_1, x_2, \dots, x_n} represents an unoccupied region from the n -Dimensional space. If $C_{x_1, x_2, \dots, x_n} = 1$, then the black hypervoxel C_{x_1, x_2, \dots, x_n} represents a used region from the n -Dimensional space. Hence, the set of black cells defines a polytope whose vertices coincide with some of the occupied cells' vertices.

It is clear that Bribiesca's definition of Discrete Compactness can be extended for considering n -Dimensional polytopes, and in the particular dominion of this work, n -Dimensional Orthogonal Pseudo-Polytopes (n D-OPPs). We will consider those n D-OPPs that can be seen as the result of a hypervoxelization such that hypervoxels are unit n D hypercubes with integer coordinates. It is well known that an n D hypercube has $2n$ boundary $(n - 1)$ D cells [8]: 2 vertices in a segment, 4 edges in a square, 6 faces in a cube, 8 volumes in a 4D hypercube, and so on. Hence, $L_C(p)$ denotes the number of shared $(n - 1)$ D cells within polytope p consisting of m hypervoxels while $L_{C_{\max}}$ and $L_{C_{\min}}$ correspond, respectively, to the maximum and minimum number of shared $(n - 1)$ D cells achieved with objects composed by m hypervoxels.

When using hypervoxelizations for representing and manipulating n D polytopes some compromises should be taken in account. In particular is the one related to the representation's spatial complexity. It is well known that spatial complexity of a hypervoxelization is at least

$$\prod_{i=1}^n m_i, \quad (2.1)$$

where $m_i, 1 \leq i \leq n$, is the length of the grid along X_i -axis. For example, a 4D grid with $m_1 = m_2 = m_3 = m_4 = 1,000$ is required to store 1 trillion (1×10^{12}) hypervoxels. For that reason, in this work we will concentrate on expressing n D-OPPs via a polytopes' concise representation scheme known as the Extreme Vertices Model in the n -Dimensional Space (n D-EVM). The EVM is a model originally established by Aguilera and Ayala in [9, 10] for representing 1D, 2D, and 3D-OPPs. In [11], EVM's properties in the n -Dimensional Euclidean Space were formally proved, leading to characterize it as a complete representation scheme for n D-OPPs. In [10, 11] there are described efficient algorithms for performing some of the most used operations in solid and polytopes modeling such as regularized boolean operations, set membership classification, and measure queries. The conciseness of the EVM lies in the fact it only stores some specific vertices of a polytope: the Extreme Vertices. Via such subset of the polytope's vertices, it is possible to obtain much geometrical and topological information about the polytope and to perform operations as the ones previously mentioned. Now, returning to the question related to hypervoxelizations' complexity, we commented before that in a hypervoxelization the set of black hypervoxels defines a polytope, actually an n D-OPP. The hypervoxels in such set can be seen as a set of quasidisjoint n D-OPPs. Based on this observation, we will describe a straight methodology for converting a

hypervoxelization to a concise EVM representation. In Section 3 we will describe formally to nD -OPPs (Section 3.1), and we will present a summary of the foundations behind the nD -EVM (Section 3.2). Section 3.3 presents basic algorithms under the nD -EVM while Section 3.4 deals with two algorithms for interrogating and manipulating nD -OPPs represented via the nD -EVM: the first one computes the nD content of an OPP while the second corresponds to computation of regularized boolean operations between OPPs. Finally, Section 3.5 presents the procedure for hypervoxelizations to nD -EVM conversion.

The main contribution of this work is the specification of efficient procedures that provide us a way to infer the number of hypervoxels that originally composed an nD -OPP. In this sense, we assume that such OPP fits exactly in a hypervoxelization, where hypervoxels are unit nD hypercubes whose vertices have integer coordinates. More specifically, given an EVM we will determine the number of internal contacts, that is, the number of shared $(n - 1)D$ cells, that took place between the hypervoxels that originally defined the polytope. By this way, we will present in Section 4 an EVM-based algorithm for determining $L_C(p)$ for a given nD -OPP p . Such algorithm will lead to the specification of a procedure for computing the Discrete Compactness of an nD -OPP. After that, in Section 5, it will be described a methodology, originally presented in [11, 12], for representing and manipulating color 2D animations via EVM-modeled polytopes embedded in 4D space-color-time geometry. Then, in Section 6, we will describe an application of our proposed EVM-based algorithms for computing Discrete Compactness. Specifically, we present an application oriented to classification of image sequences which correspond to a volcano's activity. It will be seen how the obtained results provide experimental evidence about the applicability of Discrete Compactness in higher dimensional contexts.

3. The Extreme Vertices Model in the n -Dimensional Space (nD -EVM)

This section is a summary of results originally presented in [10, 11]. For the sake of brevity, some propositions are only enunciated. Their corresponding proofs can be found in the two aforementioned references.

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

Definition 3.1 (see [13]). A *Singular n -dimensional hyper-box* in \mathbb{R}^n is given by the continuous function

$$\begin{aligned} I^n : [0, 1]^n &\longrightarrow [0, 1]^n, \\ x &\sim I^n(x) = x. \end{aligned} \tag{3.1}$$

A *general singular k -dimensional hyper-box* in the closed set $A \subset \mathbb{R}^n$ is the continuous function

$$c : [0, 1]^k \longrightarrow A. \tag{3.2}$$

Definition 3.2 (see [13]). 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 $I_{(i,0)}^n(x) = (x_1, \dots, x_{i-1}, 0, x_i, \dots, x_{n-1})$ and $I_{(i,1)}^n(x) = (x_1, \dots, x_{i-1}, 1, x_i, \dots, x_{n-1})$.

Definition 3.3 (see [13]). The *orientation* of an $(n-1)$ D cell $c \circ I_{(i,\alpha)}^n$ is given by $(-1)^{\alpha+i}$. It is said cell $c \circ I_{(i,\alpha)}^n$ is *oriented* when it is expressed by the scalar-function product $(-1)^{i+\alpha} \cdot c \circ I_{(i,\alpha)}^n$.

Definition 3.4 (see [13]). A formal linear combination of general singular k D hyper-boxes, $1 \leq k \leq n$, for a closed set A is called *ak-chain*.

Definition 3.5 (see [13]). The *boundary* of an n -chain $\sum c_j$, where each c_j is a general singular n D hyper-box, is given by

$$\partial\left(\sum c_j\right) = \sum \partial(c_j) = \sum \left(\sum_{i=1}^n \left(\sum_{\alpha=0,1} (-1)^{i+\alpha} \cdot c_j \circ I_{(i,\alpha)}^n \right) \right). \quad (3.3)$$

Definition 3.6. A collection c_1, c_2, \dots, c_k , $1 \leq k \leq 2^n$, of general singular n D hyper-boxes is a *combination of n D hyper-boxes* if and only if

$$\left[\bigcap_{\alpha=1}^k c_\alpha([0,1]^n) = \left(\underbrace{0, \dots, 0}_n \right) \right] \wedge [(\forall i, j, i \neq j, 1 \leq i, j \leq k) (c_i([0,1]^n) \neq c_j([0,1]^n))]. \quad (3.4)$$

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

Definition 3.7. An n -dimensional orthogonal pseudo-polytope p , or just an n D-OPp, is an n -chain composed by n D hyper-boxes arranged in such way that by selecting a vertex, in any of these hyper-boxes, it describes a combination of n D hyper-boxes (Definition 3.6) composed up to 2^n hyper-boxes.

3.2. n D-EVM's Fundamentals

Definition 3.8. Let c be a combination of hyper-boxes in the n -Dimensional space. An *Odd Adjacency Edge* of c , or just an *Odd Edge*, is 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 is called *Even Adjacency Edge*, or just *Even Edge*.

Definition 3.9. A *brink* or *extended edge* is the maximal uninterrupted segment, built out of a sequence of collinear and contiguous odd edges of an n D-OPP.

From the above definition, every even edge of an n D-OPP does not belong to brinks. On the other hand, every brink consists of m odd 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.

Definition 3.10. The ending vertices of all the brinks in p will be called *Extreme Vertices* of an n D-OPP p . $EV(p)$ will denote to the set of Extreme Vertices of p .

Property 3.11. Any extreme vertex of an n D-OPP, $n \geq 1$, when is locally described by a set of surrounding n D hyper-boxes, has exactly n incident linearly independent odd edges.

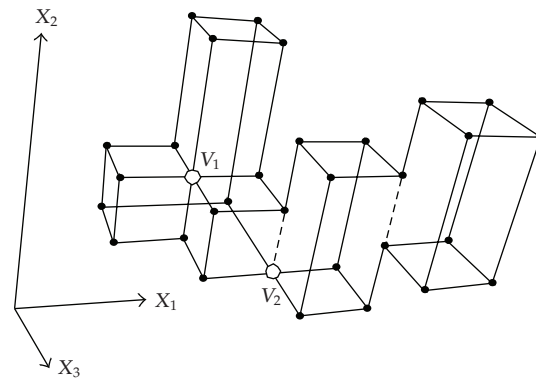


Figure 2: Example of a 3D-OPP and its set of Extreme Vertices (Continuous lines indicate odd edges, dotted lines indicate even edges, black points correspond to extreme vertices, and white points correspond to non extreme vertices).

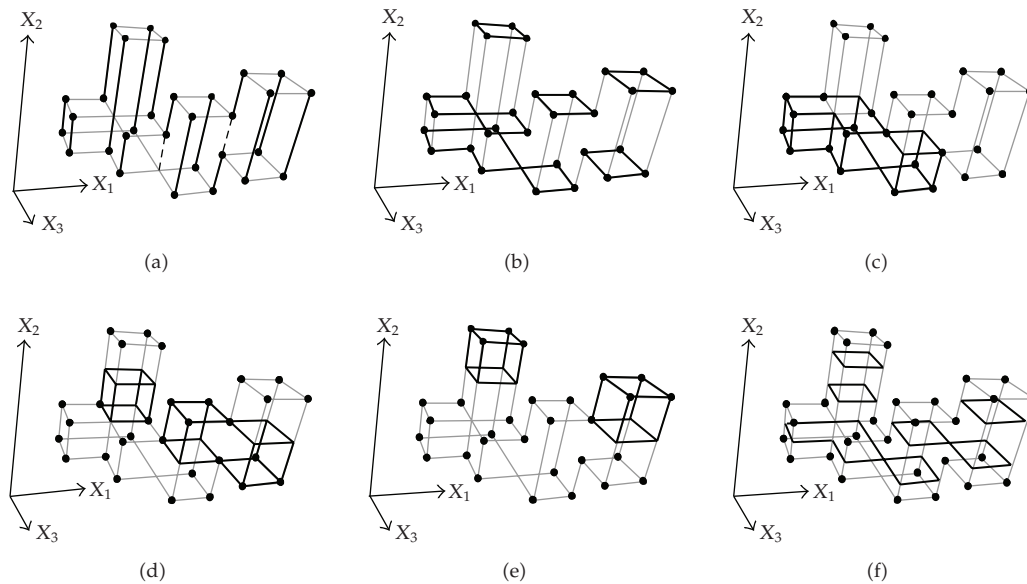


Figure 3: Illustrating some EVM concepts over a 3D-OPP (See text for details).

Figure 2 shows an example of a 3D-OPP p . Vertices v_1 and v_2 are nonextreme vertices because, in the case of v_1 , it has six incident odd edges, while vertex v_2 has four incident coplanar odd edges (see Property 3.11). 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 n D-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, $1 \leq i \leq n$, produce to the same set $EV(p)$. See in Figure 3(a) the brinks parallel to X_2 -axis for the 3D-OPP originally presented in Figure 2.

Definition 3.12. Let p be an n D-OPP. A k D extended hypervolume of p , $1 < k < n$, denoted by $\phi(p)$, is the maximal set of k D cells of p that lies in a k D space, such that a k D cell e_0 belongs to a k D extended hypervolume if and only if e_0 belongs to an $(n - 1)$ D cell present in $\partial(p)$ (see Definition 3.5), that is,

$$(e_0 \in \phi(p)) \iff (\exists c, c \text{ belongs to } \partial(p)) \left(e_0 \left([0, 1]^k \right) \subseteq c \left([0, 1]^{n-1} \right) \right). \quad (3.5)$$

Definition 3.13. Let p be an n D-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 .

Let Q be a finite set of points in \mathbb{R}^3 . In [10] 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. Now, let p be a 3D-OPP. According to [10] the Extreme Vertices Model of p , $EVM_3(p)$, denotes the ABC sorted set of the extreme vertices of p . Then $EVM_3(p) = EV(p)$ except by the fact that coordinates of points in $EV(p)$ are not necessarily sorted. In general, it is always assumed that coordinates of extreme vertices in the Extreme Vertices Model of an n D-OPP p , $EVM_n(p)$, have a fixed coordinates ordering. Moreover, when an operation requires manipulating two EVMs, it is assumed that both sets have the same coordinates ordering.

Definition 3.14. The *Projection Operator* for $(n - 1)$ D cells, points, and set of points is, respectively, defined as follows.

- (i) Let $c(I_{(i,\alpha)}^n(x)) = (x_1, \dots, x_n)$ be an $(n - 1)$ D cell embedded in the n D space. Let $\pi_j(c(I_{(i,\alpha)}^n(x)))$ denote the projection of the cell $c(I_{(i,\alpha)}^n(x))$ onto an $(n - 1)$ D space embedded in n D space whose supporting hyperplane is perpendicular to X_j -axis

$$\pi_j \left(c \left(I_{(i,\alpha)}^n(x) \right) \right) = (x_1, \dots, \hat{x}_j, \dots, x_n). \quad (3.6)$$

- (ii) 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:

$$\pi_j(v) = (x_1, \dots, \hat{x}_j, \dots, x_n). \quad (3.7)$$

- (iii) Let Q be a set of points in \mathbb{R}^n . The projection of the points in Q , denoted by $\pi_j(Q)$, is defined as the set of points in \mathbb{R}^{n-1} such that

$$\pi_j(Q) = \left\{ p \in \mathbb{R}^{n-1} : p = \pi_j(x), x \in Q \subset \mathbb{R}^n \right\}, \quad (3.8)$$

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

Definition 3.15. Consider an n D-OPP p as follows.

- (i) Let np_i be the number of distinct coordinates present in the vertices of p along X_i -axis, $1 \leq i \leq n$.
- (ii) 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 Figure 3(b) the set of 2D-couplets perpendicular to X_2 -axis for the 3D-OPP presented in Figure 2.

Definition 3.16. A slice is the region contained in an n D-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 3.17. A section is the $(n - 1)$ D-OPP, $n > 1$, resulting from the intersection between an n D-OPP p and an $(n - 1)$ D hyperplane perpendicular to the coordinate axis X_i , $n \geq i \geq 1$, which does 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.

Property 3.18. Let p be an n D-OPP. All the $(n - 1)$ D hyperplanes perpendicular to X_i -axis, $1 \leq i \leq n$, which intersect to $\text{Slice}_k^i(p)$ give the same section $S_k^i(p)$.

The Figures 3(c), 3(d), and 3(e) show the slices for a 3D-OPP (originally presented in Figure 2) according to the supporting planes of its 2D-couplets perpendicular to X_2 -axis. Figure 3(f) presents its corresponding set of internal sections perpendicular to X_2 -axis.

3.2.1. A Note about Regularized Boolean Operations

Independently of the scheme we consider for the representation of n D polytopes, it should be feasible to combine them in order to compose new objects [14]. One of the most common methods to combine polytopes is the set theoretical Boolean operations, as the Union, Difference, Intersection, and Exclusive OR. However, the application of an ordinary set theoretical Boolean operation on two polytopes does not necessarily produce a polytope. For example, the ordinary intersection between two cubes with only a common vertex is a point. Instead of using ordinary set theoretical Boolean operators, the *Regularized Boolean Operators* [15, 16] will be used. The practical purpose of regularization of polytope models is to make them dimensionally homogeneous [17]. The regularization operation can be defined as $\text{Regularized}(S) = \text{Closure}(\text{Interior}(S))$ which results in a closed regular set. Each regularized Boolean operator is defined in function of an ordinary operator in the following way:

$$A \text{ op}^* B = \text{Closure}(\text{Interior}(A \text{ op} B)). \quad (3.9)$$

These operators are defined as the closure of the interior of the corresponding set theoretical Boolean operation [16, 18]. In this way, the regularized operations between polytopes always will generate polytopes or a null object (the empty set) [17].

3.2.2. Computing Couplets from Sections

Theorem 3.19. The projection of the set of $(n - 1)$ D couplets, $\pi_i(\Phi_k^i(p))$, $1 \leq i \leq n$, of an n D-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, that is, $\pi_i(\Phi_k^i(p)) = \pi_i(S_{k-1}^i(p)) \otimes * \pi_i(S_k^i(p))$, for all $k \in [1, np_i]$.

3.2.3. Computing Sections from Couplets

Theorem 3.20. *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 $\pi_i(\Phi_k^i(p))$:*

$$\begin{aligned} S_0^i(p) &= \emptyset, \\ \pi_i(S_k^i(p)) &= \pi_i(S_{k-1}^i(p)) \otimes * \pi_i(\Phi_k^i(p)), \quad \forall k \in [1, np_i]. \end{aligned} \tag{3.10}$$

3.2.4. The Regularized XOR Operation on the nD -EVM

Theorem 3.21. *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 formulae for computing nD -OPPs sections from couplets and viceversa by means of their corresponding Extreme Vertices Models. They are obtained by combining Theorem 3.21 with Theorem 3.19 and Theorem 3.21 with Theorem 3.20, respectively.

Corollary 3.22. *One has that $EVM_{n-1}(\pi_i(\Phi_k^i(p))) = EVM_{n-1}(\pi_i(S_{k-1}^i(p))) \otimes EVM_{n-1}(\pi_i(S_k^i(p)))$.*

Corollary 3.23. *One has that $EVM_{n-1}(\pi_i(S_k^i(p))) = EVM_{n-1}(\pi_i(S_{k-1}^i(p))) \otimes EVM_{n-1}(\pi_i(\Phi_k^i(p)))$.*

3.2.5. The Regularized Boolean Operations on the nD -EVM

Corollary 3.24. *Let p and q be two nD -OPPs and $r = p \text{ op}^* q$, where op^* is in $\{\cup^*, \cap^*, -, \otimes^*\}$. Then $\pi_i(S_k^i(r)) = \pi_i(S_k^i(p)) \text{ op}^* \pi_i(S_k^i(q))$. Moreover, if all these sections lie in the same $(n-1)D$ hyperplane then $S_k^i(r) = S_k^i(p) \text{ op}^* S_k^i(q)$.*

Now we present the following.

Theorem 3.25. *A regularized Boolean operation, op^* , where $\text{op}^* \in \{\cup^*, \cap^*, -, \otimes^*\}$, over two nD -OPPs p and q , both expressed in the nD -EVM, can be carried out by means of the same op^* applied over their own sections, expressed through their Extreme Vertices Models, which are $(n-1)D$ -OPPs.*

This result leads into a recursive process for computing the Regularized Boolean operations using the nD -EVM, which descends on the number of dimensions [10]. The base or trivial case of the recursion corresponds to the $1D$ -Boolean operations which can be performed using direct methods. In Section 3.4.2 will be described an algorithm for performing Regularized Boolean operations under the nD -EVM which implements the above results.

3.3. Basic Algorithms for the nD -EVM

Now, we introduce some primitive operations which are in fact based on those originally presented in [10]. When an operation requires manipulating two EVMs, it is assumed that both sets have the same coordinates ordering. In the following, X_A -axis refers to the nD space's coordinate axis associated to the first coordinate present in the vertices of $EVM_n(p)$. For example, given coordinates ordering $X_1X_2X_3$, for a 3D-OPP, then $X_A = X_1$.

- (i) $InitEVM()$: returns the empty set.
- (ii) $PutHv1(EVM\ hv1, EVM\ p)$: appends and $(n-1)D$ couplet hvl , embedded in nD space and perpendicular to X_A -axis, to the nD -EVM p .
- (iii) $ReadHv1(EVM\ p)$: extracts and returns the next $(n-1)D$ couplet perpendicular to X_A -axis from p .
- (iv) $EndEVM(EVM\ p)$: returns true if the end of p along X_A -axis has been reached, otherwise, returns false.
- (v) $SetCoord(EVM\ hv1, CoordType\ coord)$: sets the X_A -coordinate to $coord$ on every vertex of the $(n-1)D$ couplet hvl . For $coord = 0$, it performs the projection $\pi_A(p)$. $CoordType$ is the chosen type for the vertex coordinates.
- (vi) $GetCoord(EVM\ hv1)$: returns the common X_A -coordinate of the $(n-1)D$ couplet hvl .
- (vii) $GetCoordNextHv1(EVM\ p)$: returns the common X_A -coordinate of the next available $(n-1)D$ couplet, perpendicular to X_A -axis, from p .
- (viii) $MergeXor(EVM\ p, EVM\ q)$: applies the Exclusive OR operation to the vertices of p and q and returns the resulting EVM.

Since the EVM is a sorted model, $MergeXor$ function consists in a simple merging-like algorithm, and therefore, it runs on linear time [10]. Its complexity is given by $O(\text{Card}(EVM_n(p)) + \text{Card}(EVM_n(q)))$ because each vertex from $EVM_n(p)$ and $EVM_n(q)$ needs to be processed just once. Moreover, according to Theorem 3.21, the resulting set corresponds to the regularized XOR operation between p and q .

From the above primitive operations, and specifically $MergeXor$, the following pair of algorithms can be easily derived ([10, 11]).

- (i) $GetHv1(EVM\ S_i, EVM\ S_j)$: implements Corollary 3.22, and returns the projection of the couplet between consecutive sections S_i and S_j .
- (ii) $GetSection(EVM\ S, EVM\ hv1)$: implements Corollary 3.23, and returns the projection of the next section of an nD -OPP whose previous section is S .

the Algorithm 1 computes the sequence of sections of an nD -OPP p from its nD -EVM using the previous functions [10, 11]. 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).

```

Input: An nD-EVM p.
Procedure EVM_to.SectionSequence(EVM p)
    EVM hv1 // Current couplet.
    EVM Si, Sj // Previous and next sections about hv1.
    hv1 = InitEVM( )
    Si = InitEVM( )
    Sj = InitEVM( )
    hv1 = ReadHv1(p)
    while(Not(EndEVM(p)))
        Sj = GetSection(Si, hv1)
        Process(Si, Sj)
        Si = Sj
        hv1 = ReadHv1(p) // Read next couplet.
    end-of-while
end-of-procedure

```

Algorithm 1: Computing the sequence of sections from an nD-OPP p.

3.4. Interrogating and Manipulating nD-OPPs Represented via the nD-EVM

3.4.1. Computing the Content of an nD-OPP

An nD hyperprism can be 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 [19]. Consider an nD hyperprism P_n whose base is an $(n - 1)$ D polytope P_{n-1} of content C_{n-1} . If h_n is the distance between its bases, that is, the height of the hyperprism, then its content is given by [19]

$$\text{Content}(P_n) = \text{Content}(P_{n-1}) \cdot h_n = C_{n-1} \cdot h_n. \quad (3.11)$$

If it is the case where P_{n-1} is an $(n - 1)$ D hyperprism with height h_{n-1} generated by the parallel motion of an $(n - 2)$ D polytope P_{n-2} , then C_{n-1} is given by the expression $C_{n-1} = C_{n-2} \cdot h_{n-1}$, where C_{n-2} is the content of P_{n-2} . This last expression yields to rewrite the above equation as

$$\text{Content}(P_n) = (\text{Content}(P_{n-2}) \cdot h_{n-1}) \cdot h_n = (C_{n-2} \cdot h_{n-1}) \cdot h_n. \quad (3.12)$$

By considering that each $(n-k)$ D hyperprism P_{n-k} is generated by the parallel motion of an $(n-k-1)$ D hyperprism P_{n-k-1} , where $k = 0, 1, 2, \dots, n - 1$, then we have that the content of P_n can be computed according to the formula

$$\text{Content}(P_n) = \begin{cases} h_1, & n = 1, \\ \text{Content}(P_{n-1}) \cdot h_n, & n > 1, \end{cases} \quad (3.13)$$

where h_n is the height of hyperprism P_n . In the base case, where $n = 1$, the content of a segment is given directly by its "height", that is, the distance between its two boundary points.

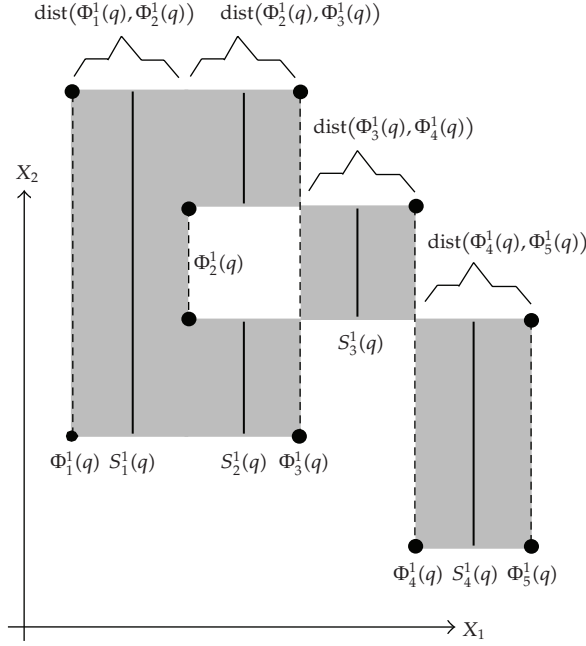


Figure 4: A 2D-OPP q whose area is being computed (see text for details).

The previous idea is extended in order to compute the content of nD space enclosed by an nD -OPP. In this case we will consider the partition induced by its slices (Definition 3.16). 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 [10] 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 $\text{Slice}_k^i(p)$, is given by the product between the area of its respective section $S_k^i(p)$ (the 2D base of $\text{Slice}_k^i(p)$) and the distance between $\Phi_k^i(p)$ and $\Phi_{k+1}^i(p)$ (the height of the 3D prism $\text{Slice}_k^i(p)$). Now let $q = S_k^i(p)$. The area of the 2D-OPP q (see Figure 4 for an example) can be computed as the sum of the areas of its 2D slices, where the area of a $\text{Slice}_k^i(q)$, is given by the product between the length of its respective section $S_k^i(q)$ (the 1D base of $\text{Slice}_k^i(q)$) and the distance between $\Phi_k^i(q)$ and $\Phi_{k+1}^i(q)$ (the height of the "2D prism" $\text{Slice}_k^i(q)$). Finally let $r = S_k^i(q)$. In the basic case the length of the 1D-OPP r is computed as the sum of the lengths of its brinks.

Let p be an nD -OPP. The nD space enclosed by p , denoted by $\text{Content}_{(n)}(p)$, can be computed as the sum of the contents of its nD slices [11]

$$\text{Content}_{(n)}(p) = \begin{cases} \text{Length}(p) & n = 1, \\ \sum_{k=1}^{np_i-1} \text{Content}_{(n-1)}(S_k^i(p)) \cdot \text{dist}(\Phi_k^i(p), \Phi_{k+1}^i(p)) & n > 1. \end{cases} \quad (3.14)$$

Algorithm 2 implements the above equation in order to compute the content of nD space enclosed by an nD -OPP p expressed through the nD -EVM [10, 11].

```

Input: An nD-EVM p.
          The number n of dimensions.
Output: The content of nD space enclosed by p.
Procedure Content(EVM p, int n)
  real cont = 0.0 // The content of nD space enclosed by p.
  EVM hv1, hv2 // Consecutive couplets perpendicular to XA-axis.
  EVM s // Current section of p.
  if(n = 1) then return Length(p) // Base case: p is a 1D-OPP.
  else
    s = InitEVM( )
    hv1 = ReadHvl(p)
    while(Not(EndEVM(p)))
      hv2 = ReadHvl(p)
      s = GetSection(s, hv1)
      cont = cont + Content(s, n-1) * dist(hv1, hv2)
      hv1 = hv2
    end-of-while
  return cont
end-of-else
end-of-procedure

```

Algorithm 2: Computing the content of nD space enclosed by an OPP.

3.4.2. The Boolean Operations Algorithm for the nD-EVM

Let p and q be two nD-OPPs represented through the nD-EVM, and let op^* be a regularized Boolean operator in $\{\cup^*, \cap^*, -^*, \otimes^*\}$. The algorithm, originally presented in [10] and based on Theorem 3.25, computes the resulting nD-OPP $r = p \text{op}^* q$. Note that $r = p \otimes^* q$ can also be trivially performed using Theorem 3.21 [10]. The idea behind the algorithm is the following.

- (i) The sequence of sections from p and q , perpendicular to X_A -axis, is obtained first, based on Theorem 3.20.
- (ii) Then, according to Corollary 3.24, every section of r can recursively be computed as

$$S_k^A(r) = S_k^A(p) \text{op}^* S_k^A(q). \quad (3.15)$$

- (iii) Finally, r can be obtained from its sequence of sections, perpendicular to X_A -axis, according to Theorem 3.19.

Nevertheless, Algorithm 3 does not work in this sequential form. It actually works in a wholly merged form in which it only needs to store one section for each of the operands p and q and two consecutive sections for the result r .

The following are some functions present in Algorithm 3 but not defined previously.

- (i) Function *BooleanOperation1D* performs 1D Boolean operations between p and q that are two 1D-OPPs.
- (ii) Procedure *NextObject* considers both input objects p and q and returns the common *coord* value of the next *hvl* to process, using function *GetCoord*. It also returns two

flags, *fromP* and *fromQ*, which signal from which of the operands (both inclusive) is the next *hvl* to come.

- (iii) The main loop of procedure *BooleanOperation* gets couplets from *p* and/or *q*, using function *GetSection*. These sections are recursively processed to compute, according to Corollary 3.24, the corresponding section of *r*, *sRcurr*. Since two consecutive sections, *sRprev* and *sRcurr*, are kept, then the projection of the resulting *hvl*, is obtained by means of function *GetHvl*, and then, it is correctly positioned by procedure *SetCoord*.

When the end of one of the polytopes *p* or *q* is reached then the main iteration finishes, and the remaining couplets of the other polytope are either appended or not to the resulting polytope depending on the Boolean operation considered. Procedure *PutBool* performs this appending process.

3.5. Hypervoxelizations to *nD*-EVM Conversion

First, consider the following.

Corollary 3.26 (see [10, 11]). *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 $EVM_n(p \cup q) = EVM_n(p) \otimes EVM_n(q)$.*

As previously commented in the introduction of this paper, a hypervoxelization is a set of quasi-disjoint black and white hypervoxels each one being a convex orthogonal polytope. Since the set of black hypervoxels represents an *nD*-OPP *p* we have [10]

$$p = \bigcup_{\lambda} \text{BlackHypervoxel}_{\lambda}. \quad (3.16)$$

For each black hypervoxel it should be possible to generate a list of its 2^n vertices. Because all of these 2^n vertices are Extreme Vertices, and by considering Corollary 3.26, all we have to do is [10]

$$EVM_n(p) = EVM_n\left(\bigcup_{\lambda} \text{BlackHypervoxel}_{\lambda}\right) = \otimes_{\lambda} EVM_n(\text{BlackHypervoxel}_{\lambda}). \quad (3.17)$$

This provides a straight and simple method for converting a hypervoxelization to the *nD*-EVM.

4. Computing Discrete Compactness for an *nD*-OPP Expressed in the *nD*-EVM

In the following it is assumed an *nD*-OPP fits exactly in a hypervoxelization, where hypervoxels are unit *nD* hypercubes whose vertices have integer coordinates. Given a polytope *p* expressed in the *nD*-EVM, and in order to compute $L_C(p)$, that is, the number of shared $(n - 1)D$ cells between the *m* hypervoxels that originally described *p*, we start by

```

Input: The nD-OPPs p and q expressed in the nD-EVM.
          The number n of dimensions and regularized Boolean operation op.
Output: The output nD-OPP r, such that  $r = p \text{ op}^* q$ , codified as an nD-EVM.
Procedure BooleanOperation(EVM p, EVM q, BooleanOperator op, int n)
    EVM sP, sQ // Current sections of p and q respectively.
    EVM hvl // I/O couplet.
    boolean fromP, fromQ // flags for the source of the couplet hvl.
    CoordType coord // the common coordinate of couplets.
    EVM r, sRprev, sRcurr // nD-OPP r and two of its sections.
    if(n = 1) then // Base case
        return BooleanOperation1D(p, q, op)
    else
        n = n - 1
        sP = InitEVM( )
        sQ = InitEVM( )
        sRcurr = InitEVM( )
        NextObject(p, q, coord, fromP, fromQ)
        while(Not(EndEVM(p)) and Not(EndEVM(q)))
            if(fromP = true) then
                hvl = ReadHvl(p)
                sP = GetSection(sP, hvl)
            end-of-if
            if(fromQ = true) then
                hvl = ReadHvl(q)
                sQ = GetSection(sQ, hvl)
            end-of-if
            sRprev = sRcurr
            sRcurr = BooleanOperation(sP,sQ,op,n) // Recursive call
            hvl = GetHvl(sRprev, sRcurr)
            SetCoord(hvl, coord)
            PutHvl(hvl, r)
            NextObject(p, q, coord, fromP, fromQ)
        end-of-while
        while(Not(EndEVM(p)))
            hvl = ReadHvl(p)
            PutBool(hvl, r, op)
        end-of-while
        while(Not(EndEVM(q)))
            hvl = ReadHvl(q)
            PutBool(hvl, r, op)
        end-of-while
        return r
    end-of-if
end-of-procedure

```

Algorithm 3: Computing regularized Boolean operations on the EVM.

processing its $(n - 1)$ D sections perpendicular to X_A -axis. By considering a section, it can be determined the number of hypervoxels that originally described its corresponding slice. Furthermore, it is possible to obtain information about internal contacts, perpendicular to X_A -axis, between the hypervoxels that described such n D slice. Let S_j be the section associated to $\text{Slice}_j^A(p)$. Then, the following steps are performed.


```

Input:    An  $nD$ -EVM  $p$  and the number  $n$  of dimensions.
Output: The number of internal contacts, perpendicular to the  $X_A$ -axis, between
            the hypervoxels that originally composed  $p$ .
Procedure InternalContacts(EVM  $p$ , int  $n$ )
    EVM  $hvl$ // Current couplet of  $p$ .
    EVM  $S_i, S_j$ // Previous and next sections about  $hvl$ .
    EVM  $S_{int}$ // The result of intersecting the projections of  $S_i$  and  $S_j$ .
    int  $c_1$ // Common  $X_A$ -coordinate of couplet  $hvl$ .
    int  $c_2$ // Common  $X_A$ -coordinate of couplet next to  $hvl$ .
    int  $nCoords$ // Number of integer coordinates between  $c_1$  and  $c_2$ .
    int  $L = 0$ // Number of internal contacts ( $(n-1)D$  adjacencies).
     $S_i = \text{InitEVM}()$ 
     $c_1 = \text{GetCoordNextHvl}(p)$ 
     $hvl = \text{ReadHvl}(p)$ 
    while(Not(EndEVM( $p$ )))
         $S_j = \text{GetSection}(S_i, hvl)$ 
         $c_2 = \text{GetCoordNextHvl}(p)$ 
         $nCoords = c_2 - c_1 - 1$ 
         $L = L + nCoords * \text{Content}(S_j, n - 1)$ 
         $S_{int} = \text{BooleanOperation}(S_i, S_j, \text{Intersection}, n - 1)$ 
         $L = L + \text{Content}(S_{int}, n - 1)$ 
         $S_i = S_j$ 
         $c_1 = c_2$ 
         $hvl = \text{ReadHvl}(p)$ // Read next couplet.
    end-of-while
    return  $L$ 
end-of-procedure

```

Algorithm 4: Computing the number of $(n - 1)D$ adjacencies between the hypervoxels that originally defined an nD -OPP p .

- (i) Let c_1 and c_2 be the common X_A coordinates of previous and next couplets, perpendicular to X_A -axis, about section S_j . Let $nCoords = c_2 - c_1 - 1$ be the number of integer coordinates between c_1 and c_2 . In fact, $nCoords = \text{Card}(\{c_1 + 1, c_1 + 2, \dots, c_2 - 1\})$.
- (ii) Because of p 's assumed original representation, it is clear by computing the nD content of $\text{Slice}_j^A(p)$ that it is obtained the number of black nD hypervoxels that originally defined it. Now, let H be an $(n - 1)D$ hyperplane perpendicular to X_A -axis such that it intersects $\text{Slice}_j^A(p)$ and does not coincide with the supporting hyperplanes of the previous and next couplets about section S_j . Assuming the common X_A -coordinate of the points in H is in $\{c_1 + 1, c_1 + 2, \dots, c_2 - 1\}$, and by computing the $(n - 1)D$ content of the projection of section S_j , there is obtained the number of those hypervoxels which have an $(n - 1)D$ boundary cell embedded in hyperplane H .
- (iii) Finally, the number of internal contacts, perpendicular to X_A -axis, between the hypervoxels that originally defined $\text{Slice}_j^A(p)$, is given by the product of $nCoords$ (the number of integer coordinates between the previous and next couplets about S_j) and the $(n - 1)D$ content of S_j .

The above procedure determines the number of internal contacts, perpendicular to X_A -axis, between the hypervoxels that originally composed to $\text{Slice}_j^A(p)$. Now, we must determine the number of internal contacts ($(n-1)$ D adjacencies) between those hypervoxels that belong to $\text{Slice}_j^A(p)$ with hypervoxels in the next slice, that is, $\text{Slice}_{j+1}^A(p)$. Those shared cells are embedded in the supporting $(n-1)$ D hyperplane of the couplet between sections S_j and S_{j+1} .

- (i) The projections of consecutive sections S_j and S_{j+1} are intersected in such way that it is obtained an $(n-1)$ D-OPP S_{int} .
- (ii) The number of those $(n-1)$ D boundary cells, embedded in the supporting $(n-1)$ D hyperplane of the couplet between S_j and S_{j+1} , is given by computing the $(n-1)$ D content of S_{int} . If S_{int} is empty then it implies that (1) one of the sections, S_j or S_{j+1} , is empty or (2) projections of S_j and S_{j+1} are disjoint or quasi-disjoint. In both cases no $(n-1)$ D contact occurs between $\text{Slice}_j^A(p)$ and $\text{Slice}_{j+1}^A(p)$.

Algorithm 4 implements the above procedures.

We will clarify the way Algorithm 4 works with the following example. Consider the object defined by the 3D boxes presented in Figure 5(a). The 3D grid where the object is embedded has dimensions $8 \times 8 \times 8$. The coordinates of all the boxes' vertices are in the set $\{0, 1, \dots, 8\}^3$. When the boxes are united, according to the procedure described in Section 3.5, it is obtained the 3D-OPP p shown, as a wireframe model, in Figure 5(b). Its 3D-EVM is composed by 14 extreme vertices. Figure 5(c) shows p 's three couplets perpendicular to X_1 -axis, while Figure 5(d) presents its pair of internal sections, $S_1^1(p)$ and $S_2^1(p)$, also perpendicular to X_1 -axis. 2D section $S_1^1(p)$ is associated to $\text{Slice}_1^1(p)$ (see Figure 5(e)), while $S_2^1(p)$ is associated to $\text{Slice}_2^1(p)$ (Figure 5(f)). Suppose that 2D couplets $\Phi_1^1(p)$, $\Phi_2^1(p)$, and $\Phi_3^1(p)$ have common X_1 coordinates given by 0, 4, and 8, respectively. Hence, the number of integer coordinates, along X_1 -axis, between couplets $\Phi_1^1(p)$ and $\Phi_2^1(p)$ is 3. Such coordinates are 1, 2, and 3. Now, consider the projection of section $S_1^1(p)$ (Figure 5(g)). Because the boxes in the original voxelization were unitary, then the area of $S_1^1(p)$, $64 u^2$, corresponds to the number of faces embedded in a plane intersecting $\text{Slice}_1^1(p)$ which does not coincide with $\Phi_1^1(p)$ nor $\Phi_2^1(p)$, and whose common X_1 coordinate is in $\{1, 2, 3\}$. Therefore, the number of internal contacts (face adjacencies), between the boxes that originally composed to $\text{Slice}_1^1(p)$ is given by the product $3 \cdot 64 = 192$.

Now, we proceed to determine the number of internal contacts in $\text{Slice}_2^1(p)$. The number of integer coordinates, along X_1 -axis, between couplets $\Phi_2^1(p)$ and $\Phi_3^1(p)$ is 3. Such coordinates are 5, 6, and 7. Consider the projection of section $S_2^1(p)$ (Figure 5(h)). The area of $S_2^1(p)$ is $48 u^2$. It corresponds to the number of faces embedded in a plane intersecting $\text{Slice}_2^1(p)$ which does not coincide with $\Phi_2^1(p)$ nor $\Phi_3^1(p)$, and whose common X_1 coordinate is in $\{5, 6, 7\}$. The number of internal contacts (face adjacencies), between the boxes that originally composed to $\text{Slice}_2^1(p)$ is given by the product $3 \cdot 48 = 144$. Finally, we compute the regularized intersection between projections of sections $S_1^1(p)$ and $S_2^1(p)$, that is, $S_{\text{int}} = \pi_1(S_1^1(p)) \cap * \pi_1(S_2^1(p))$. See Figure 5(i). The area of 2D-OPP S_{int} is $48 u^2$. This number corresponds to the number of faces, in the original voxelization, embedded in the supporting plane of couplet $\Phi_2^1(p)$. Each one of these faces implies a face adjacency between a voxel that belongs to $\text{Slice}_1^1(p)$ and a voxel in $\text{Slice}_2^1(p)$. Finally, the number of face adjacencies, perpendicular to X_1 -axis, between the boxes in p 's original voxelization, is 384. Because of p 's symmetry, it can be easily verified that the number of face adjacencies perpendicular to X_2 -axis and X_3 -axis is also 384. Therefore, $L_C(p) = 384 + 384 + 384 = 1,152$.

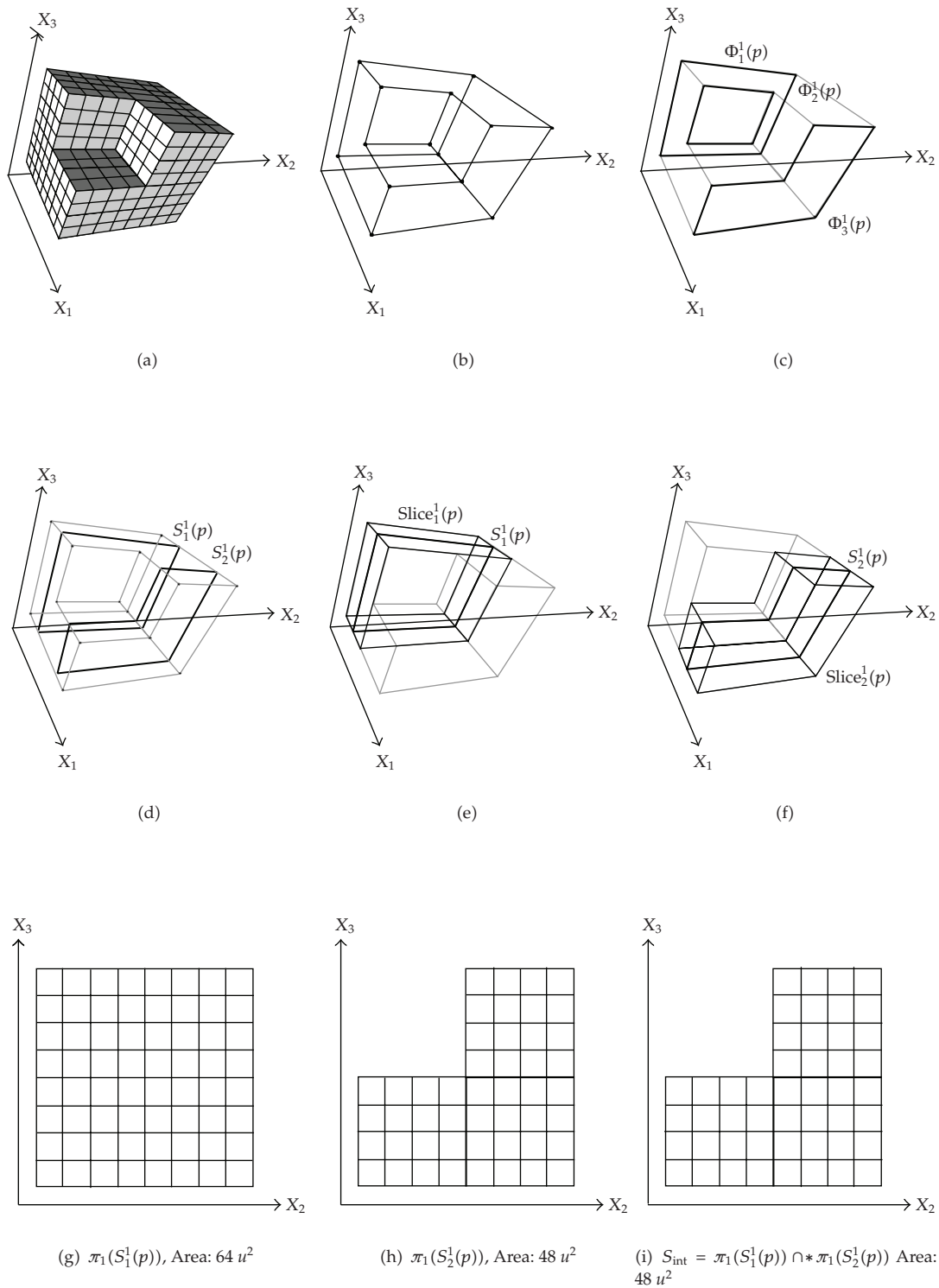


Figure 5: Computing the number of face adjacencies, perpendicular to X_1 -axis, between the voxels that originally defined a 3D-OPP now expressed with a 3D-EVM (see text for details).

```

Input:   An  $n$ D-EVM  $p$  and the number  $n$  of dimensions.
Output: The number of internal contacts, perpendicular to  $X_i$ -axis,
             $i=1,2,\dots,n$ , between the hypervoxels that originally composed  $p$ .
Procedure TotalInternalContacts(EVM  $p$ , int  $n$ )
    int  $L_c = 0$ 
    for each sorting in  $\{X_1X_2\dots X_{n-1}X_n, X_2X_3\dots X_nX_1, X_3X_4\dots X_1X_2, \dots, X_nX_1\dots X_{n-2}X_{n-1}\}$  do
        SortEVM( $p$ ,  $n$ , sorting)
         $L_c = L_c + \text{InternalContacts}(p, n)$  // Call to Algorithm 4.
    end-of-for
    return  $L_c$ 
end-of-procedure

```

Algorithm 5: Computing $L_C(p)$ for an n D-OPP expressed in the EVM.

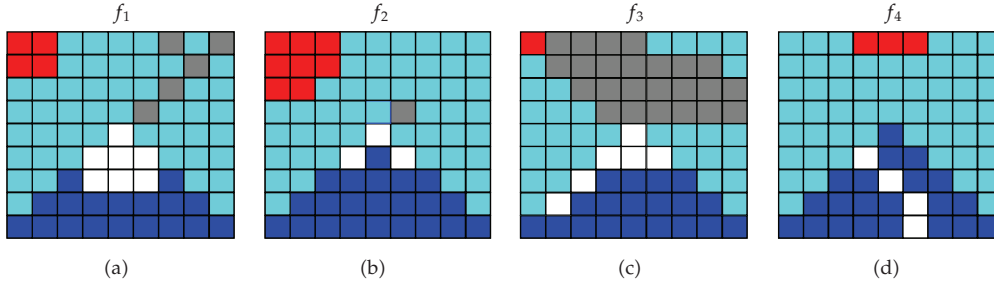


Figure 6: Example of a simple color 2D-animation.

As seen, Algorithm 4 extracts those sections of n D-OPP p which are perpendicular to X_A -axis which implies that it only counts internal contacts perpendicular to this same coordinate axis. Suppose that the coordinates ordering in $\text{EVM}_n(p)$ is given by $X_1X_2\dots X_{n-1}X_n$. Each one of the coordinates orderings in the set $\{X_1X_2\dots X_{n-1}X_n, X_2X_3\dots X_nX_1, X_3X_4\dots X_1X_2, \dots, X_nX_1\dots X_{n-2}X_{n-1}\}$ is now used for sorting $\text{EVM}_n(p)$. Such sorting is performed by calling a procedure *SortEVM*. Given a permutation $X_{\alpha_1}X_{\alpha_2}\dots X_{\alpha_n}$, the function sorts the extreme vertices of p first according to coordinate X_{α_1} , after according to coordinate X_{α_2} , and so on until p is sorted according to coordinate X_{α_n} . Next, it is performed a calling to Algorithm 4. By this way, it is computed the number of contacts perpendicular to each one of the axes in n D space. The sum of all contacts perpendicular to all coordinate axes leads to determine $L_C(p)$. Algorithm 5 implements this procedure.

5. Representing Color 2D-Animations through 4D-OPPs and the EVM

The procedure described in [10] for processing black and white 2D animations can be directly extended to control colored frames through a 4D-OPP represented through the EVM. This methodology was originally presented in [11, 12]. In Figure 6 an example of a simple color 2D animation composed by four frames whose resolution is 9×9 pixels is shown. In each frame can be identified white, red, blue, gray, and cyan regions. We will use this simple animation to exemplify our procedure. We will label each colored frame in the animation as f_k and m will be the number of such frames.

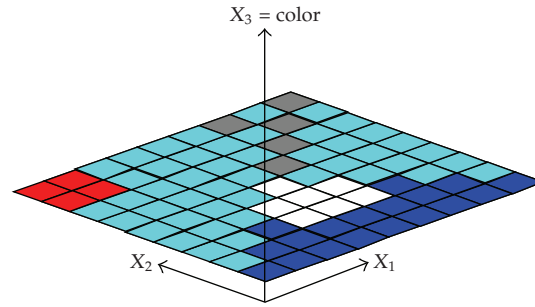


Figure 7: The 3D space defined for the extrusion of color 2D-pixels.

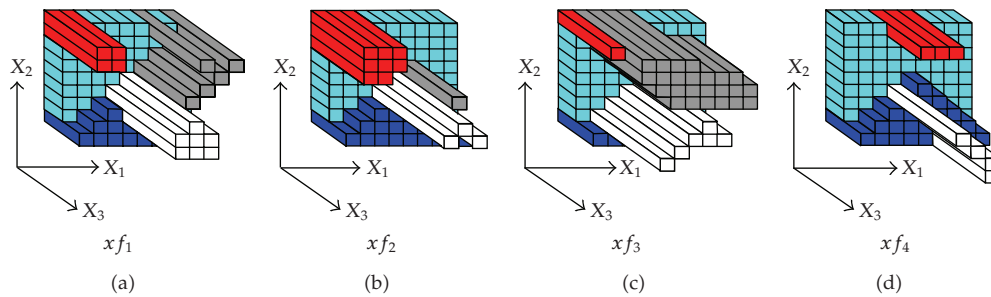


Figure 8: The sets of prisms resulting of the extrusion of the frames of an animation (presented in Figure 6).

A color animation can be handled as a 4D-OPP in the following way [11, 12].

(a) The Red-Green-Blue (RGB) components of each pixel will be integrated into a single value. Such value represents the RGB components as an integer with 32 bits. Bits 0–7 correspond to the blue value, bits 8–15 correspond to the green value, bits 16–23 correspond to the red value, and bits 24–31 to the *alpha* (transparency) value. Each pixel will now be extruded towards the third dimension, where the value integrating its RGB components will now be considered as its X_3 coordinate (coordinates X_1 and X_2 correspond to the original pixels' coordinates). See Figure 7.

Let us call xf_k to the set composed by the rectangular prisms (the extruded pixels) of each extruded frame f_k . It is very important to avoid the zero value in the X_3 coordinate because a pixel could not be extruded, and therefore its associated prism (a 3D-OPP) will not be obtained. See Figure 8.

(b) Let $prism_i$ be a prism in xf_k and npr the number of prisms in that set. Due to all the prisms in xf_k are quasi disjoint 3D-OPPs, we can easily obtain the final 3D-OPP and its respective 3D-EVM of the whole 3D frame via Corollary 3.26 (all the vertices in a $prism_i$ are extreme vertices)

$$EVM_3(F_k) = \otimes_{i=1}^{npr} EVM_3(prism_i \in xf_k), \quad (5.1)$$

where F_k is the 3D frame (a 3D-OPP) that represents the union of all the prisms in xf_k . See Figure 9.

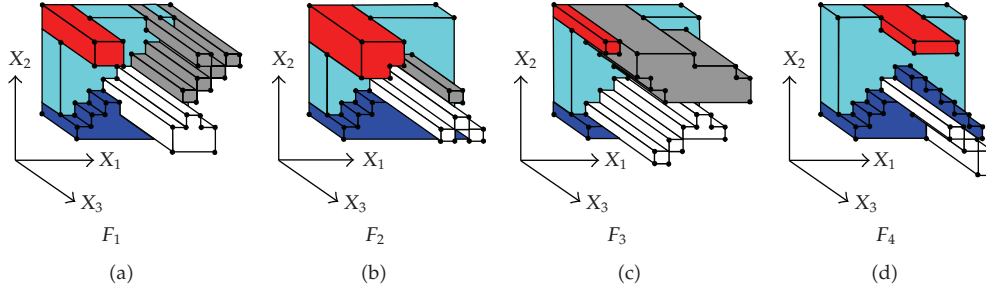


Figure 9: The 3D frames that represent a 2D colored animation (presented in Figure 6; Some of their extreme vertices are shown).

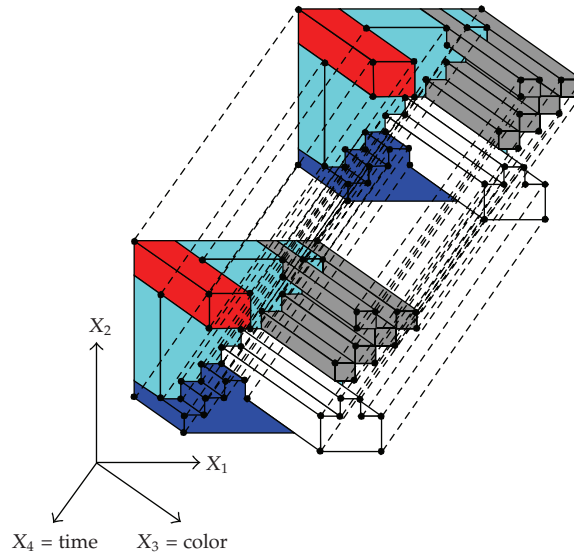


Figure 10: The process of extrusion of a 3D frame in order to obtain a *hyperprism* (some of its extreme vertices are shown).

(c) Let us extrude F_k into the fourth dimension and thus obtain a 4D hyperprism $_k$ whose bases are F_k and its length is proportional to the time f_k is to be displayed. The new fourth dimension will measure and represent the time. See Figure 10.

(d) Let p be the 4D-OPP that represents the given color 2D-animation. Polytope p is defined by

$$p = \bigcup_{k=1}^m \text{hyperprism}_k. \quad (5.2)$$

Due to all the m hyperprisms are quasi disjoint 4D-OPPs, then the 4D-EVM for p can be obtained by

$$\text{EVM}_4(p) = \bigotimes_{k=1}^m \text{EVM}_4(\text{hyperprism}_k). \quad (5.3)$$

By representing a given color 2D-animation using a 4D-OPP p and its 4D-EVM, we have the following characteristics [11, 12].

- (i) The sequence of the projections of sections in p corresponds to the sequence of 3D frames, that is, $\pi_4(S_k^4(p)) = F_k$.
- (ii) Computation of 3D frames: by Corollary 3.23 the 3D-EVM of the frame F_k is computed by $EVM_3(F_k) = EVM_3(F_{k-1}) \otimes EVM_3(\pi_4(\Phi_k^4(p)))$.
- (iii) Displaying the 2D colored animation: each couplet perpendicular to the X_3 axis in each 3D frame F_k contains the polygons to display. The colors to apply to those polygons are referred through the X_3 coordinate that contains the integrated RGB components.

6. Discrete Compactness of 4D-OPPs: An Application






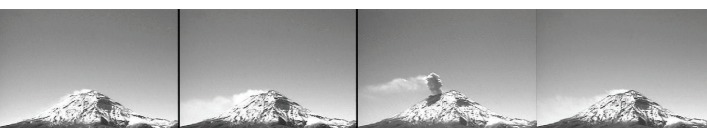
In this section there will be described our first steps towards an application of Discrete Compactness in a higher dimensional context, via the algorithms presented in Section 4, in the classification and indexing of video sequences. In the previous section, it was presented a methodology for representing color 2D animations through 4D-OPPs and the EVM. In the case to be boarded now, we will consider image sequences associated to the Popocatepetl volcano which is located in the Mexican State of Puebla. The images were obtained from the CENAPRED web site [20]. The CENAPRED is a Mexican research center which has as one of its functions monitoring the volcano's activity.

Each sequence presents the volcano's activity along four consecutive days. The sequences are then composed by 4 images, or frames, each one taken each day. All the images were captured in the time range comprehended from 10 AM to 2 PM. We have a set of 521 images which correspond to dates between January 1, 2007 and June 4, 2008. They have resolution 640×480 under the color model RGB. In a preprocessing phase, images were scaled to the new resolution 320×240 , and the color model was changed to grayscale. Moreover, a multilevel threshold was applied in order to eliminate noise (specifically we applied some procedures defined in [21]). There were generated 520 sequences to which was applied the procedure described in previous section in such way that for each one was generated its corresponding 4D polytope: two geometrical dimensions (X_1 and X_2), one color dimension (X_3), and one dimension associated to time (X_4). Consequently, the obtained 4D polytopes were expressed in the EVM.

For each 4D polytope it was computed its Discrete Compactness. It is clear each generated polytope is embedded in a hyper-box with main diagonal defined by the pair of points $(0,0,0,0)$ and $(321,241,256,4)$. Such hyper-box will be called H_{\max} , and it corresponds to a sequence composed by 4 white frames. On the other hand, let H_{\min} be the hyper-box described by the main diagonal with start and end points given, respectively, by $(0,0,0,0)$ and $(321,241,1,4)$. It corresponds to a sequence where all the frames are black. The required values $L_{C_{\max}}$ and $L_{C_{\min}}$ are computed through Algorithm 5 by using as input the EVMs associated to hyper-boxes H_{\max} and H_{\min} , respectively. That is, $L_{C_{\max}} = L_C(H_{\max})$ and $L_{C_{\min}} = L_C(H_{\min})$.

The 520 sequences were sorted according to the discrete compactness of their corresponding 4D polytope. Compactness' maximum obtained value was 0.8367 while the minimum was given by 0.7236. Sequences were grouped in 12 classes (see Tables 1 and 2). Each class contains sequences whose discrete compactness is inside an arbitrarily defined range. The classes 1 to 4 group sequences with compactness major or equal to 0.72 and more minor

Table 1: Using discrete compactness for classifying 4D-OPPs associated to images sequences (Part 1).

Class	Discrete compactness range	Representative sequence (arbitrarily chosen)		
1	[0.72, 0.73)	 Members 1	Average card($EVM_4(p)$) 382, 148	Average processing time (milliseconds) 91, 167
2	[0.73, 0.74)	 Members 3	Average card($EVM_4(p)$) 360, 241.33	Average processing time (milliseconds) 85, 238.67
3	[0.74, 0.75)	 Members 1	Average card($EVM_4(p)$) 388, 512	Average processing time (milliseconds) 93, 616
4	[0.75, 0.76)	 Members 16	Average card($EVM_4(p)$) 375, 392.13	Average processing time (milliseconds) 91, 033.27
5	[0.76, 0.77)	 Members 31	Average card($EVM_4(p)$) 370, 449.46	Average processing time (milliseconds) 91, 159.7
6	[0.77, 0.78)	 Members 70	Average card($EVM_4(p)$) 365, 405.73	Average processing time (milliseconds) 90, 480.33

than 0.76. These classes have 1, 3, 1, and 16 members respectively and they have as an interesting characteristic the fact that some of their members are sequences that describe intense volcanic activity. See for example the representative sequences for classes 1, 2, and 3 (Table 1). They describe the same event: an eruption which took place in December 1, 2007 [20].

Classes 5 to 8 group sequences with discrete compactness inside the range [0.76, 0.80]. The majority of these sequences describe volcanic activity that goes from null to moderate. The classes 9 to 12 (Table 2) have as members sequences whose 4D polytopes have Discrete Compactness in the range [0.80, 0.84). In this case, a great part of these sequences have the property that they correspond to days where the visibility towards the volcano was minimized by clouds or even was null. All the sequences in classes 9, 10, and 11 have cloudy days with low visibility. Moreover, there are also sequences in which the number of days with null visibility oscillated between 1 and 3. Class 12 is the only one where there are sequences such that all four days had null visibility.

The above results are promising in the sense that discrete compactness could be applied as a mechanism for indexing/classification/comparison of n -Dimensional Polytopes. In our example, we have obtained preliminary results that allow classifying 4D objects which in time describe image sequences. We implemented the algorithms and procedures described in this paper in the Java Language (Java Development Kit version 1.6). They were executed in a computer with an Intel Processor Core 2 Duo, 2.40 GHz, and 2 Gigabytes in RAM. In Tables 1 and 2 can be appreciated, for each class, the average processing time required for computing discrete compactness for our 4D polytopes. These time measures provide experimental evidence of efficiency in Discrete Compactness' computation via the EVM. Another provided data are the referent to the average number of extreme vertices required for representing sequences. A 4D polytope that corresponds to the representation of a sequence of images could be also represented and manipulated via a 4D hypervoxelization. As described previously, all our polytopes are embedded in a 4D hyper-box with main diagonals defined by points (0,0,0,0) and (321,241,256,4). In fact, the required 4D grid for representing our polytopes through a hypervoxelization should be at least of size $320 \times 240 \times 256 \times 4$. This implies to store 78,643,200 4D hypervoxels. The average number of extreme vertices, required for representing the sequences contained in class 3, is 388,512. In fact, this is the maximum average from the 12 proposed classes. The ratio


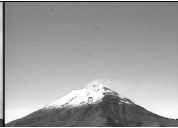




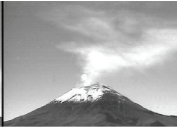

















$$\frac{78,643,200}{388,512} = 202.4215 \quad (6.1)$$

clearly shows how the EVM is effectively a suitable representation for these sequences because its memory requirements are much less when compared with a hypervoxelization scheme.

7. Concluding Remarks

Nowadays it is common to find applications where the search of results and properties is more suitable to be performed in hyperspaces. On one side, this implies that the representations to use must be powerful enough in the sense that conciseness, efficiency, and robustness are mandatory characteristics to be taken in account. On the other side, we consider that the notion of Discrete Compactness in higher dimensional contexts plays a fundamental role in tasks related to classification, description, and indexing of nD

Table 2: Using Discrete Compactness for classifying 4D-OPPs associated to images sequences (Part 2).

Class	Discrete compactness range	Representative sequence (arbitrarily chosen)			
7	[0.78, 0.79)				
		Members 112	Average card($EVM_4(p)$) 314, 840.79	Average processing time (milliseconds) 74, 513.05	
8	[0.79, 0.8)				
		Members 121	Average card($EVM_4(p)$) 292, 567.66	Average processing time (milliseconds) 68, 287.25	
9	[0.8, 0.81)				
		Members 95	Average card($EVM_4(p)$) 273, 454.38	Average processing time (milliseconds) 62, 492.4	
10	[0.81, 0.82)				
		Members 54	Average card($EVM_4(p)$) 237, 095.92	Average processing time (milliseconds) 52, 931.11	
11	[0.82, 0.83)				
		Members 15	Average card($EVM_4(p)$) 186, 286.28	Average processing time (milliseconds) 40, 392.71	
12	[0.83, 0.84)				
		Members 8	Average card($EVM_4(p)$) 69, 550.57	Average processing time (milliseconds) 13, 005.71	

polytopes. In such sense, this paper has been devoted to present a set of methodologies that allow the efficient computation of Discrete Compactness of those n D-OPPs whose original representation was based on a hypervoxelization. In Section 3 we described the Extreme Vertices Model in the n -Dimensional space which represents concisely n D-OPPs. It was described a straight method for conversion of hypervoxelizations to the n D-EVM. By expressing a polytope via the EVM, we have, on one hand, the important advantage related to the fact that EVM's storing requirements are much less than those required by a hypervoxelization model. On the other hand, it is available a set of EVM-based algorithms that allow the manipulation and querying of polytopes represented under our model. As commented in the previous sections, the EVM of a polytope is in fact a subset of its vertices. However, the algorithms presented in Section 3 obtain, from such subset of vertices, useful topological and geometrical information. Supported in those algorithms, we have proposed, specifically in Section 4, methodologies for performing the computation of Discrete Compactness. We have extended, in direct way, concepts presented by Bribiesca in [1, 5], in order to apply them on n D-OPPs expressed under the EVM.

In Section 6 we presented how by using our EVM-based implementation of Discrete Compactness it was possible to classify in efficient way images sequences associated to Popocatepetl volcano. They were grouped according to the value of their corresponding compactness. Informally, we saw how sequences with visual common properties have near compactness values. On the other hand, currently we are working in procedures for extracting the number of k D elements on the boundary, $k = 0, 1, 2, \dots, n - 1$, of an n D-OPP expressed in the EVM. Such counts by themselves are geometrical factors describing the polytope, but they can also be useful for determining, for example, topological factors such as the well-known Euler characteristic. The first steps presented in this paper, the obtained results and observations, and the new methodologies we are defining for extracting another geometrical and topological factors, encourage us to concentrate our efforts towards the following main idea. we want to determine how factors, such as Discrete Compactness, Euler characteristic, boundary elements counts, and so forth, that describe a higher dimensional polytope representing a sequence, can be used for a specification for indexing and classification of video sequences. Moreover, we will investigate if such factors, that provide us descriptive information about a higher dimensional object, can be used to infer properties and relations about the events taking place in the video sequences under consideration. The n D-EVM will play a paramount role because it is expected to take full advantage of its storage requirements and its algorithms' efficiency.

Finally, we conclude by commenting the n D-EVM is a model that has been successfully applied in tasks such as (1) image comparison by enhanced image-based reasoning, (2) collision detection of 2D and 3D objects whose trajectories are, respectively, modeled as 3D and 4D-OPPs, (3) concise representation and efficient querying of volume datasets, (4) morphological operations on binary images, and (5) connected components labeling. Applications 1, 2, and 3 are appropriately developed in [11] while applications 4 and 5 are presented in [22, 23]. In [9–11, 23] there are described algorithms based on the n D-EVM, besides the ones described in this paper, which are useful and efficient for performing other interrogations and manipulations on n D-OPPs.

References

- [1] R. S. Montero and E. Bribiesca, "State of the art of compactness and circularity measures," *International Mathematical Forum*, vol. 4, no. 25–28, pp. 1305–1335, 2009.

- [2] S. Marchand-Maillet and Y. M. Sharaiha, *Binary Digital Image Processing: A Discrete Approach*, Academic Press, San Diego, Calif, USA, 2000.
- [3] R. Osserman, "The isoperimetric inequality," *Bulletin of the American Mathematical Society*, vol. 84, no. 6, pp. 1182–1238, 1978.
- [4] J. Einenkel, U.-D. Braumann, L.-C. Horn et al., "Evaluation of the invasion front pattern of squamous cell cervical carcinoma by measuring classical and discrete compactness," *Computerized Medical Imaging and Graphics*, vol. 31, no. 6, pp. 428–435, 2007.
- [5] E. Bribiesca, "Measuring 2-D shape compactness using the contact perimeter," *Computers & Mathematics with Applications*, vol. 33, no. 11, pp. 1–9, 1997.
- [6] H. S. M. Coxeter, *Regular Polytopes*, The Macmillan, New York, NY, USA, 2nd edition, 1963.
- [7] A. Jonas and N. Kiryati, "Digital representation schemes for 3-D curves," Tech. Rep. CC PUB #114, The Technion—Israel Institute of Technology, Haifa, Israel, 1995.
- [8] T. Banchoff and J. Wermer, *Linear Algebra through Geometry*, Springer, New York, NY, USA, 2nd edition, 1992.
- [9] A. Aguilera and D. Ayala, "Orthogonal polyhedra as geometric bounds in constructive solid geometry," in *Proceedings of the 4th Symposium on Solid Modeling and Applications (SM '97)*, pp. 56–67, May 1997.
- [10] A. Aguilera, *Orthogonal polyhedra: study and application*, Ph.D. thesis, Universitat Politècnica de Catalunya, 1998.
- [11] R. Pérez-Aguila, *Orthogonal polytopes: study and application*, Ph.D. thesis, Universidad de las Américas—Puebla (UDLAP), 2006, http://catarina.udlap.mx/u_dl_a/tales/documentos/dsc/perez_a_r/.
- [12] R. Pérez-Aguila, "Representing and visualizing vectorized videos through the extreme vertices model in the n-dimensional space (nD-EVM)," *Journal Research in Computer Science*, vol. 29, pp. 65–80, 2007, Special issue: Advances in Computer Science and Engineering.
- [13] M. Spivak, *Calculus on Manifolds. A Modern Approach to Classical Theorems of Advanced Calculus*, W. A. Benjamin, Amsterdam, The Netherlands, 1965.
- [14] J. D. Foley, A. van Dam, S. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice in C*, Addison-Wesley Professional, 2nd edition, 1995.
- [15] L. K. Putnam and P. A. Subrahmanyam, "Boolean operations on n-dimensional objects," *IEEE Computer Graphics and Applications*, vol. 6, no. 6, pp. 43–51, 1986.
- [16] A. A. G. Requicha, "Mathematical models for rigid solids," Tech. Memo. 28, Production Automation Project, University of Rochester, Rochester, NY, USA, 1977.
- [17] T. A. Takala, "Taxonomy on geometric and topological models," in *Computer Graphics and Mathematics*, pp. 146–171, Springer, Berlin, Germany, 1992.
- [18] M. Mäntylä, "Boolean operations on 2-manifolds through vertex neighborhood classification," *ACM Transactions on Graphics*, vol. 5, no. 1, pp. 1–29, 1986.
- [19] D. M. Y. Sommerville, *An Introduction to the Geometry of N Dimensions*, Dover, New York, NY, USA, 1958.
- [20] CENAPRED (Centro Nacional de Prevención de Desastres), Mexico, February 2010, <http://www.cenapred.unam.mx/es/>.
- [21] E. Kurmyshev and R. Sánchez-Yáñez, "Particiones difusas para la umbralización multinivel de Imágenes," in *Proceedings of the 12th International Conference on Electronics, Communications and Computers (CONIELECOMP '02)*, pp. 224–227, Acapulco, Mexico, February 2002.
- [22] J. Rodríguez and D. Ayala, "Erosion and dilation on 2D and 3D digital images: a new size-independent approach," in *Proceedings of the 6th International Workshop on Vision Modeling and Visualization*, pp. 143–150, Stuttgart, Germany, November 2001.
- [23] J. Rodríguez and D. Ayala, "Fast neighborhood operations for images and volume data sets," *Computers & Graphics*, vol. 27, no. 6, pp. 931–942, 2003.