Chapter 7
Future Work

In this chapter we will discuss some lines of future work and some results based in our first approaches to these topics. Summarizing, there are two wide fields to consider in our future research:

- Precise determination of the new geometric transformation(s) that makes possible to determine the equivalencies between the 402 Hill's configurations and the 253 Aguilera & Pérez's Configurations for the 4D-OPP's (section 7.1).
- Formal specifications of the theoretical foundations and algorithms related to the extension of the Extreme Vertices Model to the four and five-dimensional spaces (section 7.2) and their related applications (sections 7.3 and 7.4) for the analysis and visualization of multidimensional data.

Although in this chapter we will present some results and applications based on the first approaches to these topics, the reader must consider them carefully because the basis in which they are supported still require a formal and careful inspection.

7.1 Future Work: Towards the Determination of the Equivalencies Between Hill's Configurations and Aguilera & Pérez's Configurations for the 4D-OPP's through a New Geometric Transformation

When Hill's configurations are represented through their adjacencies counting (see section 5.7.2) it results that some of them (different Hill's configurations) have the same counting, and therefore they could be considered that belong to a same Aguilera & Pérez's

configuration. For example, in the **Table 7.1** there are shown six hyper-boxes' sets that are representatives of six Hill's configurations, however all of them have the same adjacencies counting.

TABLE 7.1
Six Hill's configurations that supposedly belong to a same Aguilera & Pérez's configuration (own elaboration).

Hill's Configurations	**Aguilera & Pérez's Configuration**			
Hyper-boxes' combinations	Volume Adjacencies	Face Adjacencies	Edge Adjacencies	Vertex Adjacencies
0011110110000000 1001011110000000 0110101011000000 1001101011000000 0101011011000000 1000000111101000	4	6	4	1

As we saw in section 5.7.1 the determination of the Hill's configurations is based in the fact that the 402 hyper-boxes' sets that represent these configurations cannot be reduced to a lesser number because all the possible compositions of rotations and reflections were exhaustively tested on them. In others words, the Hill's configurations presented in **Table 7.1** cannot be reduced to the Aguilera & Pérez configuration with 4 volume, 6 face, 4 edge and 1 vertex adjacencies by using a composition exclusively integrated with rotations and/or reflections.

However, it should be possible to reduce the number of the Hill's configurations by considering the application of new geometric transformations. Our initial approach is the following: we will consider the application of only one geometric transformation which is additional to the possible composition of rotations and reflections defined in section 5.7.1.

We will generate all possible matrices with 4 columns and 4 rows and whose values will be in {-1, 0, 1}. One of these matrices will be added to the compositions considered in the Hill's configurations determination. The main idea behind this approach is that given two different Hill's configurations Cn_1 and Cn_2 but with the same adjacencies count, there is a composition T^n plus a matrix transformation T_X (i.e. a possible new geometric transformation) such that:

$$Cn_1 \equiv T^n \cdot T_X (Cn_2)$$

The following is an implementation of this idea. We will generate all possible matrices with 4 columns and 4 rows and whose sixteen values will be in {-1, 0, 1}. Each one of the possible 3^{16} = 43,046,721 matrices (matrix *xTransformation*) will be added to each one of the 20,480 possible combinations of rotations and/or reflections (see section 5.7.1); by this way we get a possible composition $T^n \cdot T_X$. The algorithm will receive as input a set of different Hill's configurations (the vector *HillConfigurations*) but with the same adjacencies counting; and a binary string that represents an Aguilera & Pérez's configuration (obviously with the same adjacencies counting of the Hill's configurations; one of the Hill's configurations can be selected for this end; *aguileraPerezConf* in the code). In the calling of the function *evaluateTransformationMatrixWithComposition*, each Hill's configuration will be transformed with all the possible $T^n \cdot T_X$ and evaluated against the *aguileraPerezConf*. We expect to find a valid transformation matrix when the Hill's configuration is converted into the *aguileraPerezConf*.

```
void findTransformationMatrices(Vector HillConfigurations,
                                                    BinaryString aguileraPerezConf)
{
    for(int i = 0; i < HillConfigurations.size( ); i++) {
        int xTransformation[4][4] = {{0,0,0,0}, {0,0,0,0}, {0,0,0,0}, {0,0,0,0}};
        BinaryString hill_conf = HillConfigurations.elementAt(i);
        for(int j = 1; j < 43046721; j++) {
            evaluateTransformationMatrixWithComposition
                                        (xTransformation, Hill_conf, aguileraPerezConf);
            getNextMatrix(xTransformation);
        }
    }
}

void evaluateTransformationMatrixWithComposition
        (int xTransformation[ ][ ], BinaryString hill_conf, BinaryString aguileraPerezConf)
{
    int composition[7] = {0,0,0,0,0,0,0};
    for(int i = 0; i < 78125; i++) {
        BinaryString cn = hill_conf.clone( );
        applyComposition(composition, cn);
        apply_XTransformation(xTransformation, cn);
        if(combinationIsValid(cn) == true)
            if(equals(cn, aguileraPerezConf) == true)
                /* It has been found a composition of rotations and/or reflections
                   and x-transformation that converts a Hill's configuration into a
                   Aguilera & Pérez's Configuration. */
        else
            return; //The matrix produces an invalid combination of hyper-boxes.
        getNextComposition(composition);
    }
}
```

Through this implementation we have found that there are at least two transformation matrices for any pair of Hill's configurations with the same adjacencies counting that convert between them. By this way it is possible to convert the set of 402 Hill's configurations into the 253 Aguilera & Pérez's configurations. For example, in the **Table 7.2** shows the transformation matrices found for two pairs of Hill's configurations with 6 hyper-boxes; these four Hill's configurations can belong to two Aguilera & Pérez's configurations through the application of their found transformation matrices between them. In the **Appendix B** are shown the possible equivalencies between the Hill and Aguilera & Pérez's configurations.

TABLE 7.2

The possible conversion of four Hill's configurations into two Aguilera & Pérez's configurations (own elaboration).

Hill's Configurations and Associated Matrix Transformation			**Aguilera & Pérez's Configurations**
Cn_1 (Binary Representation)	Transformation Matrix $T(Cn_1) = Cn_2$	Cn_2 (Binary Representation)	Adjacencies Counting
0101011110000000	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$	1011010110000000	5 volume adjacencies 5 face adjacencies 4 edge adjacencies 1 vertex adjacency
0101011110000000	$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$	1011010110000000	
0111010110000000	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$	1101100110000000	5 volume adjacencies 6 face adjacencies 3 edge adjacencies 1 vertex adjacency
0111010110000000	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 1 \end{bmatrix}$	1101100110000000	

However, through this first approach we have found that there are some pairs of Hill's configurations that have hundreds of valid transformation matrices. Moreover, we will have to assure, as part of our future work, that each one of the found valid matrices is not representing a composition of the considered transformations. By this way, we expect to count with a set of transformation matrices from which a new geometric transformation(s) should be determined.

7.2 The Extreme Vertices Model

In the following sections will be mentioned some of the basic concepts related to the Extreme Vertices Model for the representation of Orthogonal Pseudo-Polyhedra (EVM-3D) in a very concise form. This model also enables 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, other set membership classification operations and measure operations on 3D-OPP's. The EVM-3D was originally presented by Aguilera & Ayala in [Aguilera, 97] (for representing only Orthogonal Polyhedra) and widely described in [Aguilera, 98] (considering both Orthogonal Polyhedra and Pseudo-Polyhedra) where the aspects related to formalizations and proofs are treated with the proper detail.

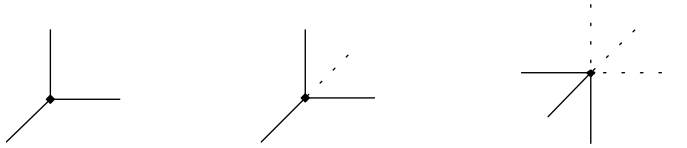
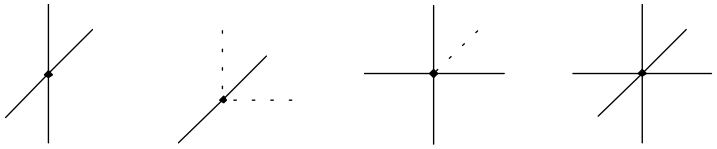
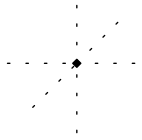
7.2.1 Brinks and Extreme Vertices in the 3D-OPP's

A brink is defined as the maximal uninterrupted segment, built out of a sequence of collinear and contiguous two-manifold edges of a 3D-OPP with the following properties:

- Non-manifold edges do not belong to brinks.
- Every two-manifold edge belongs to a brink, whereas every brink consists of m edges ($m \geq 1$), and contains $m+1$ vertices.
- Two vertices of type V3, V4N1 or V6N1 (**Table 7.3**) are at either extreme of the brink (***Extreme Vertices***). These vertices have in common that they are the only ones that have exactly three incident two-manifold and perpendicular edges, regardless of the number of incident non-manifold edges, therefore those vertices mark the end of brinks in all three orthogonal directions.

- The $m-1$ vertices of type V4, V4N2, V5N or V6 (**Table 7.3**) are the only common point of two collinear edges of a same brink (interior vertices).
- Due to all six incident edges of a V6N2 vertex (**Table 7.3**) are non-manifold edges, none of them belongs to a brink, thus this vertex does not belong to any brink.

TABLE 7.3
Vertices in the 3D-OPP's and their relation with brinks (own elaboration).

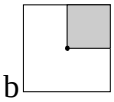
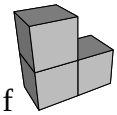
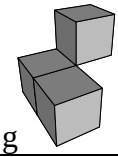
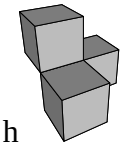
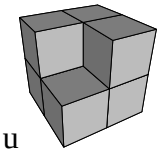
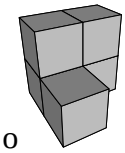
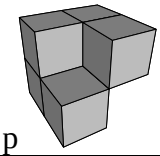
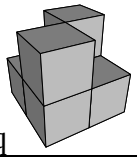
Extreme Vertices (The brinks' boundary)	 V3 V4N1 V6N1
Brinks' Interior Vertices	 V4 V4N2 V5N V6
Vertex that do not belong to any brink.	 V6N2

Based in the previous analysis for brinks we have the following properties for the **Extreme Vertices**:

- **Property 7.1:** Every Extreme Vertex of a nD-OPP ($1 \le n \le 3$) has exactly n incident manifold edges perpendicular to each other. This number is even for every non-extreme vertex.
- **Property 7.2:** Every Extreme Vertex of a 3D-OPP has an odd number of incident faces, and every non-extreme vertex has an even number of incident faces.

- **Property 7.3:** Any Extreme Vertex of a nD-OPP ($1 \leq n \leq 3$), when is locally described by a set of surrounding "boxes", is surrounded by an odd number of such "boxes". An even number of surrounding "boxes" either defines a non-extreme vertex, or does not define any vertex at all. See **Table 7.4**.

TABLE 7.4
The configurations from the nD-OPP's ($1 \leq n \leq 3$) that describe Extreme Vertices (own elaboration).

1D-OPP's	1 segment	
2D-OPP's	1 rectangle  b	3 rectangles  e
3D-OPP's	1 box  b	3 boxes  f  g  h
	7 boxes  u	5 boxes  o  p  q

The **Extreme Vertices of a 3D-OPP *p*** are the ending vertices of all the brinks in *p*. Let *V(p)* be the set of vertices in *p*, then $EV(p) \subseteq V(p)$ will denote to the set of the extreme vertices where $EV(p) = V3(p) \cup V4N1(p) \cup V6N1(p)$. *#p* will denote the cardinality of *V(p)*. For a 3D-OPP *p*, its **Extreme Vertices Model (EVM)** is the model that will only store to all extreme vertices from *p*. For any 3D-OPP *p*, *EV(P)* will have an even number of vertices, that is, *#p* is even. See the **Figure 7.1**.

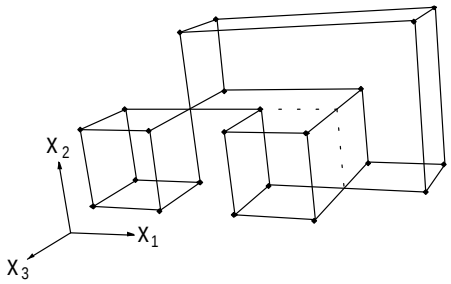
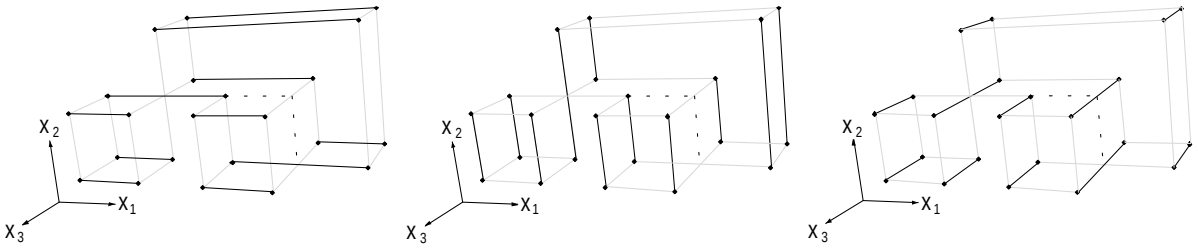


FIGURE 7.1
Example of a 3D-OPP *p* and its set of Extreme Vertices
(Continuous lines indicate manifold edges while the dotted lines indicate non-manifold edges; the points compose the set *EV(P)*; own elaboration).

The brinks in a 3D-OPP can be classified according to the main axis to which they are parallel. Since the extreme vertices mark the end of brinks in the three orthogonal directions, is that any of the three possible sets of brinks (parallel to X_1-axis, parallel to X_2-axis or parallel to X_3-axis, see **Figure 7.2**) will produce to the same set *EV(p)*.



a) b) c)

FIGURE 7.2
The brinks in a 3D-OPP (the OPP presented in Figure 7.1).
a) The brinks parallel to X_1-axis, b) the brinks parallel to X_2-axis,
c) the brinks parallel to X_3-axis (own elaboration).

7.2.2 Extended Faces and Extended Edges

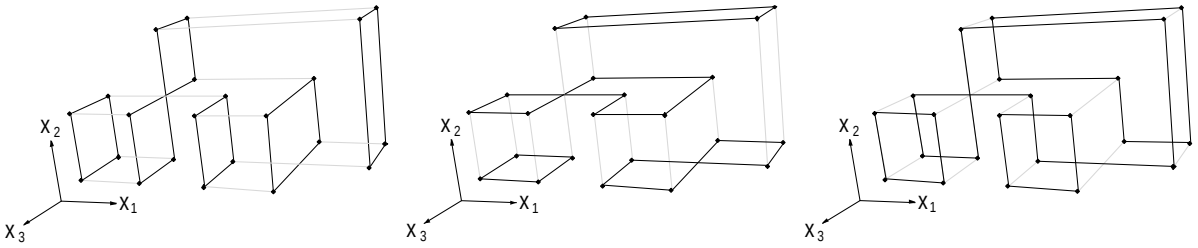
An **extended face** is the maximal set of faces lying on a plane perpendicular to one of the 3D space's main axes X_1, X_2 or X_3. These faces can be united by the edges or

vertices of a 3D-OPP. The faces of an extended face in a 3D-OPP can also be united by a non-manifold edge or by just one vertex which can be of the type V4, V5N, V6, V6N1 or V6N2. The EVM of an extended face (formerly a *plane of vertices,* according to the nomenclature presented in [Aguilera, 98]) of a 3D-OPP *p* is a subset from the *EV(p).*

An **extended edge** of a 2D-OPP *p* is the set of brinks that lie in a straight line that is parallel to a coordinate axis. The EVM of an extended edge (formerly a *line of vertices*, according to the nomenclature presented in [Aguilera, 98]) is a subset from the *EV(p).*

Both extended faces of a 3D-OPP and extended edges of a 2D-OPP will be referred here as Φ (formerly *plv*, according to [Aguilera,98]'s nomenclature) and each one will have an even number of vertices. A k-th extended face (or extended edge) of a 3D-OPP *p* will be referred as $\Phi_k(p)$.

An *EVM(p)* can be considered as a sequence of extreme vertices models *EVM(*$\Phi_1(p)$*), EVM(*$\Phi_2(p)$*), ..., EVM(*$\Phi_{np}(p)$*)* from its corresponding *np* extended faces. The number of elements *np* in this sequence is the number of different coordinates for the axis perpendicular to the extended faces $\Phi_1(p)$, $\Phi_2(p)$, ..., $\Phi_{np}(p)$. See **Figure 7.3**. The EVM of each extended face is at the same time the sequence of EVM's from its extended edges, and the EVM of a brink is defined by a pair of extreme vertices.



a) b) c)

FIGURE 7.3

The sequences of extended faces in a 3D-OPP (the OPP presented in Figure 7.1). a) The extended faces perpendicular to X_1-axis. b) The extended faces perpendicular to X_2-axis. c) The extended faces perpendicular to X_3-axis (own elaboration).

7.2.3 Slices

An **slice** is the region of a 3D-OPP contained between the corresponding supporting planes of two consecutive extended faces. A k-th slice of a 3D-OPP *p* is denoted by s*lice*$_k$*(p)*. Then $p = \bigcup_{k}^{np-1} slice_k(p)$. See **Figure 7.4** for an example.

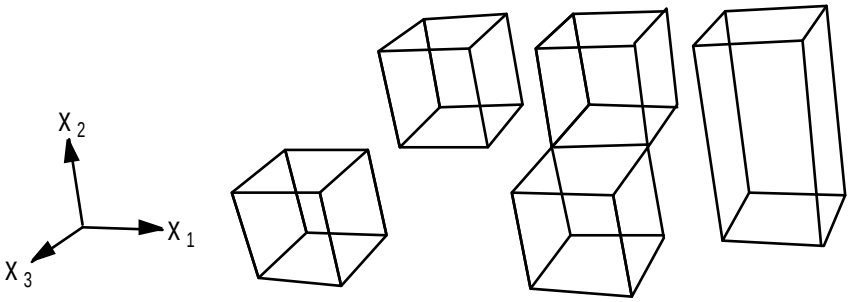
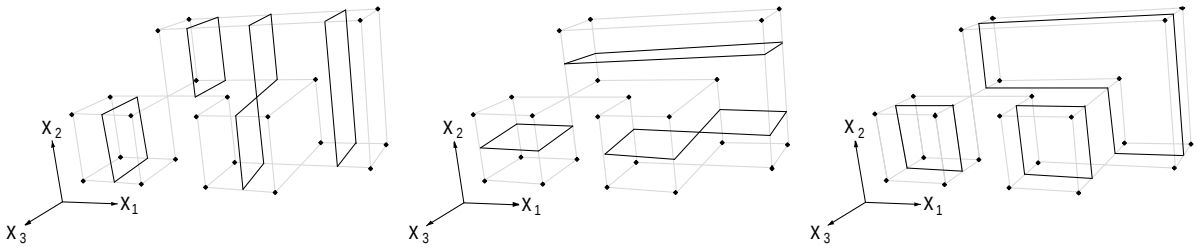


FIGURE 7.4

The slices of a 3D-OPP (presented in Figure 7.1. There are presented the regions from the 3D-OPP between the supporting planes of the planes of vertices perpendicular to X_1-axis; own elaboration).

7.2.4 Sections

A section is the resulting 2D-OPP from the intersection between a 3D-OPP and a plane which is perpendicular to one of the main axis. That 2D-OPP doesn't coincide with any extended face but it is parallel to all of them. Furthermore, it is called an *internal section* from *p* if the intersection between the 3D-OPP and the plane is not empty, otherwise it is called an **external section**. A k-th section of *p* between $\Phi_k(p)$ and $\Phi_{k+1}(p)$ is referred by $S_k(p)$. The slices of a 3D-OPP's are a set of one or more disjoint prisms whose base is the section of each slice. In **Figure 7.5** are presented the sections for the 3D-OPP from **Figure 7.1**.



a) b) c)

FIGURE 7.5

The sections of a 3D-OPP (the OPP presented in Figure 7.1). a) The internal sections perpendicular to X_1-axis. b) The internal sections perpendicular to X_2-axis. c) The internal sections perpendicular to X_3-axis (own elaboration).

7.2.5 Computing the Extended Faces Through Sections

Let *p* be a (d-1)-dimensional OPP embedded in E^d, then $\overline{p}$ will denote to the projection of *p* on a (d-1)-dimensional hyperplane parallel to *p*.

The projection of an extended face $\overline{\Phi_k(p)}$ of a 3D-OPP *p* can be obtained by computing the regularized XOR between the projections of its previous section $\overline{S_{k-1}(p)}$ and its following section $\overline{S_k(p)}$. Then we have that:

$$\overline{\Phi_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{S_k(p)}, \forall_k \in [1, np]$$

7.2.6 Computing the Sections Through the Extended Faces

The projection $\overline{S_k(p)}$ of any section from a 3D-OPP *p* can be obtained by computing the regularized XOR between the projection of the section $\overline{S_{k-1}(p)}$ and the projection of the extended face $\overline{\Phi_k(p)}$. Then we have that:

$$S_0(p) = \varnothing$$

$$\overline{S_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{\Phi_k(p)}, \forall_k \in [1, np]$$

Or in an equivalent way by computing the regularized XOR of the projection of all the previous extended faces:

$$\overline{S_k(p)} = \bigotimes_{i=1}^{k} * \overline{\Phi_i(p)}$$

The projection of the first and last extended faces of any 3D-OPP *p* must coincide with the projection of the first and last internal sections of p, that is to say, $\overline{S_1(p)} = \overline{\Phi_1(p)}$ and $\overline{S_{np-1}(p)} = \overline{\Phi_{np}(p)}$.

7.2.7 Virtual Extended Faces

An empty extended face is called a virtual extended face. Let *p* be an arbitrary 3D-OPP, then we say that *p* has a virtual extended face $\Phi_k(p)$ which is perpendicular to one of the main axes if there are no vertices of *p* in such extended face. We know that starting from the consecutive sections $S_{k-1}(p)$ and $S_k(p)$ it is possible to obtain $\Phi_k(p)$ through $\overline{\Phi_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{S_k(p)}$. But if $\overline{S_{k-1}(p)} = \overline{S_k(p)}$ then obviously $\Phi_k(p) = \varnothing$. That means that any number of virtual extended faces can be considered, as required, without altering to *p*.

7.2.8 <u>Future Work:</u> Towards the Extreme Vertices Model in the 4D and 5D Spaces

This section presents our first <u>experimental</u> results about the representation of 4D and 5D Orthogonal Pseudo-Polytopes (4D-OPP's and 5D-OPP's) through a single subset of their vertices. In order not to repeat the same words, sometimes we use parenthesis for the 5D case. *Although some of the following results are promising, the reader must consider them carefully because a deep inspection of the theoretical foundations and algorithms is still required.*

Consider the construction of a 4D-OPP as the union of several 4D-OPP's in the following way (**Figure 7.6**):

- We will have a hyperdimensional "*L-shaped*" polytope *a* in **Figure 7.6**, and
- Three four-dimensional hypercubes *b*, *c* and *d*.
- The polytope *a* will share a vertex with hypercube *c* and a face with hypercube *b*.
- The hypercube *b* will share an edge with hypercube *d*.

See the final 4D-OPP in the **Figure 7.7**. We will use it the following sections to exemplify some aspects about the EVM-4D.

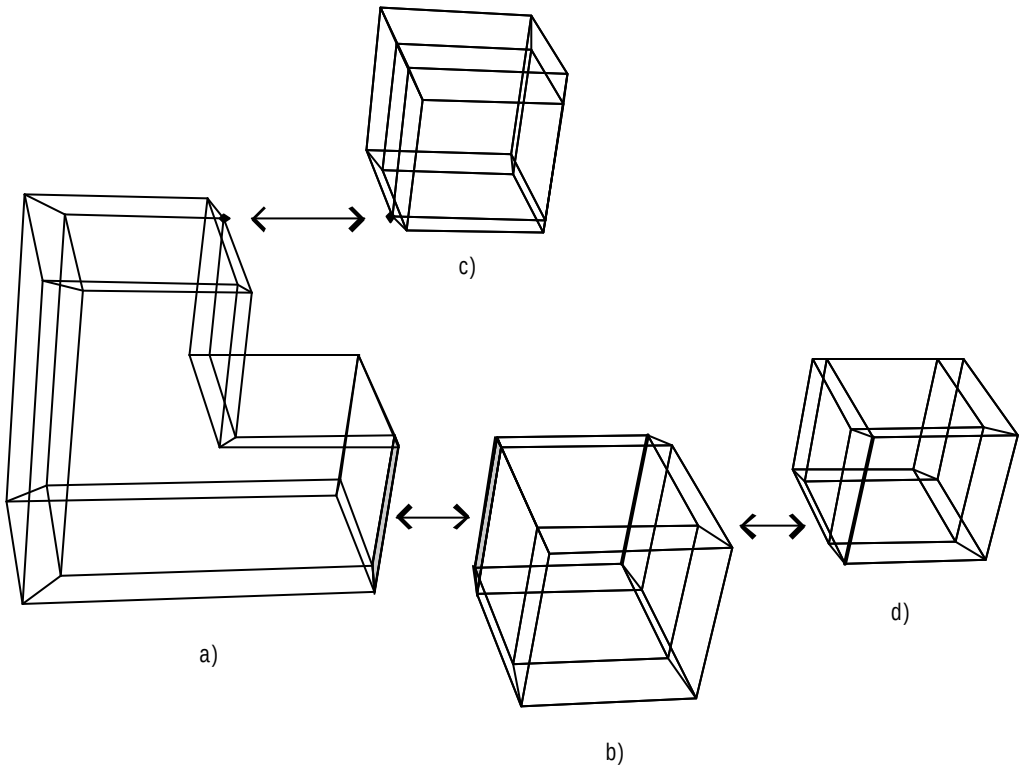


FIGURE 7.6
The construction of a 4D-OPP by the union of several 4D-OPP's
(a hyperdimensional "*L-shaped*" polytope and three hypercubes; own elaboration).

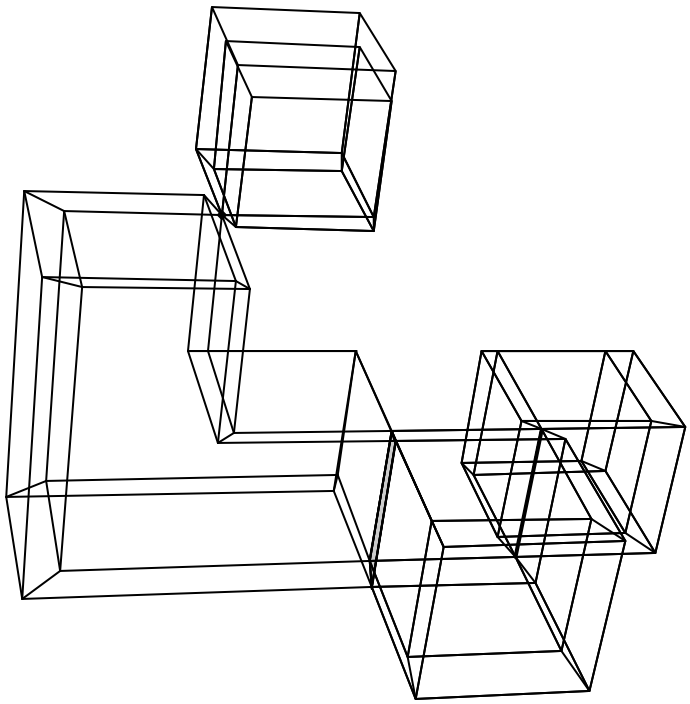


FIGURE 7.7
A 4D-OPP resulting from the union of several 4D polytopes
(seen in Figure 7.6; own elaboration).

The **property 7.3** says that any Extreme Vertex of a nD-OPP ($1 \leq n \leq 3$), when is locally described by a set of surrounding boxes, is surrounded by an odd number of such boxes. We will assume now that this is true for $n \geq 1$. Then, by instantiation, any vertex of a (5D) 4D-OPP when is surrounded by an odd number of (5D) 4D hyper-boxes should be an extreme vertex. In fact, there are 1 configuration with 1 or 15 hyper-boxes, 6 configurations with 3 or 13 hyper-boxes, 20 configurations with 5 or 11 hyper-boxes and 30 configurations with 7 or 9 hyper-boxes in the 4D-OPP's that should describe extreme vertices (see **Appendix A**). Moreover, through the 5D-OPP's configurations counting (presented in Section 5.3) there are 1 configuration with 1 or 31 hyper-boxes, 10 configurations with 3 or 29 hyper-boxes, 66 configurations with 5 or 27 hyper-boxes, 236 configurations with 7 or 25 hyper-boxes, 570 configurations with 9 or 23 hyper-boxes, 989

configurations with 11 or 21 hyper-boxes, 1,406 configurations with 13 or 19 hyper-boxes and 1,607 configurations with 15 or 17 hyper-boxes that should describe Extreme Vertices.

We will consider that every (5D) 4D-OPP *p* is initially represented through a hypervoxelization (see section 6.4.3). Then, we will select only such vertices with an odd number of incident (5D) 4D hyper-boxes. We will call to the resultant set of vertices the extreme vertices of *p*, that is, *EV(p)*. **Figure 7.8** shows the set of extreme vertices of the 4D-OPP of **Figure 7.7**.

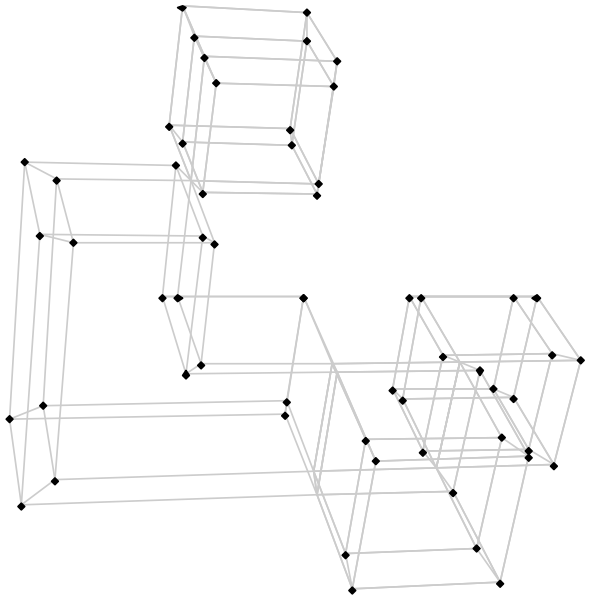


FIGURE 7.8
A 4D-OPP (from Figure 7.7) and its set of Extreme Vertices (own elaboration).

We will refer to a brink as the segment defined by two consecutive extreme vertices that lie on a line parallel to one of the (5D) 4D space's main axes. Moreover, brinks must be between and odd-numbered extreme vertex and an even-numbered extreme vertex in that order (see **Figure 7.9**). There is no brink between an even one and any odd one. In each

dimension, every extreme vertex has just one incident brink, thus in all the (five) four dimensions every extreme vertex has exactly (five) four incident brinks perpendicular to each other.

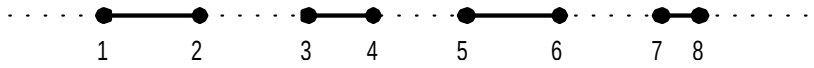


FIGURE 7.9

Numbering the extreme vertices that lie on a line parallel to one of the space's main axes and composing their corresponding brinks (own elaboration).

For example, in **Figure 7.10** are respectively shown:

- The parallel brinks to X_1-axis (**7.10.a**);
- The parallel brinks to X_2-axis (**7.10.b**);
- The parallel brinks to X_3-axis (**7.10.c**);
- The parallel brinks to X_4-axis (**7.10.d**);

From the 4D-OPP presented in **Figure 7.7**.

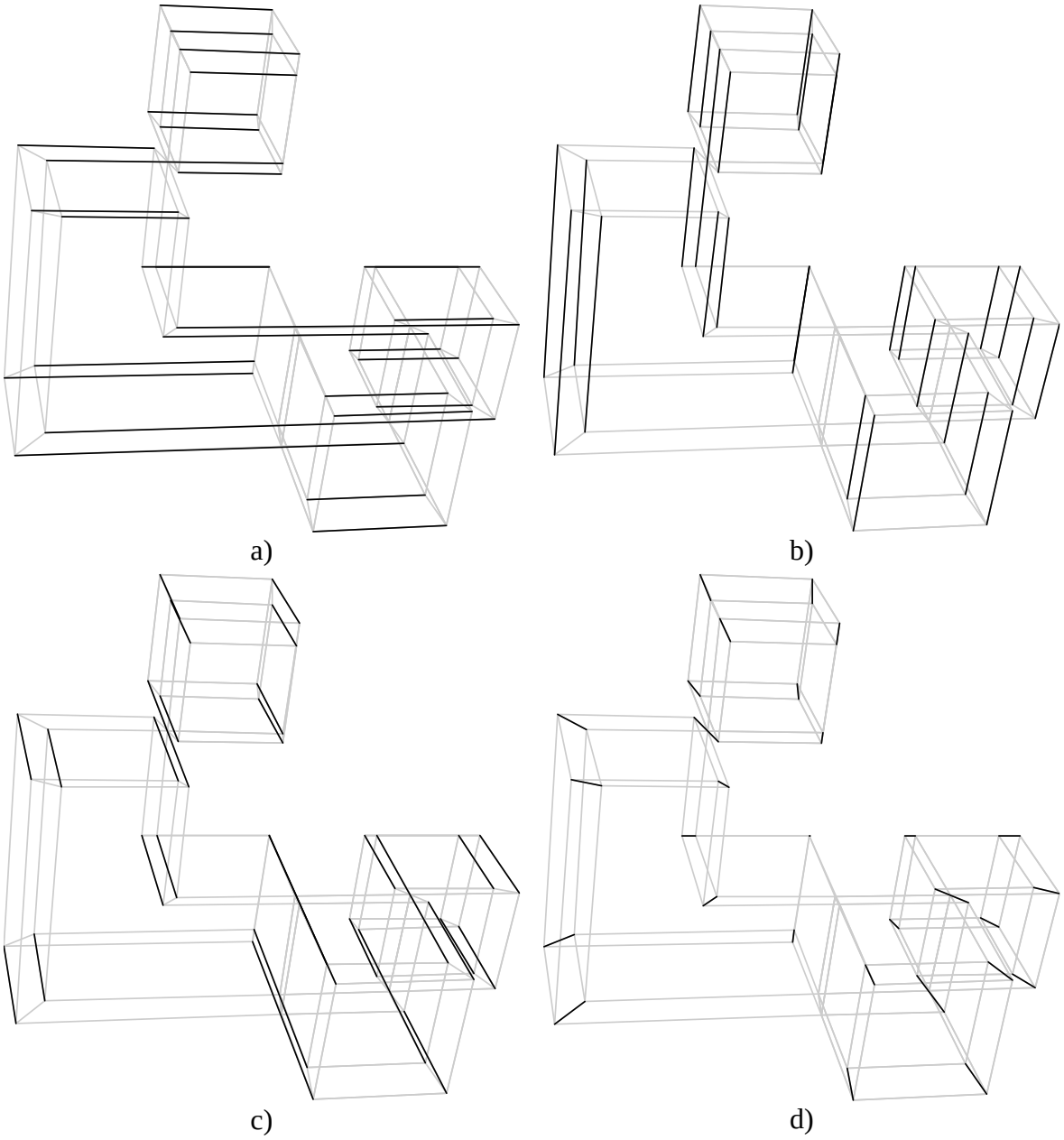


FIGURE 7.10
A 4D-OPP (from Figure 7.7) and its brinks parallel to X_1 (a), X_2 (b), X_3 (c) and X_4 (d) axes (own elaboration).

We will consider an extended (hypervolume) volume as the set of (hypervolumes) volumes of a (5D) 4D-OPP *p* lying on a (4D) 3D hyperplane perpendicular to one of the (5D) 4D space's main axes X_1, X_2, X_3, X_4 (or X_5). The (hypervolumes) volumes in an

extended (hypervolume) volume can be joined by lower dimensional elements as (volumes), faces, edges or vertices of a (5D) 4D-OPP. We will assume that the set of extreme vertices of a extended (hypervolume) volume is a subset of *EV(p)*.

Moreover, we will also refer to the extended (hypervolumes) volumes for (5D) 4D Orthogonal Pseudo-Polytopes as Φ. Moreover, $\Phi_k(p)$ will refer to the k-th (5D) 4D extended (hypervolume) volume of a (5D) 4D polytope *p*. We will also expect that the number *np* of Φ's in a (5D) 4D-OPP *p* is the number of different coordinates for the axis perpendicular to these Φ's.

For example, in **Figure 7.11** are respectively shown (extended volumes):

- The Φ's perpendicular to X_1-axis (**7.11.a**);
- The Φ's perpendicular to X_2-axis (**7.11.b**);
- The Φ's perpendicular to X_3-axis (**7.11.c**);
- The Φ's perpendicular to X_4-axis (**7.11.d**);

From the 4D-OPP presented in **Figure 7.7**.

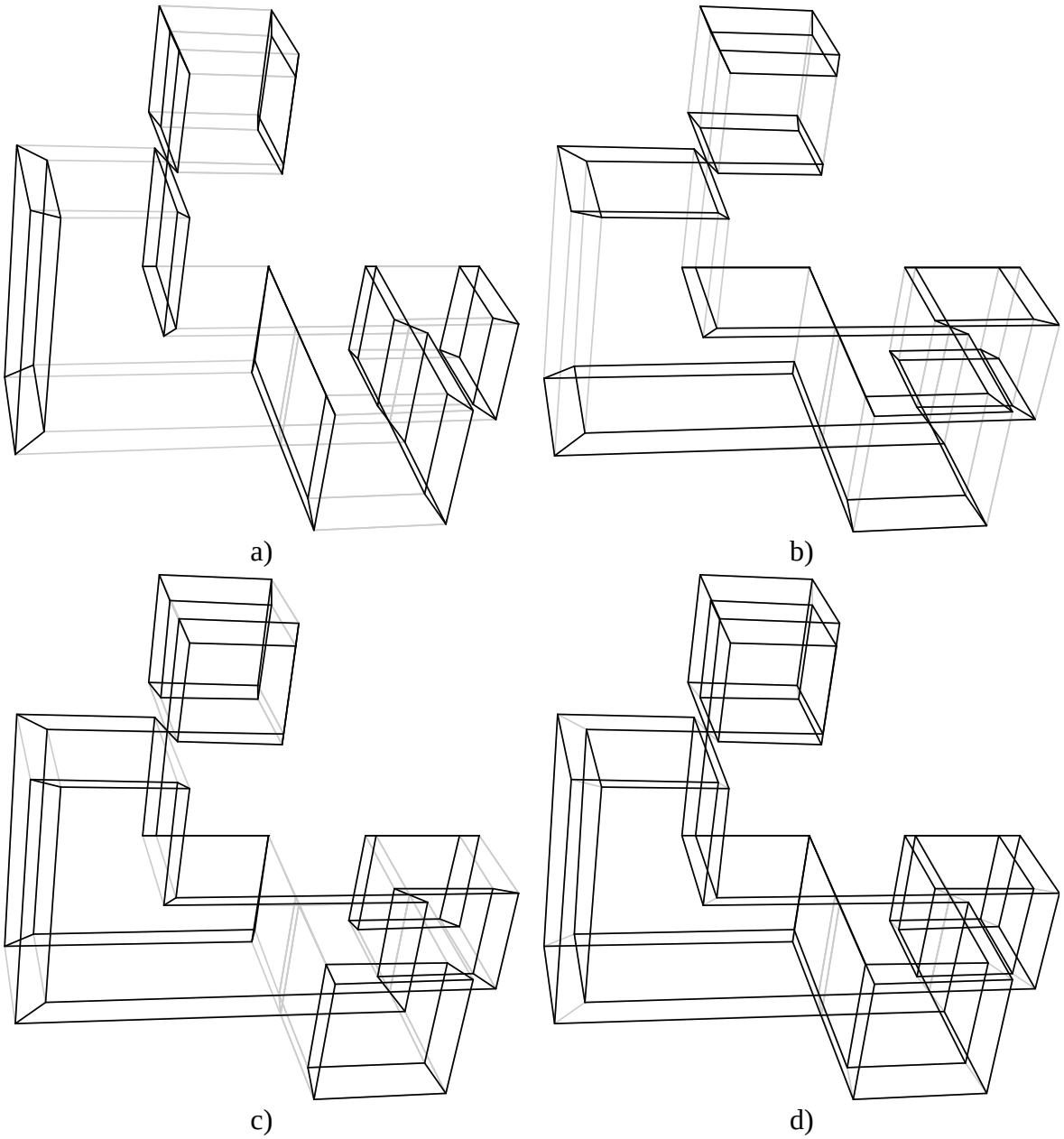


FIGURE 7.11
A 4D-OPP (from Figure 7.7) and its Φ's (extended volumes) perpendicular to X_1 (a), X_2 (b), X_3 (c) and X_4 (d) axes (own elaboration).

We will consider that a slice is the region contained in a (5D) 4D-OPP between the supporting (4D) 3D hyperplanes of two consecutive extended (hypervolumes) volumes. Therefore we can assume that $Slice_k(p)$ will denote the k-th slice of a (5D) 4D-OPP *p*. Hence $p = \bigcup_{k}^{np} slice_k(p)$.

In the **Figure 7.12.a** are shown the regions between the extended volumes perpendicular to X_1-axis of the 4D-OPP presented in **Figure 7.7**. Finally, in the **Figure 7.12.b** are shown the 4D-OPP's slices.

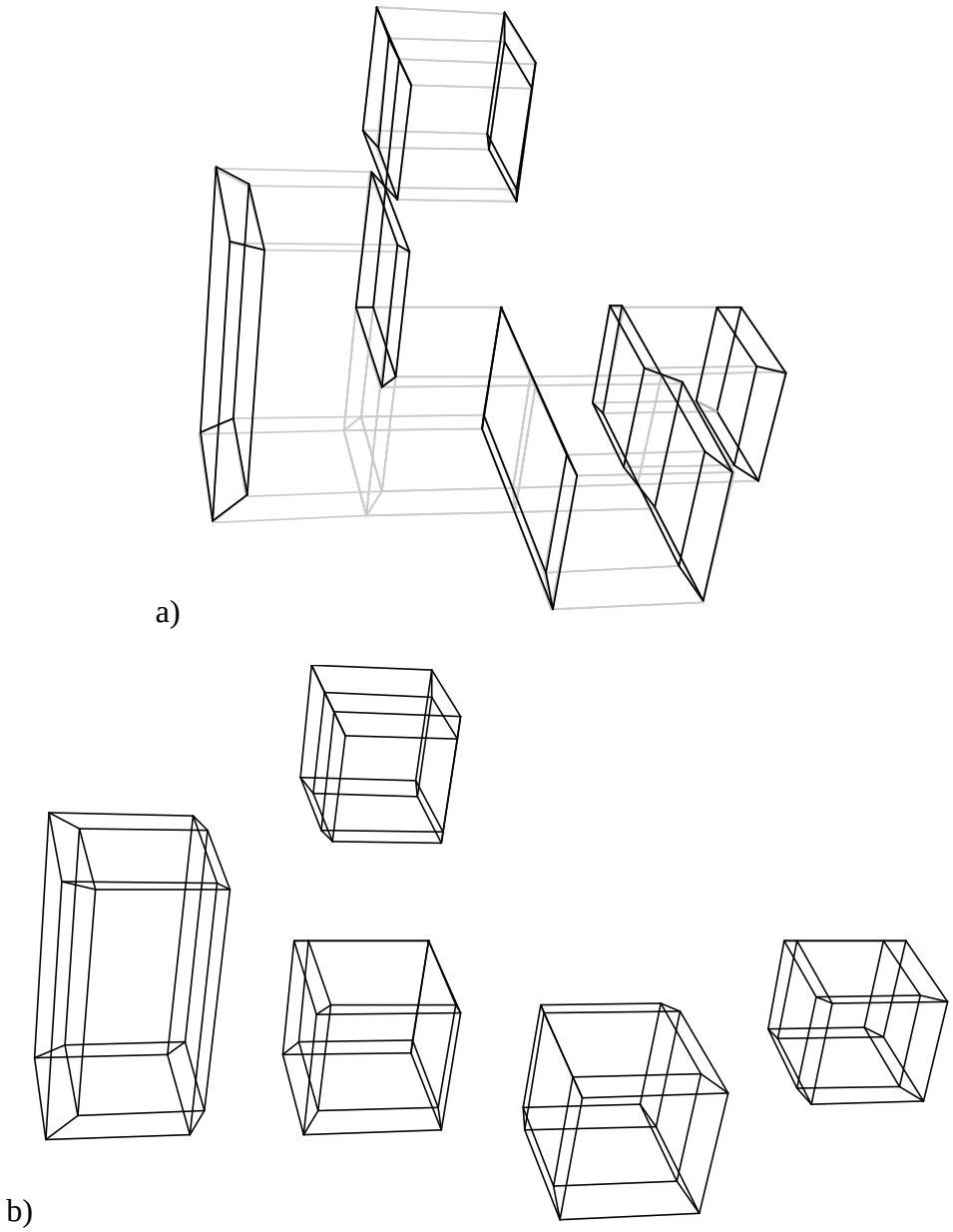


FIGURE 7.12
The regions of a 4D-OPP (presented in Figure 7.7) between its extended volumes perpendicular to X_1-axis (a) and its respective slices (b; Own elaboration).

We will say that a section is the (4D) 3D-OPP resulting from the intersection between a (5D) 4D-OPP and an orthogonal (4D) 3D hyperplane perpendicular to a coordinate axis which does not coincide with the supporting (4D) 3D hyperplane of any extended (hypervolume) volume. Furthermore, it will be called *external* or *internal* section of *p*, respectively, if this intersection is empty or not.

$S_k(p)$ will refer to the k-th section of *p* between $\Phi_k(p)$ and $\Phi_{k+1}(p)$.

For example, in **Figure 7.13** are respectively shown:

- The sections perpendicular to X_1-axis (**7.13.a**);
- The sections perpendicular to X_2-axis (**7.13.b**);
- The sections perpendicular to X_3-axis (**7.13.c**);
- The sections perpendicular to X_4-axis (**7.13.d**);

From the 4D-OPP presented in **Figure 7.7**.

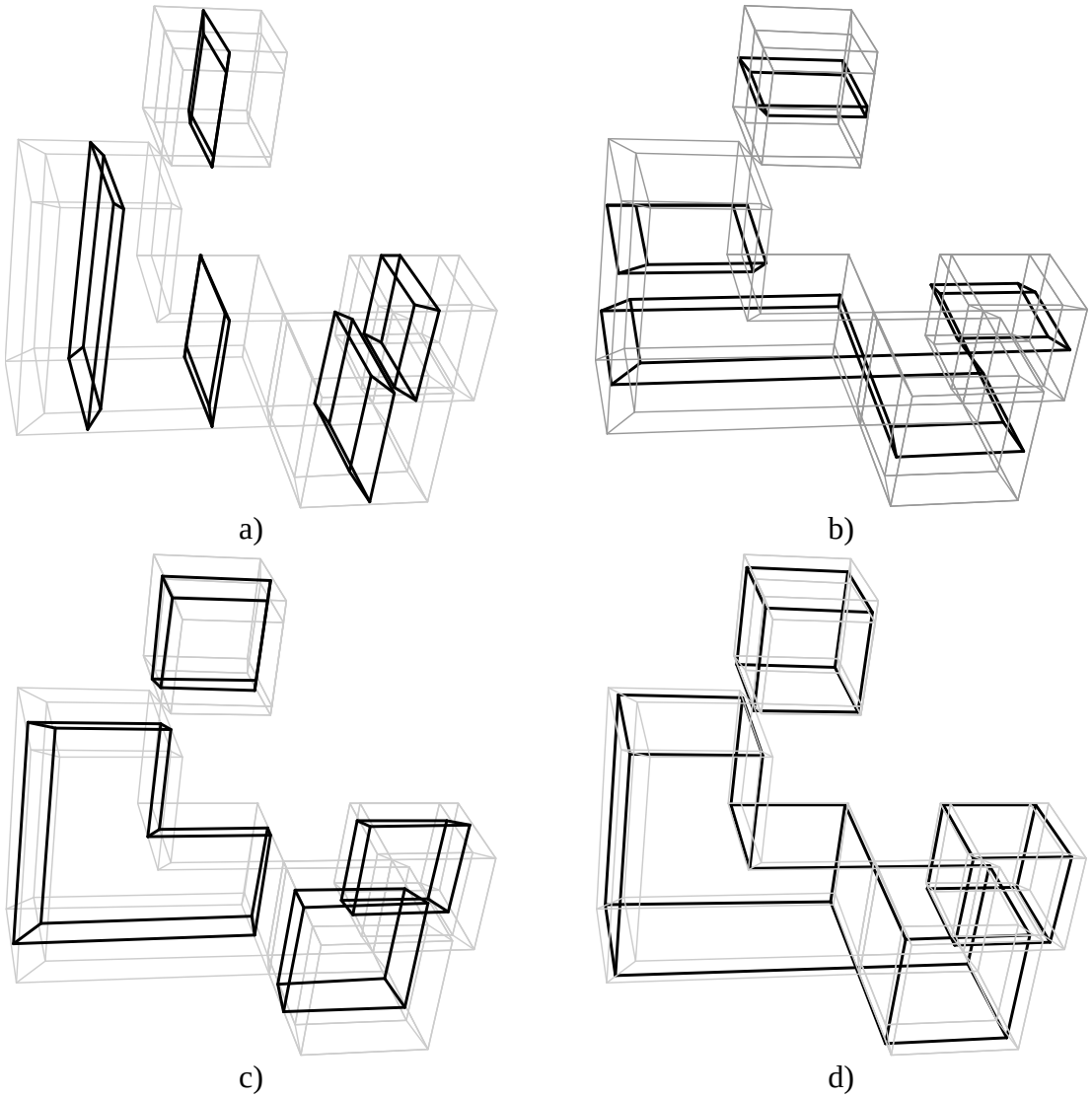


FIGURE 7.13
A 4D-OPP (from Figure 7.7) and its sections perpendicular to X_1 (a), X_2 (b), X_3 (c) and X_4 (d) axes (own elaboration).

A slice from a (5D) 4D-OPP is a set of one or more disjoint (5D) 4D hyperprisms whose base is the slice's section. A (5D) 4D hyperprism is generated by the parallel motion of a (4D polytope) polyhedron; it is bounded by the (4D polytope) polyhedron in its initial and final positions and by several prisms [Sommerville, 58] (a special case of a [5D] 4D hyperprism is a [5D] 4D hypercube generated according to section 2.2.1.1).

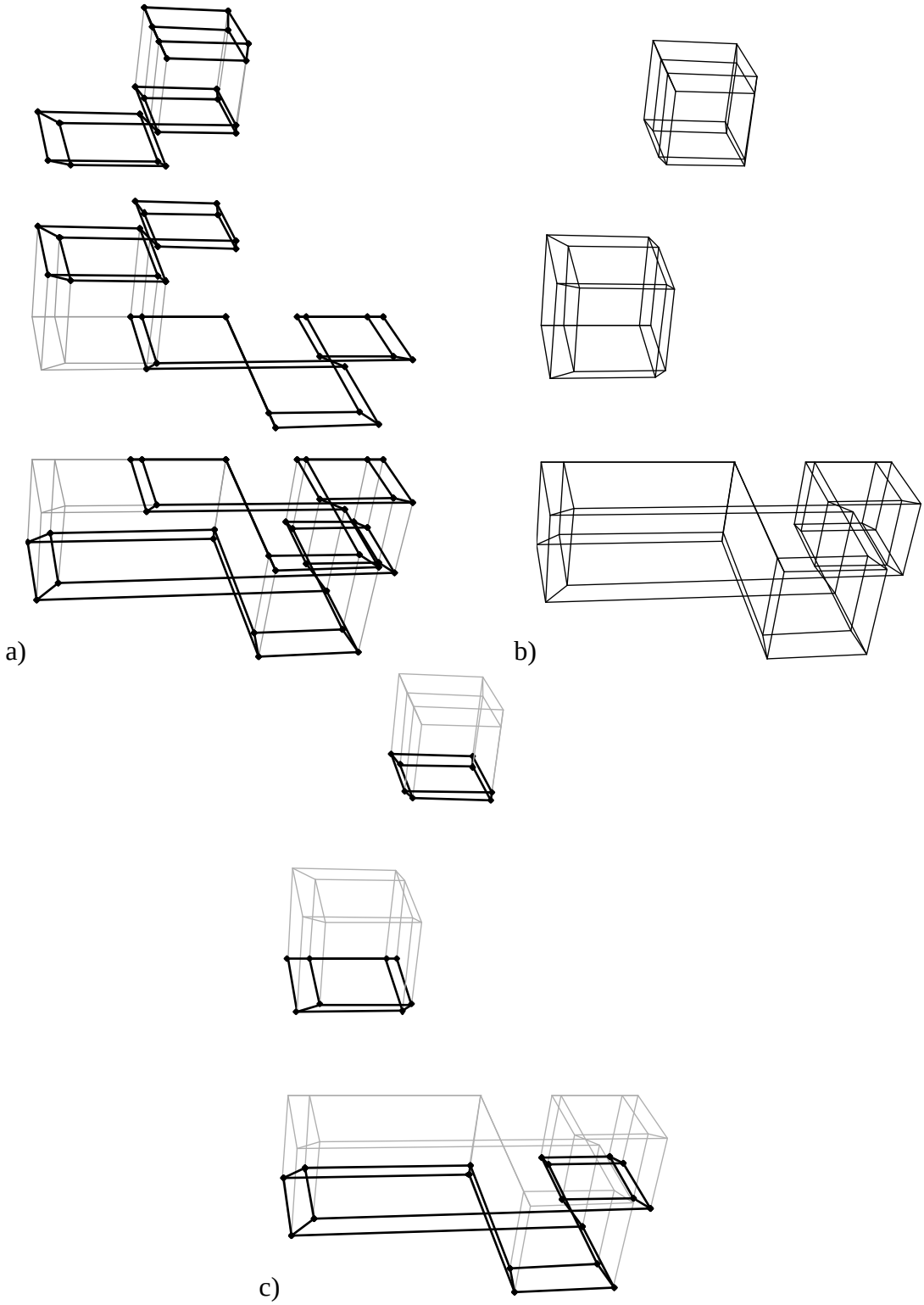


FIGURE 7.14

A 4D-OPP (presented in Figure 7.7) and a) its regions between its extended volumes perpendicular to X_2-axis; b) its slices and c) its slices showing their respective sections (also perpendicular to X_2-axis) as their bases (own elaboration).

All the orthogonal (4D) 3D hyperplanes intersecting a (5D) 4D-OPP in the same slice give the same section. Hence, every n-dimensional slice has its representing (n-1)-dimensional section. See **Figure 7.14**.

Consider p as a (four-) three-dimensional OPP embedded in the (fifth-) four-dimensional space, then $\overline{p}$ will denote the projection of p onto a (4D) 3D hyperplane parallel to p. This way we can consider that the projection of the set of extended (hypervolumes) volumes $\overline{\Phi_k(p)}$ of a (5D) 4D-OPP, p, can be obtained by computing the regularized XOR between the projections of its previous $\overline{S_{k-1}(p)}$ and next $\overline{S_k(p)}$ sections (this is an extension of the procedure presented in [Aguilera, 98], see section 7.1.6):

$$\overline{\Phi_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{S_k(p)},\ \forall_k \in [1, np].$$

Moreover, the projection of any section $\overline{S_k(p)}$, of a (5D) 4D-OPP, p, can be obtained by computing the regularized XOR between the projection of its previous section $\overline{S_{k-1}(p)}$ and the projection of its previous extended (hypervolume) volume $\overline{\Phi_k(p)}$. Or, equivalently, by computing the regularized XOR of the projection of all the previous extended (hypervolumes) volumes (this is also an application of the procedure presented in [Aguilera, 98], see section 7.1.7):

$$\begin{cases} S_0(p) = \varnothing \\ \overline{S_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{\Phi_k(p)}, \forall_k \in [1, np] \end{cases}$$

That is:

$$\overline{S_k(p)} = \bigotimes_{i=1}^{k} * \overline{\Phi_i(p)}$$

The projection of the first and last extended (hypervolumes) volumes of any (5D) 4D-OPP *p* should coincide with the projection of the first and last internal sections of *p*, that is:

$$\overline{S_1(p)} = \overline{\Phi_1(p)} \text{ and } \overline{S_{np-1}(p)} = \overline{\Phi_{np}(p)}.$$

Now, we will extend some of the concepts originally presented in [Aguilera, 98] for the achievement of Boolean operations between (5D) 4D-OPP's. Let *p* and *q* be two (5D) 4D-dimensional OPP's with *EV(p)* and *EV(q)* as their respective extreme vertices, then $EV(p \otimes^{*} q) = EV(p) \otimes EV(q)$. This expression allow us to suggest formulas for computing the (4D) 3D-dimensional hyperplanes of vertices of the (5D) 4D-OPP's through their sections and vice versa. Then we can expect that:

- $EV(\Phi_k(p)) = EV(S_{k-1}(p)) \otimes EV(S_k(p))$
- $EV(S_k(p)) = EV(S_{k-1}(p)) \otimes EV(\Phi_k(p))$

Moreover, two expressions can be also suggested for using the XOR operator. These expressions should allow the computing of the union and the difference of two (5D) 4D-OPP's whose specific situations are previously known:

- Let p and q be two <u>disjoint or quasi-disjoint</u> (5D) 4D-OPP's (that is, $p \cap^* q = \varnothing$) with *EV(p)* and *EV(q)* as their respective sets of extreme vertices, then:

$$EV(p \cup^* q) = EV(p) \otimes EV(q) .$$

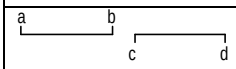
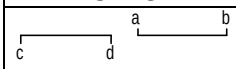
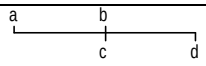
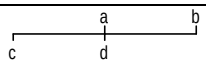
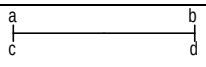
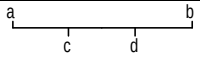
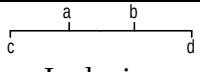
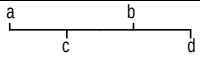
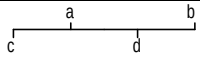
- Let p and q two (5D) 4D-OPP's such that $p \supseteq q$ with *EV(p)* and *EV(q)* as their respective sets of extreme vertices, then:

$$EV(p -^* q) = EV(p) \otimes EV(q)$$

Let p and q be two (5D) 4D-OPP's and $r = p\ op^*\ q$ where op^* is in $\{\cup^*, \cap^*, -^*, \otimes^*\}$. A Boolean regularized operation op^* between p and q, each one expressed with its set of extreme vertices, should be performed by the same op^* by applying it over their sections also expressed through their sets of extreme vertices; these sections will be (4D) 3D-OPP's. These situation lead us to a recursive process, for computing the Boolean regularized operations, which descends in the number of dimensions [Aguilera, 98]. The recursion's basic case is defined by the Boolean operations between two 1D-OPP's (**Table 7.5**).

TABLE 7.5

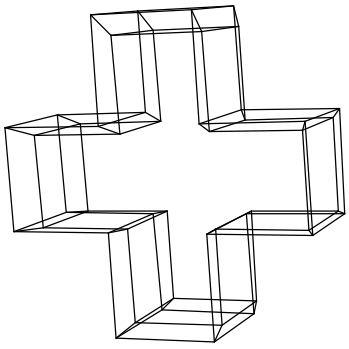
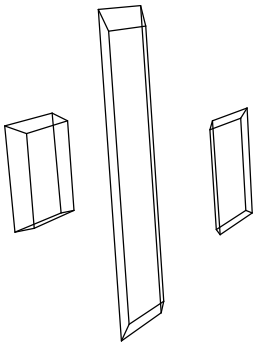
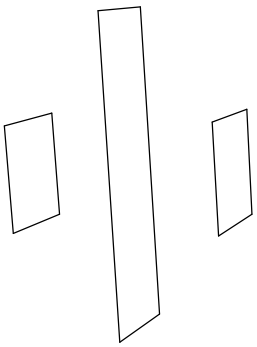
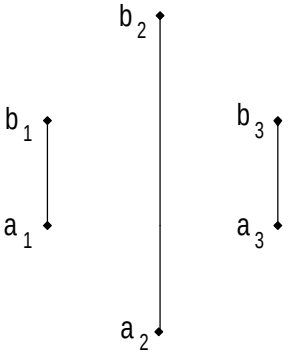
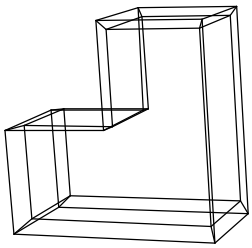
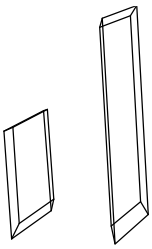
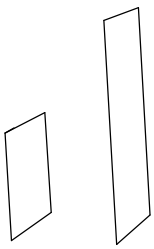
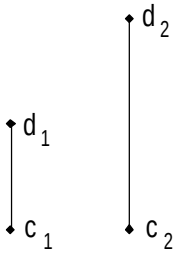
The Boolean regularized operations between two 1D-OPP's and their possible cases (own elaboration).

If $\overline{ab}$ & $\overline{cd}$ are:	$EVM(\overline{ab} \cup^* \overline{cd})$	$EVM(\overline{ab} \cap^* \overline{cd})$	$EVM(\overline{ab} -^* \overline{cd})$	$EVM(\overline{ab} \otimes^* \overline{cd})$
 Disjoint: b < c	{a, b, c, d}	∅	{a, b}	{a, b, c, d}
 Disjoint d < a	{c, d, a, b}	∅	{a, b}	{c, d, a, b}
 Contiguous: b = c	{a, d}	∅	{a, b}	{a, d}
 Contiguous: a = d	{c, b}	∅	{a, b}	{c, b}
 Coincident: a = c y b = d	{a = c, b = d}	{a = c, b = d}	∅	∅
 Inclusive ($\overline{ab} \supset \overline{cd}$): a < c < d < b a = c < d < b a < c < d = b	{a, b} {a = c, b} {a, d = b}	{c, d} {a = c, d} {c, d = b}	{a, c, d, b} {d, b} {a, c}	{a, c, d, b} {d, b} {a, c}
 Inclusive ($\overline{ab} \subset \overline{cd}$): c < a < b < d c = a < b < d c < a < b = d	{c, d} {c = a, d} {c, b = d}	{a, b} {c = a, b} {a, b = d}	∅ ∅ ∅	{c, a, b, d} {b, d} {c, a}
 Overlapping: a < c < b < d	{a, d}	{c, b}	{a, c}	{a, c, b, d}
 Overlapping: c < a < d < b	{c, b}	{a, d}	{d, b}	{c, a, d, b}

Now we will consider an example. Let A and B the two 4D-OPP's operands of the **Table 7.6**. The 4D-OPP A can be seen as a four-dimensional "*cross-shaped*" polytope and the 4D-OPP B can be considered as a four-dimensional "*L-shaped*" polytope (see **Table 7.6**'s first column). The operand A has three sections while operand B has only two (see **Table 7.6**'s second column). Each 3D section will have only one 2D section (since they are only rectangular prisms; third column). Finally, each 2D section will have only one 1D section: a segment with their respective pair of extreme vertices (fourth column). The 1D sections' extreme vertices for operand A are labeled as a_i and b_i while the 1D sections' extreme vertices for operand B are labeled as c_i and d_i.

TABLE 7.6

Two 4D-OPP's A & B and their corresponding sections since the 3D case until the 1D case (see text for details; own elaboration).

4D-OPP's	**Sections (3D-OPP's)**	**Sections (2D-OPP's)**	**Sections (1D-OPP's)**
 A			
 B			

The relative position for the Boolean operation is shown in **Figure 7.15.a** (the Boolean operation between the two 4D-OPP's). In the **Figure 7.15.b** is shown how interact the 3D sections for operands A and B (the Boolean operation between the 3D sections). In **Figure 7.15.c** are shown the interactions between the 2D sections (the Boolean operation between the 2D sections). Finally, in **Figure 7.15.d** are shown the interactions between the 1D sections (the basic case for the Boolean operations).

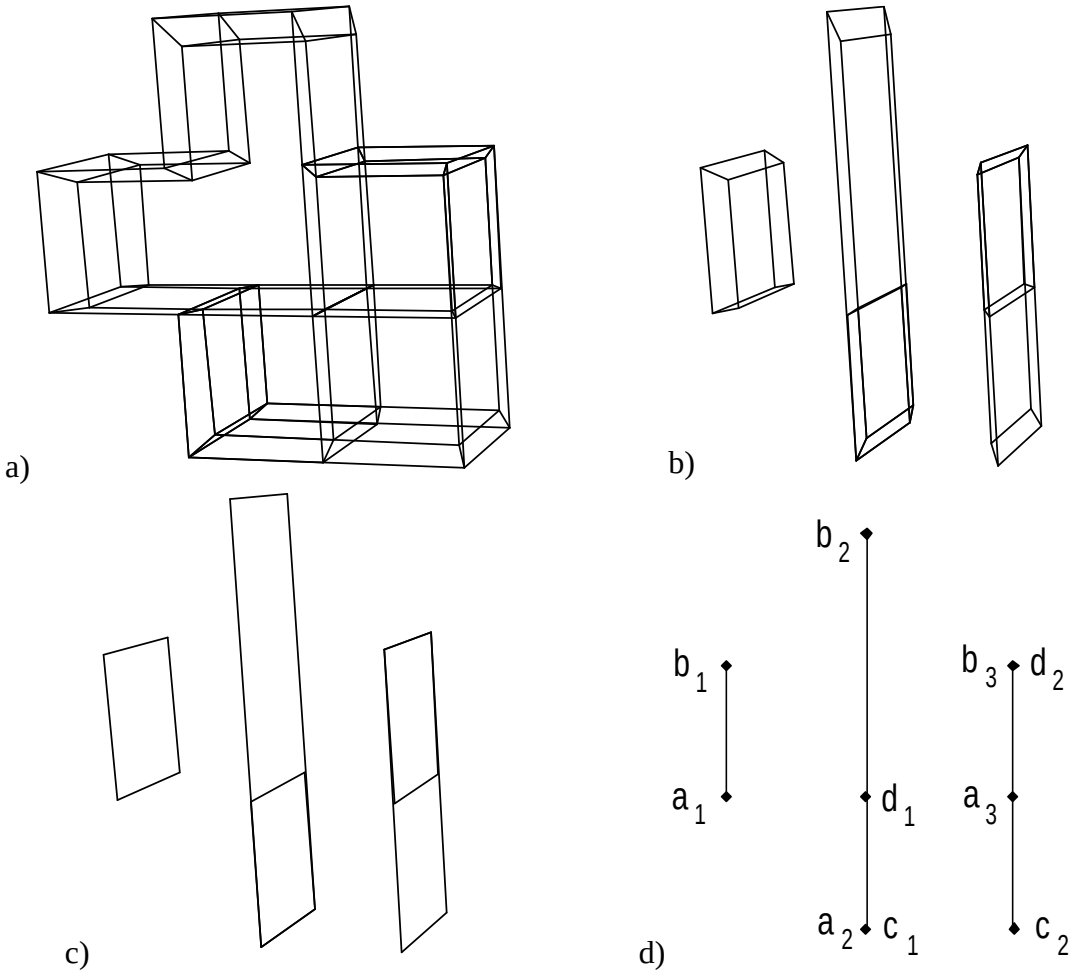


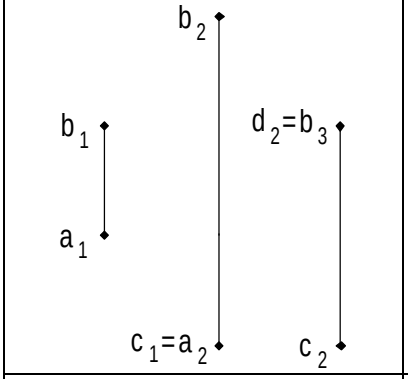
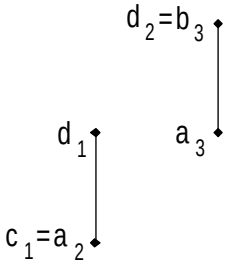
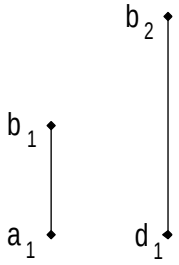
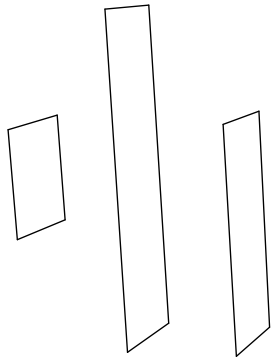
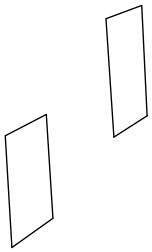
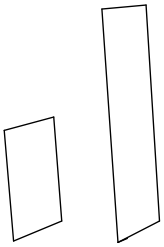
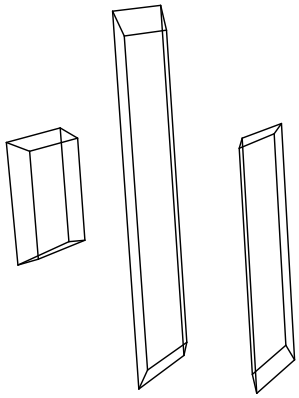
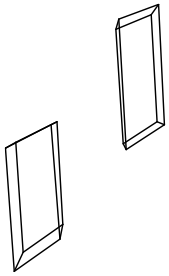
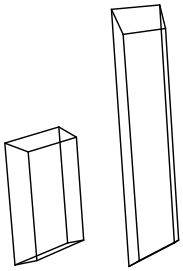
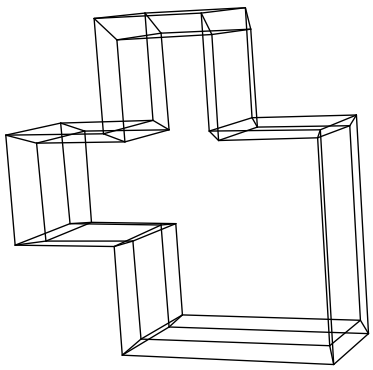
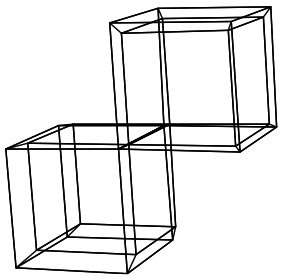
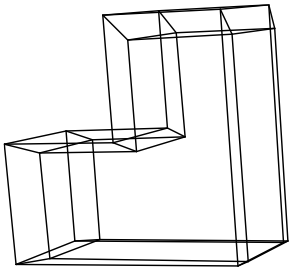
FIGURE 7.15

Two 4D-OPP's (presented in Table 7.6) with common interior regions (a). b) Their 3D sections (two of them have common interiors). c) The 2D sections from the 3D sections. d) The 1D sections from the 2D sections (own elaboration).

Since the segments in **Figure 7.15.d** represent the basic case for the regularized Boolean operations between the 4D-OPP's A and B (of **Figure 7.15.a**), it must be applied the corresponding operator. We will exemplify the operations of union, intersection and difference. In the **Table 7.7** are shown the results of these operations. **Table 7.7**'s columns 1, 2 and 3 corresponds to $A \cup^* B$, $A \cap^* B$ and $A -^* B$ respectively. The Boolean operations between 1D sections are performed according to **Table 7.5**. The resultant 1D sections will define 2D rectangular sections which in turn define the three or two (according to the operation) 3D sections of the resultant 4D-OPP.

TABLE 7.7

Boolean Operations Between 1D Sections of two 4D-OPP's (whose relative positions are shown in Figure 7.15.a) and the resultant 4D-OPP's (see text for details; own elaboration).

$A \cup^* B$	$A \cap^* B$	$A -^* B$
		
		
		
		

7.3 Application 1: Handling and Processing Animation Frames Using the EVM

7.3.1 Black & White 2D Animation Using the EVM-3D

This current application was originally suggested in [Aguilera, 98]. A black & white 2D animation, viewed as a sequence of *n* black & white 2D frames, can be handled as a 3D-OPP in the following way:

a) Let each frame f_k in the animation be coded in the EVM as a 2D-OPP, where the inside of f_k represents the black regions or pixels in the frame; and the outside, the white ones (See in **Figure 7.16.a** an example of a simple 2D black & white animation composed by four frames whose resolution is 9 x 9 pixels. **Figure 7.16.b** shows the result of the same frames considered as 2D-OPP's and their respective extreme vertices).

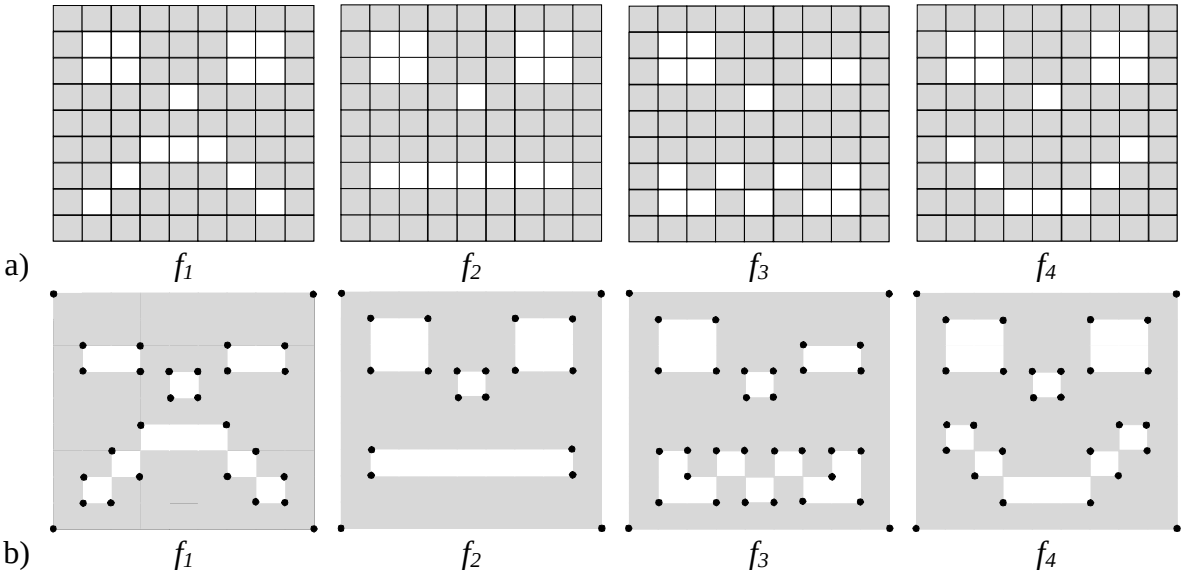


FIGURE 7.16
Example of a simple 2D black & white animation. a) Its frames with 9x9 pixels' resolution. b) The frames represented through 2D-OPP's and their extreme vertices (own elaboration).

b) Let us extrude f_k into the third dimension, and thus obtain a prism $prism_k$ whose base is f_k and its length is proportional to the time f_k is to be displayed. The new dimension will measure and represent the time. See in **Figure 7.17** the extrusion of the frames presented in **Figure 7.16.b**.

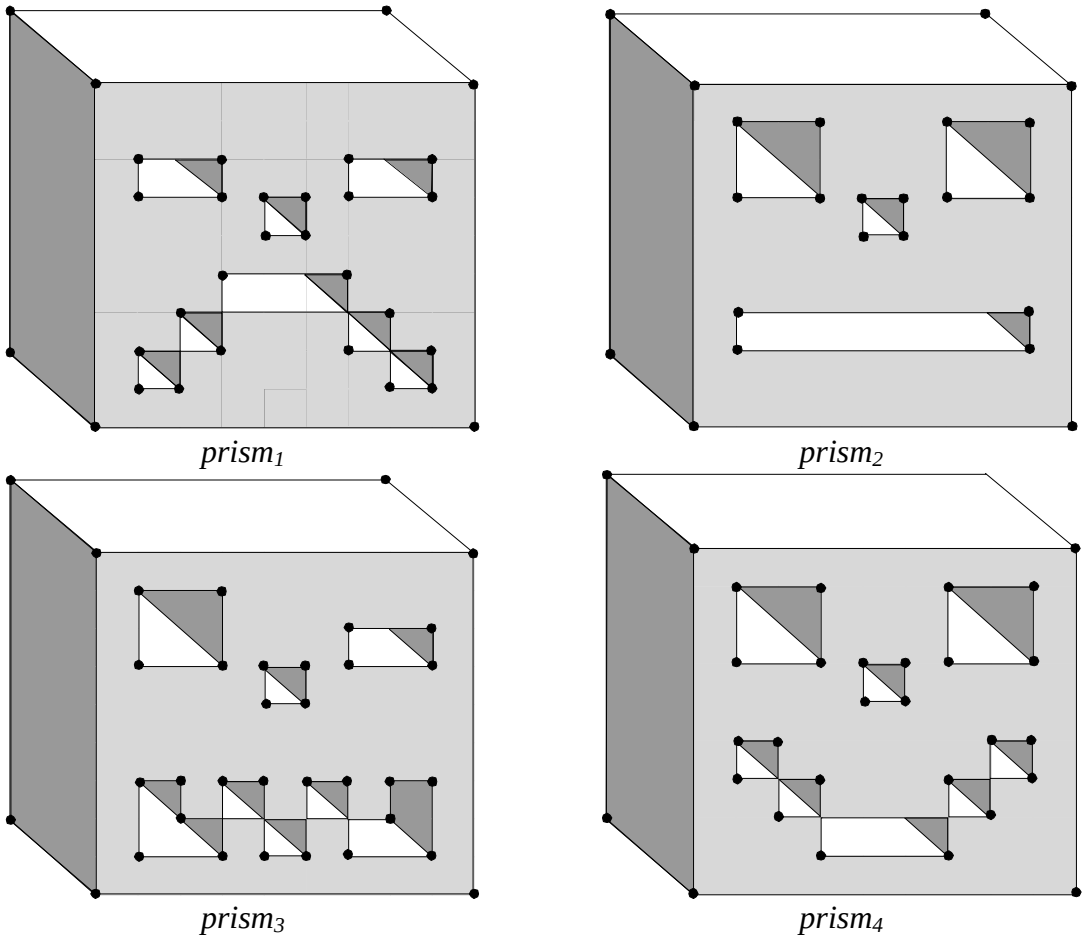


FIGURE 7.17
Extrusion of the frames of an animation (from Figure 7.16) and some of their extreme vertices (own elaboration).

c) Let $p = \bigcup_{k=1}^{n} prism_k$, then p is a 3D-OPP that represents the given 2D animation (see **Figure 7.18**). Due to all the prisms are quasi disjoint 3D-OPP's, then the EVM for p can be obtained by applying [Aguilera, 98]:

$$EVM(p) = \bigotimes_{k=1}^{n} EVM(prism_k)$$

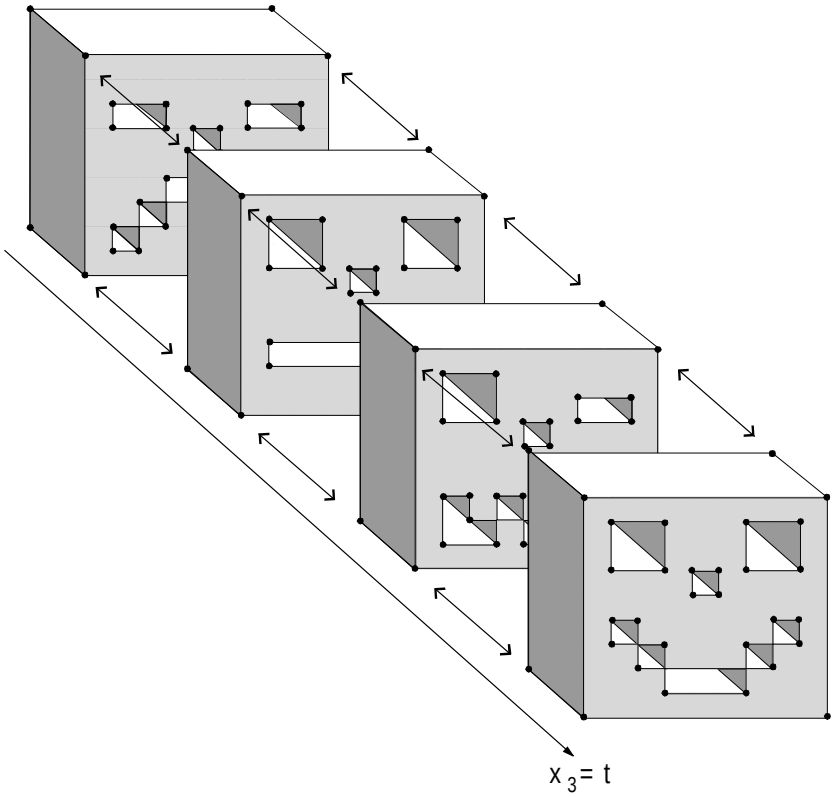


FIGURE 7.18
Composing the 3D-OPP that will represent an animation (from Figure 7.16) as the union of its extruded frames (own elaboration).

Figure 7.19.a shows the Φ's of the 3D-OPP that represents the animation from **Figure 7.16** which are perpendicular to the axis that represent the time. In **Figure 7.19.b** is shown separately each $\Phi_k(p)$.

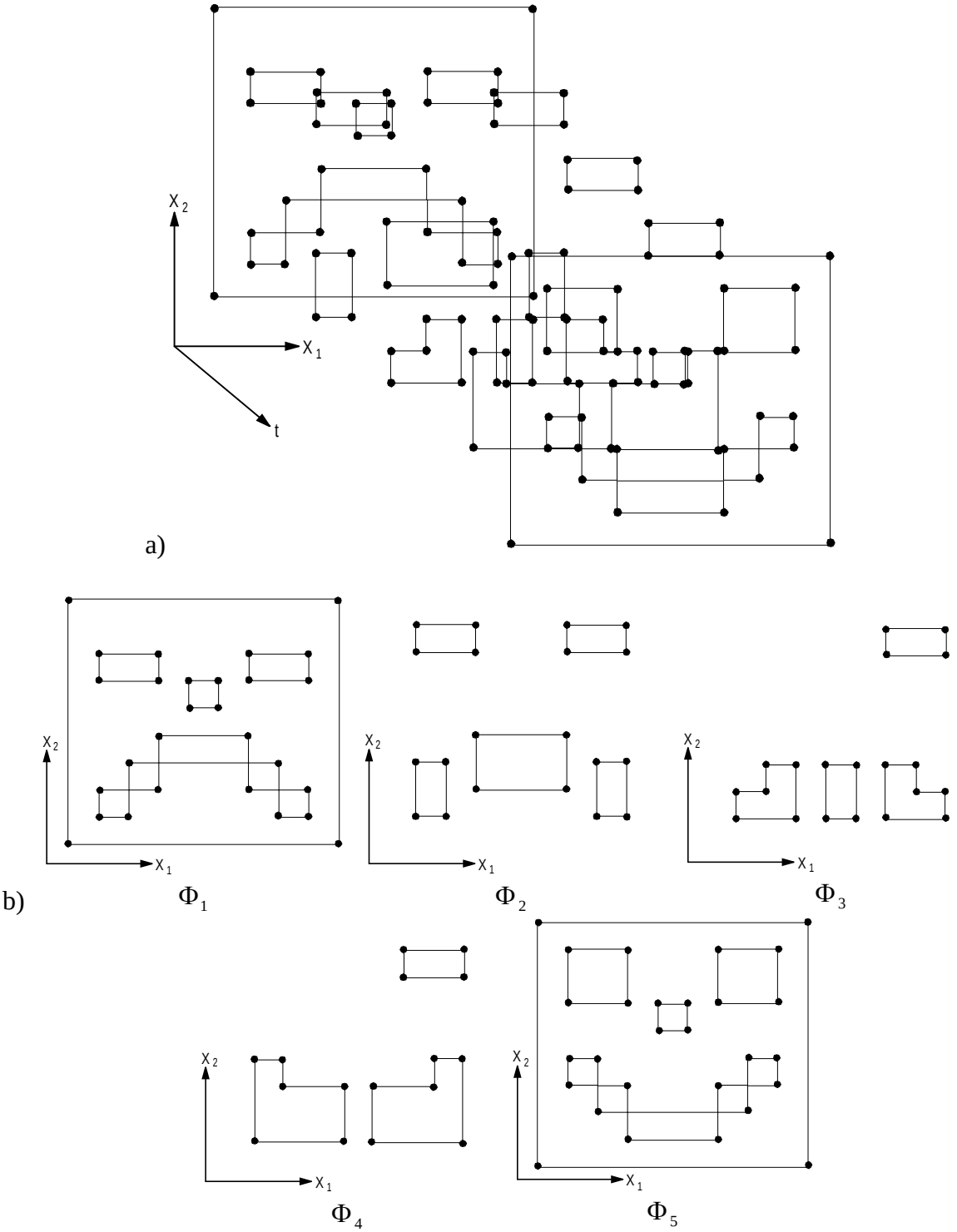


FIGURE 7.19

The Φ's, perpendicular to the corresponding axis for the time, of a 3D-OPP that represents an animation (from Figure 7.16; own elaboration).

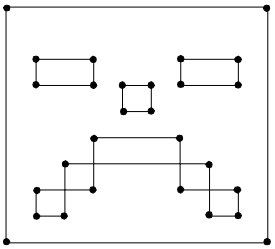
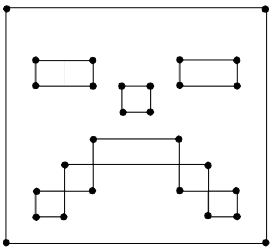
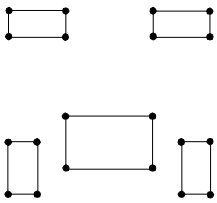
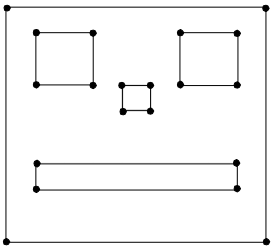
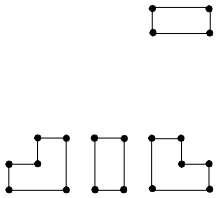
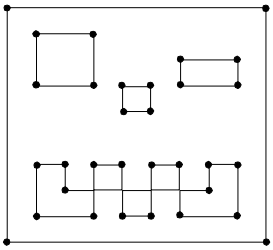
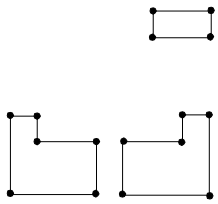
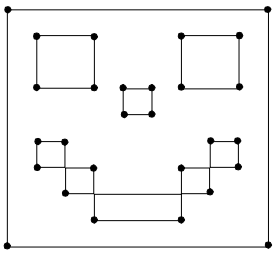
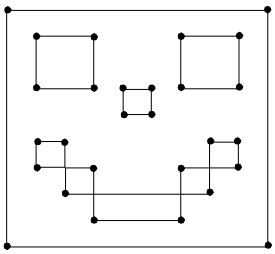
By representing a given 2D animation using a 3D-OPP *p* and the EVM-3D we have the following characteristics:

- The sequence of sections of *p* corresponds to the sequence of frames, i.e., $\overline{S_k(p)} = f_k$.

- Computation of frames: Since $\overline{S_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{\Phi_k(p)}$ (section 7.2.6) then, $EVM(f_k) = EVM(f_{k-1}) \otimes EVM(\Phi_k(p))$, i.e., the black regions at extended faces $\Phi_k(p)$ represent the regions of a previous frame f_{k-1} that need to be modified (changed from black to white, or from white to black) in order to update it to the following frame f_k. **Table 7.8** presents the sequence of the computation of the frames for the animation presented in **Figure 7.16**.

TABLE 7.8

Computing the frames for an animation (from Figure 7.16) represented through a 3D-OPP and the EVM (Own elaboration).

$\overline{S_{k-1}(p)}$	$\overline{\Phi_k(p)}$
$S_0(p) = \varnothing$	 $\overline{\Phi_1(p)}$
 $\overline{S_1(p)} = \overline{S_0(p)} \otimes^* \overline{\Phi_1(p)} = f_1$	 $\overline{\Phi_2(p)}$
 $\overline{S_2(p)} = \overline{S_1(p)} \otimes^* \overline{\Phi_2(p)} = f_2$	 $\overline{\Phi_3(p)}$
 $\overline{S_3(p)} = \overline{S_2(p)} \otimes^* \overline{\Phi_3(p)} = f_3$	 $\overline{\Phi_4(p)}$
 $\overline{S_4(p)} = \overline{S_3(p)} \otimes^* \overline{\Phi_4(p)} = f_4$	 $\overline{\Phi_5(p)}$
$\overline{S_5(p)} = \overline{S_4(p)} \otimes^* \overline{\Phi_5(p)} = \varnothing$	

The managing of a black & white 3D-animation's *n* frames can be performed in analogous way:

- Let each 3D frame f_k in the 3D-animation be coded in EVM as a 3D-OPP, where the inside of f_k represents the black regions or voxels in the 3D frame; and the outside, the white ones.

- Let us extruded f_k into the fourth dimension, and thus obtain a 4D hyperprism $hyperprism_k$ whose base is f_k and its length is proportional to the time f_k is to be displayed.

- Let $p = \bigcup_{\lambda=1}^{n} hyperprism_\lambda$ then *p* is a 4D-OPP that represents the given black & white 3D-animation.

- By representing this 4D-OPP *p* through the EVM-4D we have that the sequence of sections of *p* corresponds to the sequence of 3D frames, i.e, $\overline{S_k(p)} = f_k$.

- Computation of frames: Since $\overline{S_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{\Phi_k(p)}$ (see section 7.2.8) then, $EV(f_k) = EV(f_{k-1}) \otimes EV(\Phi_k(p))$, i.e., the black regions at extended volumes $\Phi_k(p)$ represent the regions of a previous frame f_{k-1} that need to be modified (changed from black to white, or from white to black) in order to update it to the following frame f_k.

7.3.1.1 Collision Detection

This application (which will be explored as part of our future research) was originally proposed in [Zhou, 91]. However, it can be considered under our context in the following way:

If p and q are two (4D) 3D-OPP's representing the black & white animation and/or motion of two (solids) polygons, then these two (solids) polygons collide iff $p \cap q \neq \varnothing$. Furthermore, if $p \cap q \neq \varnothing$ then the time coordinate values of $p \cap q$ indicate the precise instant of the collision (In fact, [Zhou, 91] proposed this application by representing 3D-objects' boundaries through the equations that define the set of points on the surfaces of spheres, ellipsoids, etc.).

7.3.2 Representing Color 2D-Animations Through 4D-OPP's and Their Extreme Vertices

The procedure described in [Aguilera, 98] for processing black & white 2D animations can be directly extended to control colored frames through a 4D-OPP and its extreme vertices. We will label each colored frame in the animation as f_k and n will be the number of such frames. In the **Figure 7.20** an example of a simple color 2D-animation composed by four frames whose resolution is 9 x 9 pixels is shown. In each frame can be identified yellow, red, green and blue regions.

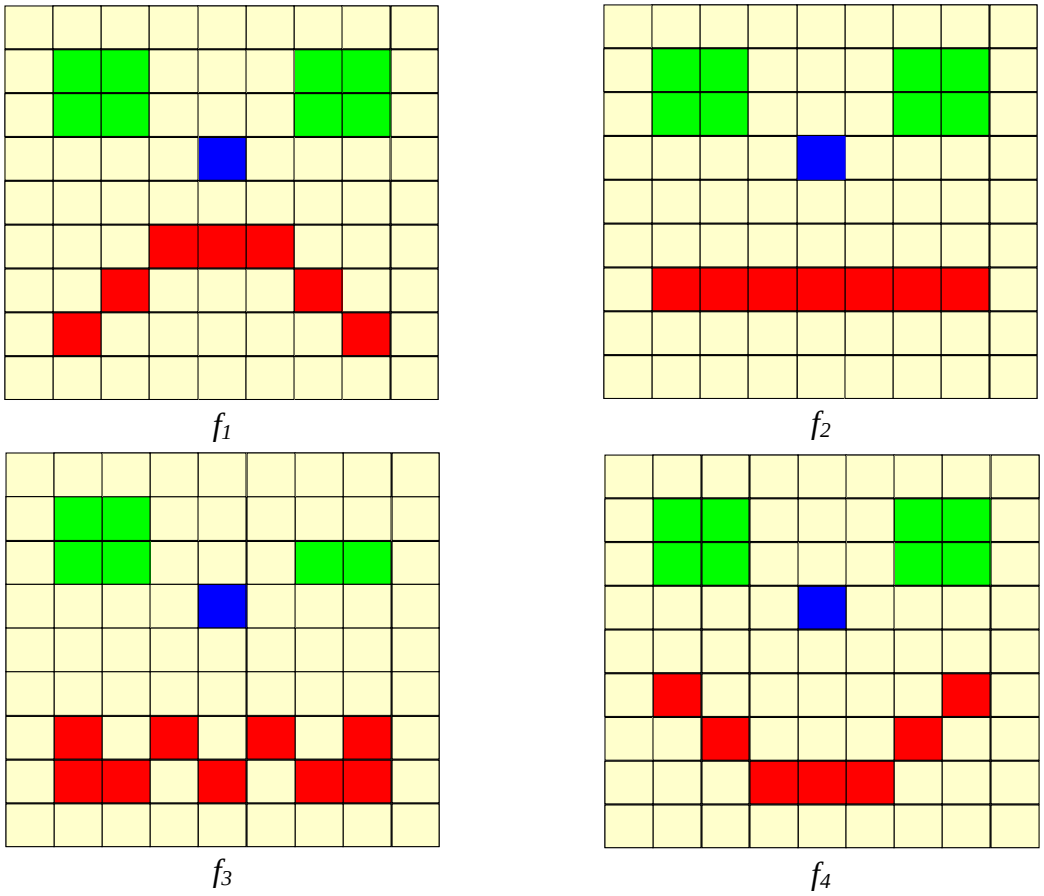


FIGURE 7.20
Example of a simple color 2D-animation (own elaboration).

A color animation can be handled as a 4D-OPP in the following way:

a) The red-green-blue components of each pixel will be integrated into a single value (for example, we can use the procedure defined in [Gosling, 00] to represent the red-green-blue 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 red-green-blue components will now be considered as its X_3 coordinate (coordinates X_1 and X_2 correspond to the original pixels' coordinates). See **Figure 7.21**.

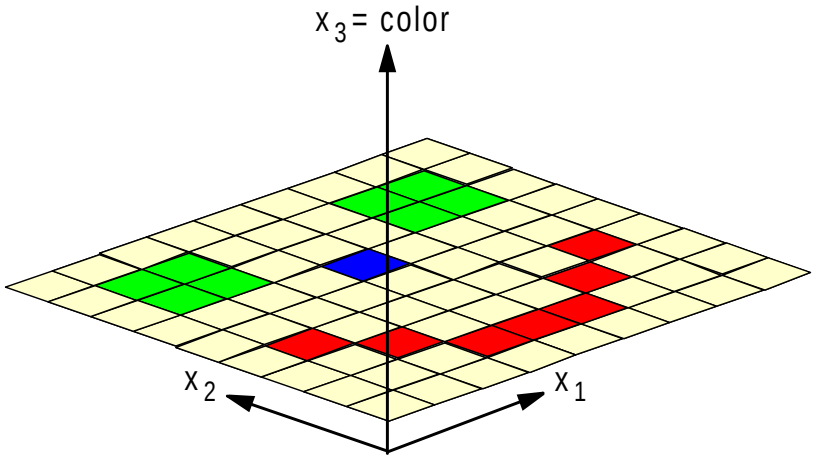


FIGURE 7.21
The 3D space defined for the extrusion of color 2D-pixels (own elaboration).

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 couldn't be extruded and therefore its associated prism (a 3D-OPP) won't be obtained. See in **Figure 7.22** the sets of prisms xf_k which are the result of the extrusion of the frames f_k of the animation presented in **Figure 7.20**.

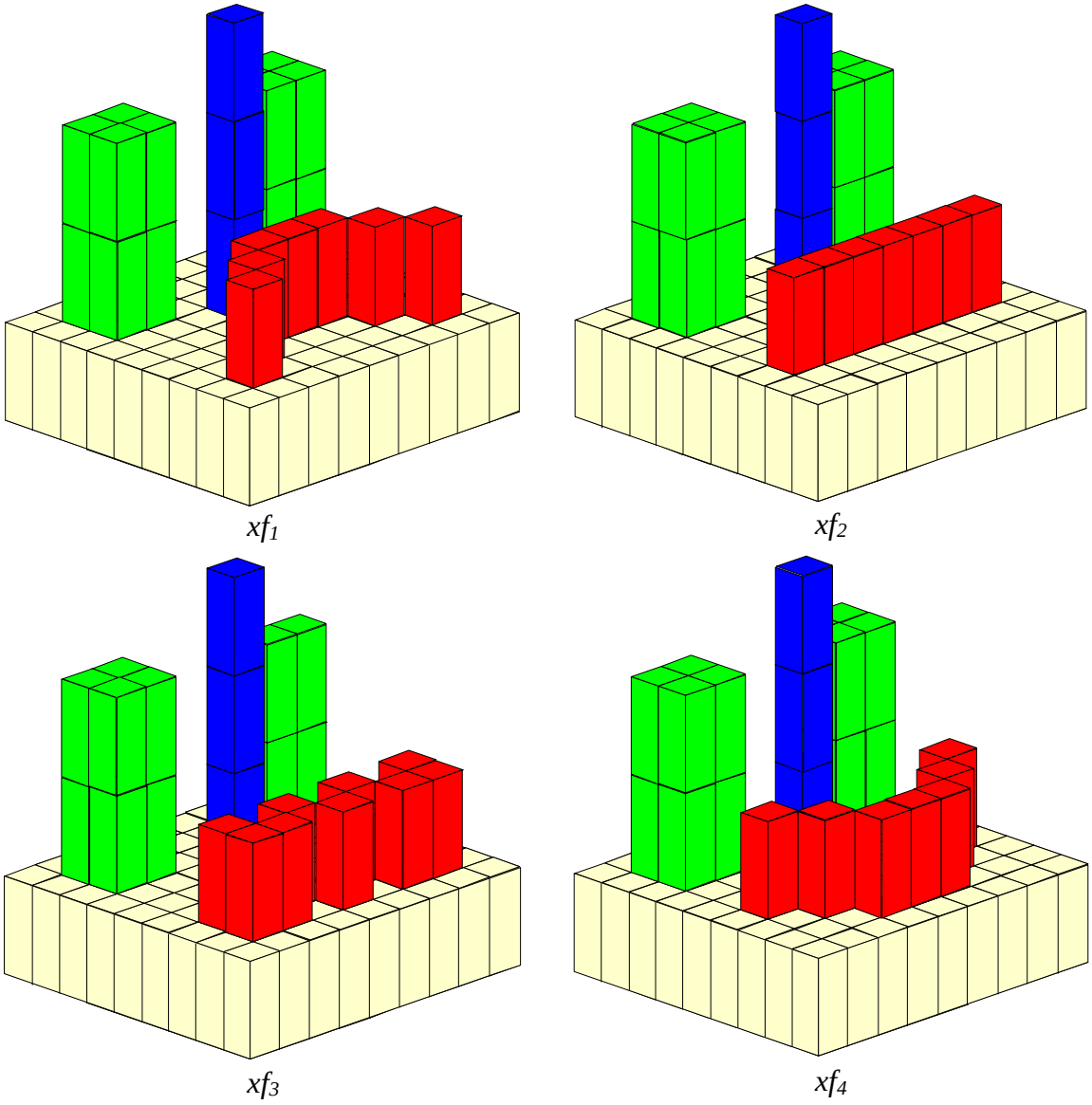


FIGURE 7.22
The sets of prisms which are the result of the extrusion of the frames of an animation (presented in Figure 7.20; own elaboration).

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-OPP's, we can easily obtain the 3D-OPP and its respective EVM of the whole 3D frame by computing the regularized union of all the prisms in xf_k. Then we have to apply (all the vertices in a $prism_i$ are extreme):

$$EVM(F_k) = \bigotimes_{i=1}^{npr} EVM(prism_i \in xf_k)$$

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

In the **Figure 7.23** are shown the 3D frames F_k from the animation presented in **Figure 7.20**).

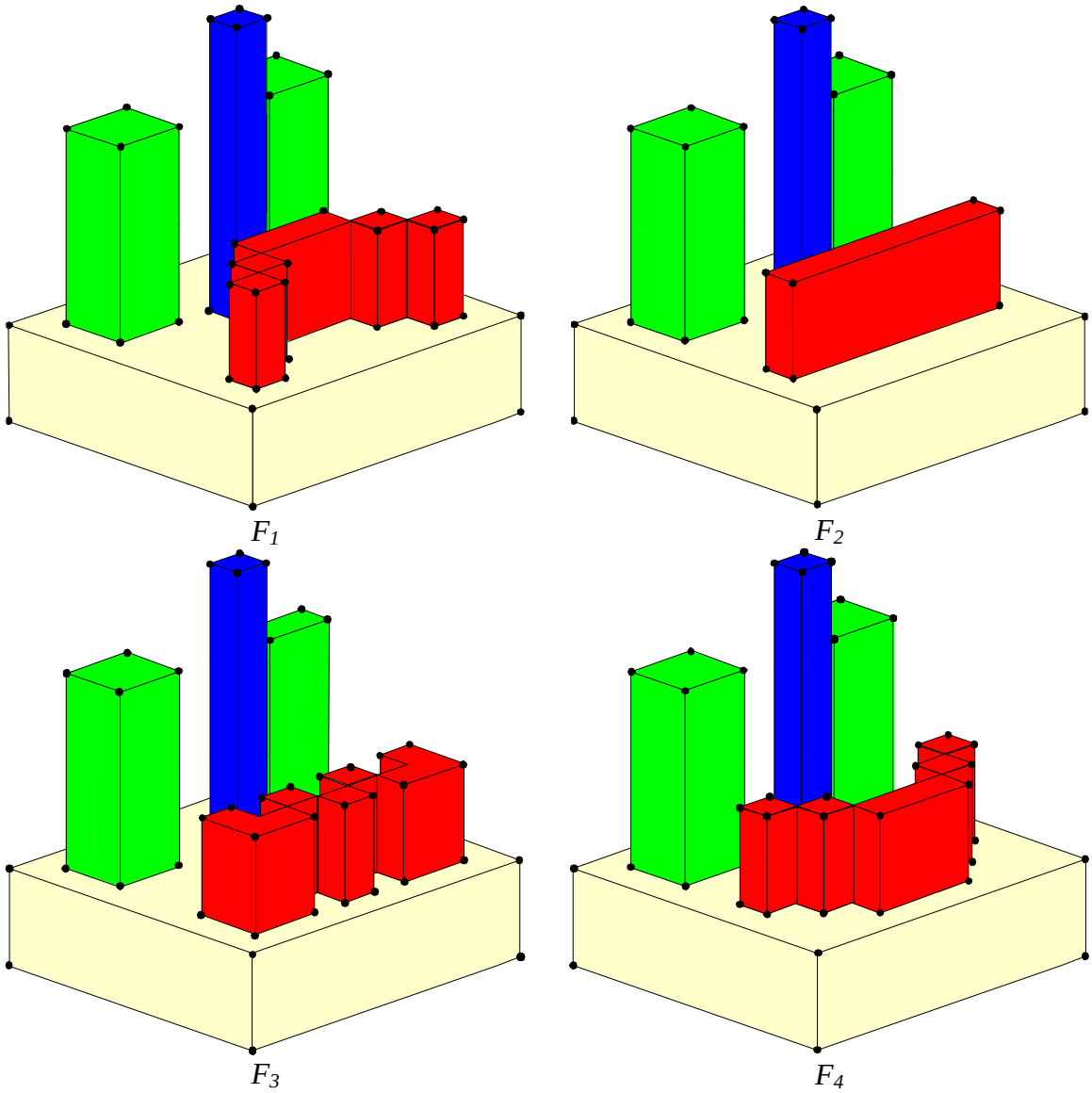


FIGURE 7.23
The 3D frames that represent a 2D colored animation
(presented in Figure 7.20. Some of their extreme vertices are shown; own elaboration).

c) Let us extrude F_k into the fourth dimension, and thus obtain a 4D hyperprism $hyperprism_k$ 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 in **Figure 7.24** the process of extrusion of the 3D frame F_1 presented in **Figure 7.23.**

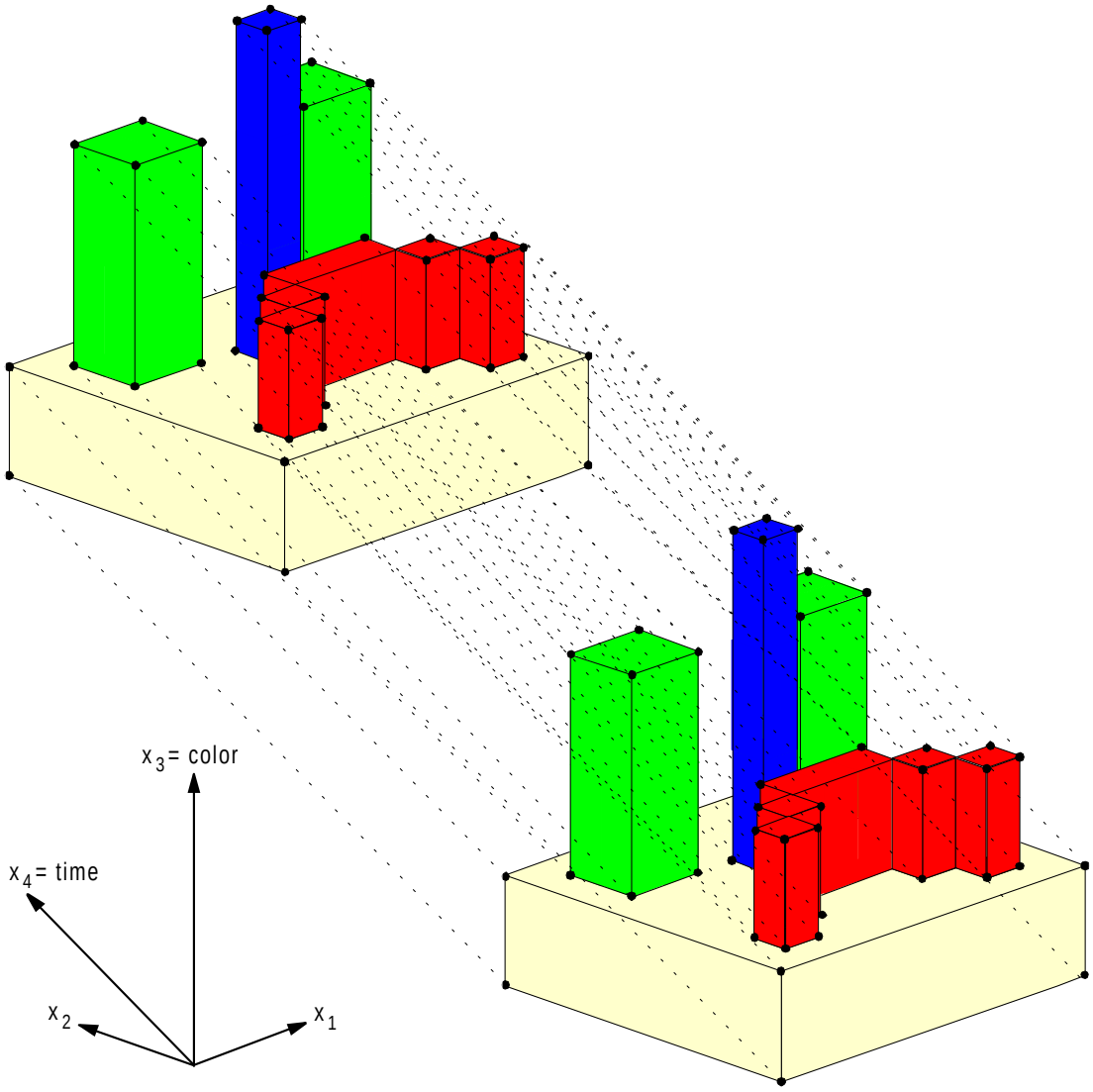


FIGURE 7.24
The process of extrusion of a 3D frame (F_3, presented in Figure 7.23) in order to obtain a $hyperprism_1$ (some of its extreme vertices are shown; own elaboration).

d) Let $p = \bigcup_{k=1}^{n} hyperprism_k$, then p is a 4D-OPP that represents the given color 2D-animation. Due to all the n hyperprisms are quasi disjoint 4D-OPP's, then the *EV(p)* for p can be obtained by applying:

$$EV(p) = \bigotimes_{k=1}^{n} EV(hyperprism_k)$$

In the **Figure 7.25** are shown the $\Phi_k(p)$'s of the 4D-OPP p that represents the animation from **Figure 7.20** which are perpendicular to the axis that represent the time.

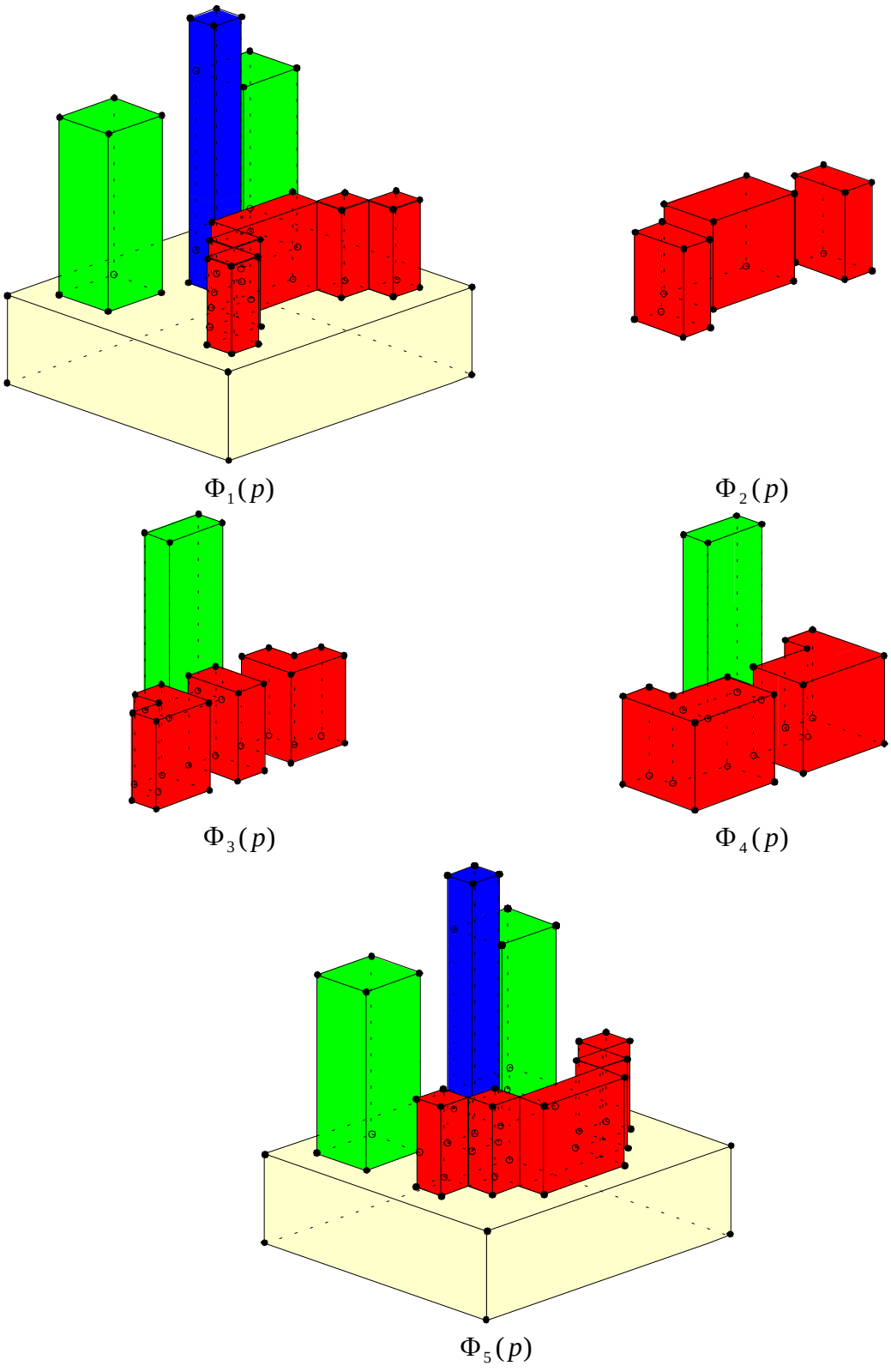


FIGURE 7.25

The extended volumes of the 4D-OPP *p* that represents a color 2D-animation (from Figure 7.20. Their extreme vertices are shown. Own elaboration).

By representing a given color 2D-animation using a 4D-OPP *p* and its *EV(p)* we have the following characteristics:

- The sequence of sections of *p* corresponds to the sequence of 3D frames, i.e., $\overline{S_k(p)} = F_k$.

- Computation of 3D frames: Since $\overline{S_k(p)} = \overline{S_{k-1}(p)} \otimes *\overline{\Phi_k(p)}$ (see section 7.2.8) then $EV(F_k) = EV(F_{k-1}) \otimes EV(\Phi_k(p))$.

- Displaying the 2D colored animation: Each extended face perpendicular to the X_3 axis of 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 red-green-blue components. In the **Figure 7.26** is presented the sequences of extended faces of the 3D frames F_k for the 2D animation presented in **Figure 7.20**.

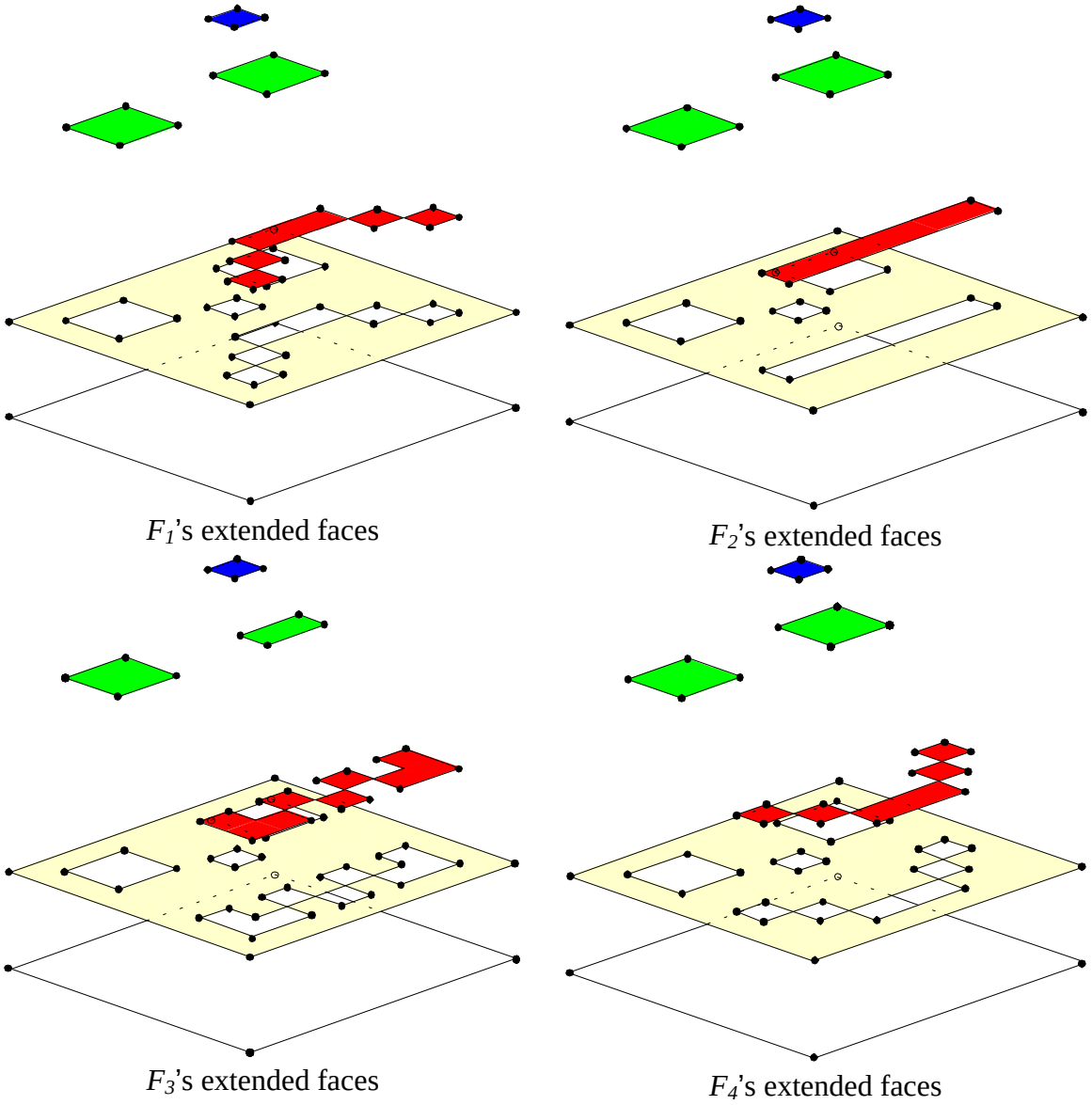


FIGURE 7.26
The sequences of extended faces (the polygons to display) of the 3D frames that represent a color 2D-animation (from Figure 7.20; own elaboration).

Another application to explore and to analyze in our future research is the managing of a color 3D-animation's *n* frames, which can be performed in analogous way (we assume that each 3D frame is defined through a voxelization, see section 6.3.2.2):

- The red-green-blue components of each voxel will be integrated into a single value. Each voxel will now be extruded towards the fourth dimension where the value

integrating its red-green-blue components will now be considered as its X_4 coordinate (coordinates X_1, X_2 and X_3 correspond to the original voxels' coordinates). Let us call xf_k to the set composed by the 4D hyperprisms (the extruded voxels) of each extruded frame f_k.

- Let pr_i be a 4D hyperprism in xf_k and *npr* the number of prisms in that set. Since all the hyperprisms in xf_k are quasi disjoint 4D-OPP's, we can easily obtain the 4D-OPP and its respective extreme vertices of the whole 4D frame by computing the regularized union of all the hyperprisms in xf_k. Then we have to apply (see section 7.2.8):

$$EV(F_k) = \bigotimes_{i=1}^{npr} EV(pr_i \in xf_k)$$

  Where F_k is the 4D frame (a 4D-OPP) that represents the union of all the hyperprisms in xf_k.

- Let us extrude F_k into the fifth dimension, and thus obtain a 5D hyperprism $hyperprism_k$ whose bases are F_k and its length is proportional to the time f_k is to be displayed. The new fifth dimension will measure and represent the time.

- Let $p = \bigcup_{k=1}^{n} hyperprism_k$, then p is a 5D-OPP that represents the given color 3D-animation. Since all the *n* hyperprisms are quasi disjoint 5D-OPP's, then the *EV(p)* for p can be obtained by applying:

$$EV(p) = \bigotimes_{k=1}^{n} EV(hyperprism_k)$$

- The sequence of sections of p corresponds to the sequence of 4D frames, i.e., $\overline{S_k(p)} = F_k$.

- Computation of 4D frames: Since $\overline{S_k(p)} = \overline{S_{k-1}(p)} \otimes^* \overline{\Phi_k(p)}$ (see section 7.2.8) then $EV(F_k) = EV(F_{k-1}) \otimes EV(\Phi_k(p))$.
- Displaying the 3D colored animation: Each extended volume perpendicular to the X_4 axis of each 4D frame F_k contains the voxels to display. The colors to apply to those voxels are referred through the X_4 coordinate that contains the integrated red-green-blue components.

7.4 Application 2: Comparing Color 2D-Images Through Their Extrusions to the 5D Colorspace

The topic related to comparing color 2D-images has been widely considered in several works by proposing specific methods to achieve this process, see for example [Huttenlocher, 93], [Pass, 96] or [Jurisica, 00]. We propose now a method for comparing color 2D-images which can be resumed in the following way:

a) Extruding color 2D-images towards the 5D colorspace (section 7.4.1).
b) Computing the 5D hypervolume of extruded images (section 7.4.2).
c) Determining if two color 2D-images are *"initially similar"* (section 7.4.3).
d) Computing the intersection between two extruded images (section 7.4.4).
e) Determining if two color 2D-images are similar (section 7.4.5).

Finally, in section 7.4.6 is presented the algorithm to perform the proposed comparison method and an application is mentioned.

7.4.1 Extruding color 2D-images towards the 5D colorspace

The color 2D-images are extruded towards the 5D colorspace: where X_1, X_2, X_3, X_4 and X_5 coordinates correspond to the pixels' values x_1, x_2, R (the red component), G (the green component) and B (the blue component), respectively [Duffin, 94]. By this way the extrusion of each pixel will be a 5D hyperprism h_j and n will indicate the total number of hyperprisms obtained for a color 2D-image. As mentioned in previous section, we have to avoid zero values for components R, G and B in order to obtain for each pixel its corresponding 5D hyperprism.

7.4.2 Computing the 5D hypervolume of extruded images

Let H the set of 5D hyperprisms for a color 2D-image. Now, we will compute the total 5D hypervolume HV of this set, i.e. the sum of the 5D hypervolume of each one of its hyperprisms:

$$HV = \sum_{i=1}^{n} hypervolume(h_i \in H)$$

The hypervolume of a 5D hyperprism can be easily computed through the product of its values $x_1Side * x_2Side * R * G * B$, where $x_1Side = x_2Side = 1$ correspond to the dimensions of a pixel in the original 2D-image.

7.4.3 Determining if two color 2D-images are *"initially similar"*

Let H_a and H_b be the corresponding sets of 5D hyperprisms for two color 2D-images *a* and *b* with their respective computed 5D hypervolumes HV_a and HV_b. If we assume that the color components R, G and B are inside the range [1 – 256] (where 1 indicates the least intensity), then we can expect that the hyperprism of a white pixel (R=256, G=256, B=256) will have the maximum 5D hypervolume (in fact $256^3\ u^5$), while the hyperprism of a black pixel (R=1, G=1, B=1) will have the minimum 5D hypervolume ($1^3\ u^5$). If the majority of the pixels of an image are dark then its associated 5D hypervolume will be less than the associated 5D hypervolume of a image whose pixels are lighter and therefore, both images will have numeric differences related with the color of their pixels. These differences can be determined through the computation of the function $Q_{a,b}$ between the total hypervolumes by according to:

$$Q_{a,b}(HV_a, HV_b) = \begin{cases} 1 - \dfrac{HV_a}{HV_b} & if \quad HV_a < HV_b \\ 1 - \dfrac{HV_b}{HV_a} & if \quad HV_b < HV_a \\ 0 & if \quad HV_a = HV_b \end{cases}$$

Let ε_1 be an arbitrary assigned value such that $0 \le \varepsilon_1 \le 1$. Then, we will propose that two images *a* and *b* are *"initially similar"* (because a second comparison will be considered) if the $Q_{a,b}$ of the 5D hypervolumes of the corresponding sets H_a and H_b satisfies the inequality (in fact ε_1 is an allowed difference):

$$Q_{a,b} \le \varepsilon_1$$

For example, consider the images presented in **Figure 7.27** and ε_1 = 0.05. Then HV_a (according to our implementation, see **Appendix F**) is 5,146,844 u^5 (where u^5 stands for 5D hypercubical units) and HV_b is 4,996,787 u^5. Therefore $Q_{a,b}(HV_a,HV_b) \approx 0.029$ and $0.029 \leq 0.05$ which implies that the images are *"initially similar"*.

Image a *Image b*

FIGURE 7.27
Two images classified as *"initially similar"*
(see text for details; images obtained from [Cenapred, 03]).

The images from **Figure 7.28** were classified, according to our proposed procedure, as not *"initially similar"*. Let ε_1=0.05, HV_a is equal to 10,742,439 u^5, HV_b is 9,819,038 u^5 and $Q_{a,b}(HV_a,HV_b) \approx 0.085$. Therefore $0.085 \leq 0.05$ is not true.

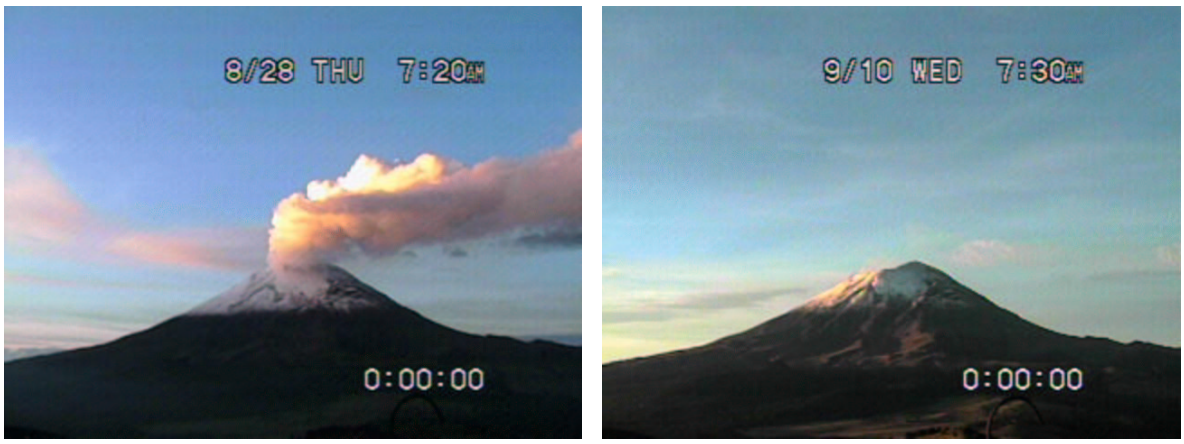


Image a *Image b*

FIGURE 7.28
Two images classified as not *"initially similar"*
(see text for details; images obtained from [Cenapred, 03]).

7.4.4 Computing the intersection between two extruded images

Now we will compute the intersection between H_a and H_b (the corresponding sets of 5D hyperprisms for color 2D-images *a* and *b*) which were classified as *"initially similar"*. If the sets of hyperprisms are represented through an scheme as the EVM-5D then this Boolean operation would be performed through its corresponding algorithm (see section 7.2.8) by intersecting the corresponding 5D hyperprisms. However, this process can be achieved in a very simple way by considering only two points of each 5D hyperprism: one of the points will be $(x_1, x_2, 0, 0, 0)$ while the other will be $(x_1 + 1, x_2 + 1, R, G, B)$. These two points will define a segment which is the main diagonal that connects the bases of a 5D hyperprism.

Let $h_i \in H_a$ and $h_j \in H_b$ be two 5D hyperprisms with the same x_1 and x_2 coordinates. The points' coordinates of the diagonal associated to h_i are then $(x_1, x_2, 0, 0, 0)$ and $(x_1 + 1, x_2 + 1, R_i, G_i, B_i)$; while the points' coordinates of the diagonal associated to h_j will be $(x_1, x_2, 0, 0, 0)$ and $(x_1 + 1, x_2 + 1, R_j, G_j, B_j)$.

The required Boolean operation, intersection, can be performed by selecting only the minimum coordinates of the points $(x_1+1, x_2+1, R_i, G_i, B_i)$ and $(x_1+1, x_2+1, R_j, G_j, B_j)$, that is, we have the new point:

$$(x_1 + 1, x_2 + 1, R_k, G_K, B_K)$$

Where

$$R_k = \min\{R_i, R_j\}$$

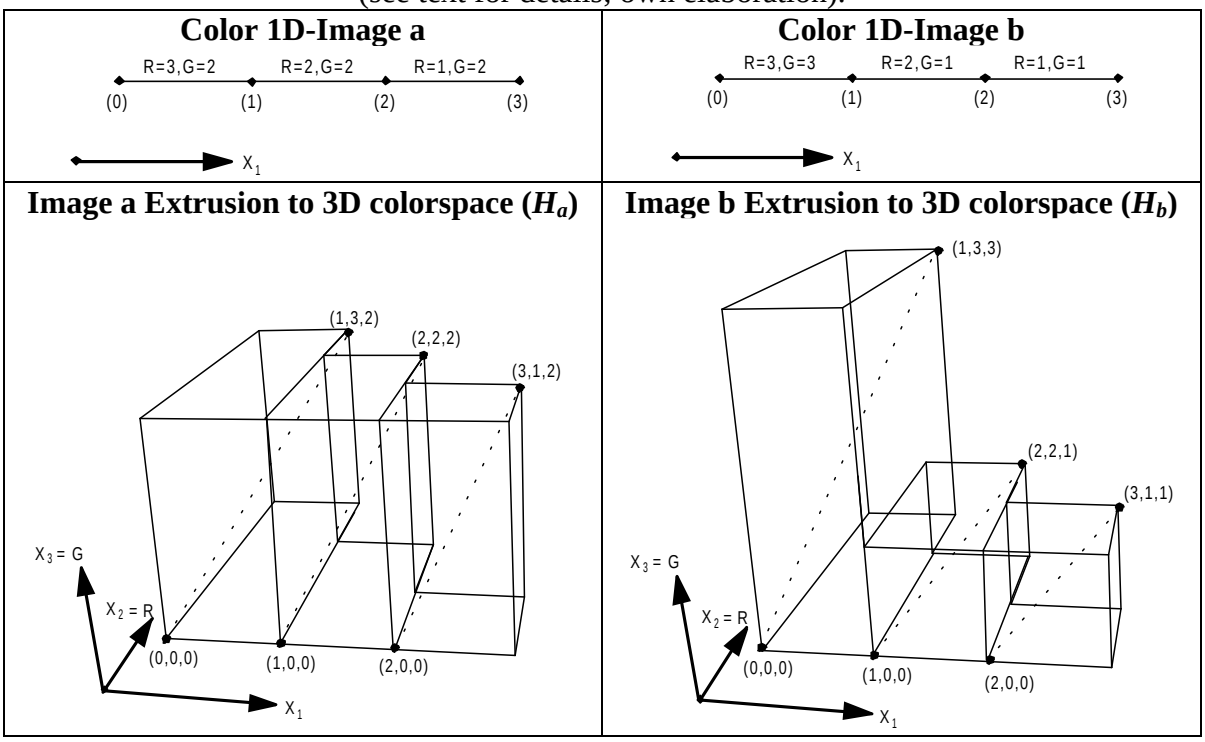
$$G_K = \min\{G_i, G_j\}$$

$$B_K = \min\{B_i, B_j\}$$

The new segment's vertices (x_1, x_2, 0, 0, 0) and (x_1 + 1, x_2 + 1, R_k, G_k, B_k) will correspond to the main diagonal of the 5D hyperprism h_k which is the intersection between 5D hyperprisms h_i and h_j. The final set H_c of hyperprisms h_k will correspond to the intersection between the 5D colorspace's extrusions of image *a* and image *b*.

We will illustrate the above step by considering a simple example of the two color 1D-images presented in **Table 7.9**. We will assume in this case that there are only two color components (R and G) whose values will be in {1, 2, 3}. Moreover, x_1Side will be equal to one. These color 1D-images will be extruded to a 3D colorspace, where X_2 axis will correspond to the red component while X_3 axis will correspond to the green component. The extrusions H_a and H_b (each one with three prisms; the dotted lines indicate their main diagonals) are also shown in **Table 7.9**.

TABLE 7.9
Two color 1D-images and their extrusion to the 3D colorspace
(see text for details; own elaboration).



In the **Table 7.10** are shown the main diagonal's pair of vertices of each one of the prisms in the extrusions H_a and H_b from **Table 7.9**. One of the vertices in a diagonal will have the values corresponding to red and green components equal to zero; the intersection will be performed by considering the opposite vertex. In this case, given two vertices i and j, the new vertex k will be $(x_1 + 1, R_k, G_k)$ where $R_k=\min\{R_i,R_j\}$ and $G_k=\min\{G_i,G_j\}$. Finally, the resulting new main diagonals can be seen in **Table 7.10**'s last column.

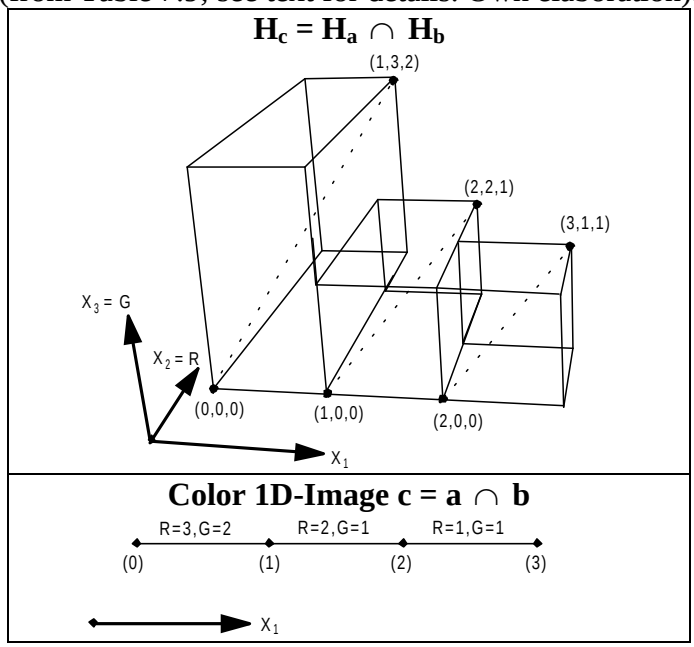
TABLE 7.10
Performing the intersection between prisms in H_a and H_b (from Table 7.9) through their corresponding main diagonals (see text for details; own elaboration).

Pair of Main Diagonals	**Vertices of the Main diagonals' for prisms in H_a**	**Vertices of the Main diagonals' for prisms in H_b**	**$H_a \cap H_b$ (H_c's vertices of the main diagonals)**
1	(0,0,0) – (1,3,2)	(0,0,0) – (1,3,3)	(0,0,0) – (1,3,2)
2	(1,0,0) – (2,2,2)	(1,0,0) – (2,2,1)	(1,0,0) – (2,2,1)
3	(2,0,0) – (3,1,2)	(2,0,0) – (3,1,1)	(2,0,0) – (3,1,1)

Finally, the new diagonals shown in **Table 7.10** will describe to three new prisms that belong to the new set H_c. See **Table 7.11**. These new prisms are the result of the intersection between the corresponding prisms in H_a and H_b. Finally, the prisms in H_c can be intruded (or projected) to get a color 1D-image *c* (shown in **Table 7.11**).

TABLE 7.11

The results of the intersection between the extrusions of two color 1D-images (from Table 7.9; see text for details. Own elaboration).



By reconsidering the application of these procedure over color 2D-images, we have to add that in order to get the resultant color 2D-image of the intersection operation, only we have to consider again its 5D hyperprisms' main diagonals. The main diagonal's first point (x_1, x_2, 0, 0, 0) will indicate the coordinates of the original pixel (obviously x_1 and x_2) while the last three coordinates of the second point ($x_1 + 1$, $x_2 + 1$, R, G, B) will indicate their appropriate color. In the **Figure 7.29** the color 2D-image, that is the result of intersecting the 5D colorspace's extrusions of images presented in **Figure 7.27,** is shown.

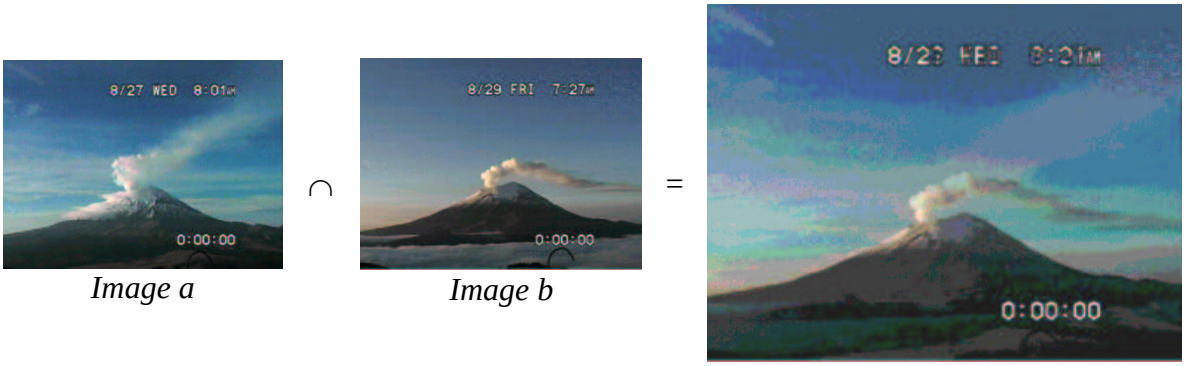


Image a ∩ *Image b* =

FIGURE 7.29
Computing the intersection between the 5D colorspace's extrusions of two color 2D-images *"initially similar"* (from Figure 7.27; images *a* and *b* obtained from [Cenapred, 03]; own elaboration).

7.4.5 Determining if two color 2D-images are similar

We will compute the 5D hypervolume HV_c (according to step 2) of the set of prisms which are the result of the intersection between H_a and H_b (the 5D extrusions of the images being compared). The intersection between H_a and H_b will imply that the set of prisms H_c is composed by the 5D hypervolume that is common to H_a and H_b. Obviously there is 5D hypervolume of H_a not included in H_c and there is 5D hypervolume of H_b not included in H_c. We will compute the proportion of the 5D hypervolume that belongs to H_a but not included in H_c by the following function:

$$Q_{a,c} = 1 - HV_c / HV_a$$

In a similar way, the proportion of the 5D hypervolume that belongs to H_b but not included in H_c can be computed by:

$$Q_{b,c} = 1 - HV_c / HV_b$$

Let ε_a be an arbitrary assigned value such that $0 \leq \varepsilon_a \leq 1$. ε_a will indicate the allowed proportion of 5D hypervolume of H_a that is not included in H_c. In a similar way, let ε_b be an arbitrary value such that $0 \leq \varepsilon_b \leq 1$ where ε_b will indicate the allowed proportion of 5D hypervolume of H_b not included in H_c. We will assume that two images *a* and *b* are similar if their $Q_{a,c}$ and $Q_{b,c}$ satisfy both inequalities:

$$Q_{a,c} \leq \varepsilon_a$$

$$Q_{b,c} \leq \varepsilon_b$$

For example, consider the images and their intersection presented in **Figure 7.29.** Since step c, we presented that HV_a = 5,146,844 u^5 (u^5 stands for 5D hypercubical units) and HV_b = 4,996,787 u^5. The 5D hypervolume HV_c of the intersection between H_a and H_b is 3,744,778 u^5. Then $Q_{a,c} \approx 0.272$ and $Q_{b,c} \approx 0.25$. Let $\varepsilon_a = \varepsilon_b = 0.30$, then we have, that through our procedure, both $Q_{a,c} \leq \varepsilon_a$ and $Q_{b,c} \leq \varepsilon_b$ are satisfied. Therefore, color 2D images *a* and *b* in **Figure 7.30** are classified as similar.

Image a *Image b*

FIGURE 7.30
Reproduction of Figure 7.27 (images *a* and *b* obtained from [Cenapred, 03]).

7.4.6 The Algorithm and Application

The whole proposed procedure (sections 7.4.1 to 7.4.5) for comparing two color 2D-images can be resumed through the following algorithm:

```
Boolean imagesAreSimilar(Image a, Image b, float e_1, float e_a, float e_b)
{
    // We get the sets of 5D hyperprisms (extruded pixels) for images a and b.
    Set Ha = getExtrudedPixelsFromImage(a);
    Set Hb = getExtrudedPixelsFromImage(b);

    // We calculate the 5D hypervolumes of the sets of the hyperprisms in Ha and Hb.
    Integer Hva = calculateHypervolume(Ha);
    Integer Hvb = calculateHypervolume(Hb);

    /* We calculate the numeric difference between the 5D hypervolumes of the sets of
       hyperprisms. */
    float Qab = calculateQab(Hva, Hvb);

    /* If the numeric difference is less or equal than the allowed difference indicated by
       the input value e_1 then the images are "initially similar". */
    If(Qab <= e_1)
    {
        /* We get the set of 5D hyperprisms which is the intersection between the
           5D hyperprisms in Ha and Hb. */
        Set Hc = intersection(Ha, Hb);

        /* It is calculated the 5D hypervolume of the intersection between the sets of
           hyperprisms Ha and Hb. */
        Integer Hvc = calculateHypervolume(Hc);

        // It is calculated the proportion of hypervolume in Ha not included in Hc.
        float Qac = calculateQac(Hva, Hvc);

        // It is calculated the proportion of hypervolume in Hb not included in Hc.
        float Qbc = calculateQbc(Hvb, Hvb);

        // If both proportions of hypervolume not included in Hc are less or equal
           than the allowed proportion indicated by input values e_a and e_b then */
        if((Qac <= e_a) && (Qbc <= e_b))

            return true; // The images are similar.

        return false; // The images are not similar.
    }
    return false; // The images are not "initially similar".
}
```

The above proposed method for comparing images has been used in an experimental application related to Popocatépetl volcano (located in the limits of Puebla state in México; and active and under monitoring since 1997) in order to evaluate its fumaroles under the context of Image Based Reasoning [Jurisica, 00]. See **Appendix F** for more details.