

Chapter 7

Applications

In this chapter we will describe five applications which were developed in the context of the EVM. In **Section 7.1** we will describe the managing of Color 2D-Animations through the EVM. We will see, in **Section 7.1.1**, how starting from this procedure we can develop analogous mechanisms for representing and controlling Color 3D-Animations. In **Section 7.2** we will describe our procedure for comparing color 2-Dimensional images through their extrusions to the 5-Dimensional colorspace. In **Section 7.3** we will discuss some proposed extensions to the paradigm of Image Based Reasoning, originally proposed by Jurisica & Glasgow, by taking in account the application presented in **Section 7.2** and the adding of an indexing scheme and a new similarity metric both based in the nD-EVM. In the **Section 7.4** we will discuss some results related to the conversion from voxelizations to our specific implementation of nD-EVM when $n = 3$. Such voxelizations correspond to “real world” 3D datasets taken from the MoViBio Research Group [MoViBio06], and the University of Tübingen’s Project VolRen [VolRen06]. Finally, in **Section 7.5** we will present the application of the nD-EVM and its algorithms in order to propose an implementation that provides a solution to the problem of the collision detection between 3D objects under Space-Time geometry.

7.1. Application 1: Representing Color 2D-Animations through 4D-OPP's and the EVM

The procedure described in [Aguilera98] for processing black & white 2D animations can be directly extended to control colored frames through a 4D-OPP represented through the EVM. In **Section 5.5.3** are given some details about the processing of black & white 2D-animations through the EVM. In the **Figure 7.1** 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 yellow, red, green and blue 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.

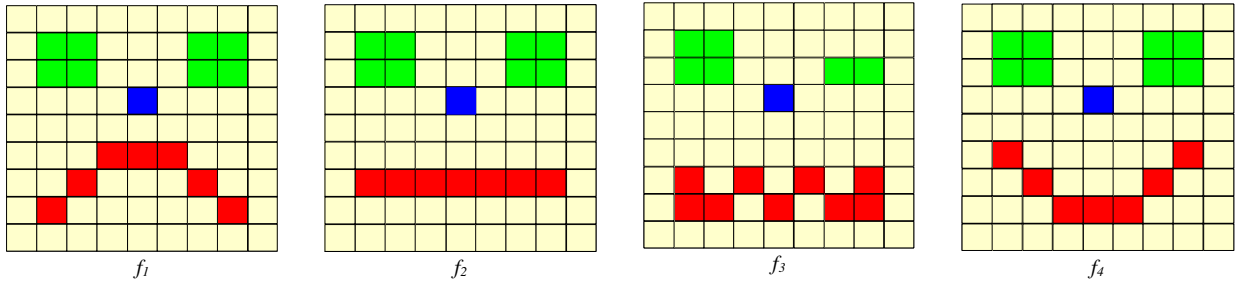


Figure 7.1. Example of a simple color 2D-animation.

- A color animation can be handled as a 4D-OPP in the following way [Pérez-Aguila03d]:
- The red-green-blue components of each pixel will be integrated into a single value. Such value represents 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 7.2**.

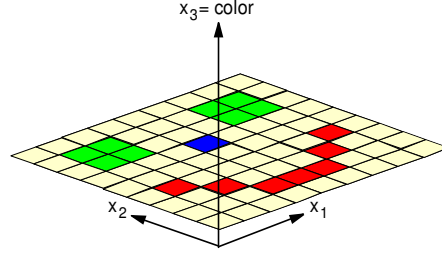


Figure 7.2. The 3D space defined for the extrusion of color 2D-pixels.

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) won't be obtained. See in **Figure 7.3** the sets of prisms xf_k which are the result of the extrusion of the frames f_k of the animation presented in **Figure 7.1**.

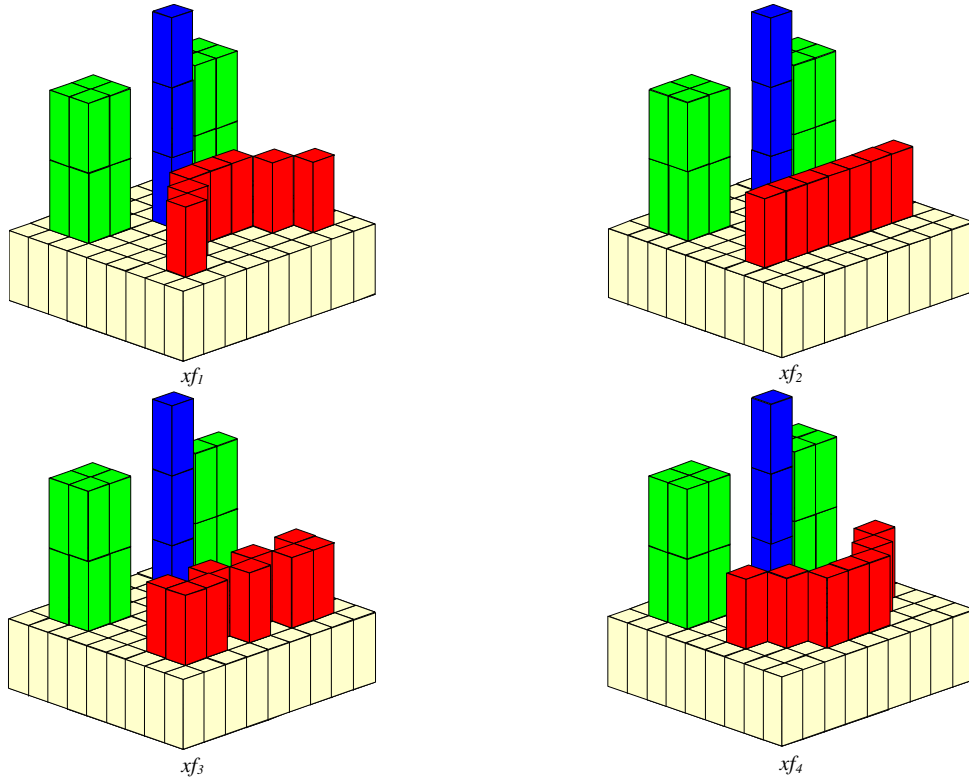


Figure 7.3. The sets of prisms which are the result of the extrusion of the frames of an animation (presented in **Figure 7.1**).

- 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 final 3D-OPP and its respective 3D-EVM of the whole 3D frame by computing the regularized union of all the prisms in xf_k . Then, according to **Corollary 5.9**, we have to apply (all the vertices in a $prism_i$ are extreme):

$$EVM_3(F_k) = \bigotimes_{i=1}^{npr} EVM_3(prism_i \in xf_k)$$

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

In the **Figure 7.4** are shown the 3D frames F_k from the animation presented in **Figure 7.1**.

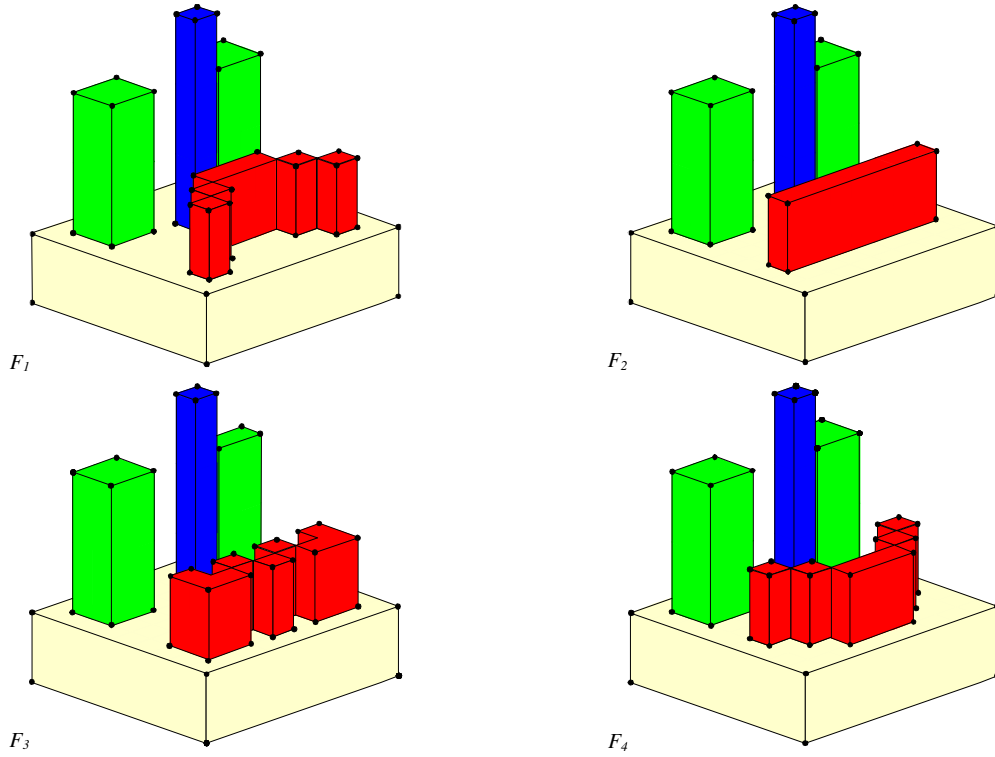


Figure 7.4. The 3D frames that represent a 2D colored animation (presented in **Figure 7.1**. Some of their extreme vertices are shown).

- 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 7.5**.

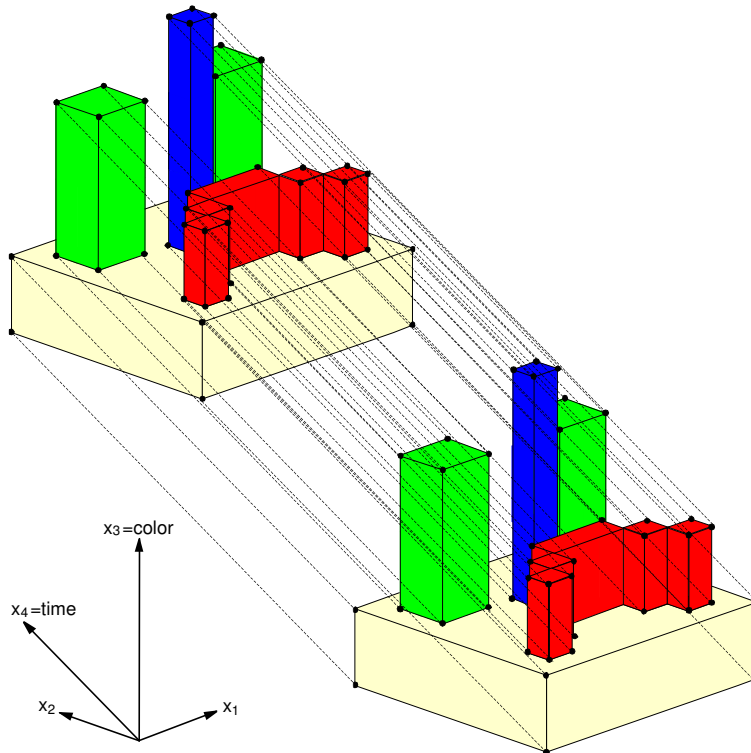


Figure 7.5. The process of extrusion of a 3D frame in order to obtain a *hyperprism* (some of its extreme vertices are shown).

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

$$EVM_4(p) = \bigotimes_{k=1}^m EVM_4(\text{hyperprism}_k)$$

In the **Figure 7.6** are shown the couplets perpendicular to the axis that represent the time, of the 4D-OPP p that represents the animation from **Figure 7.1**.

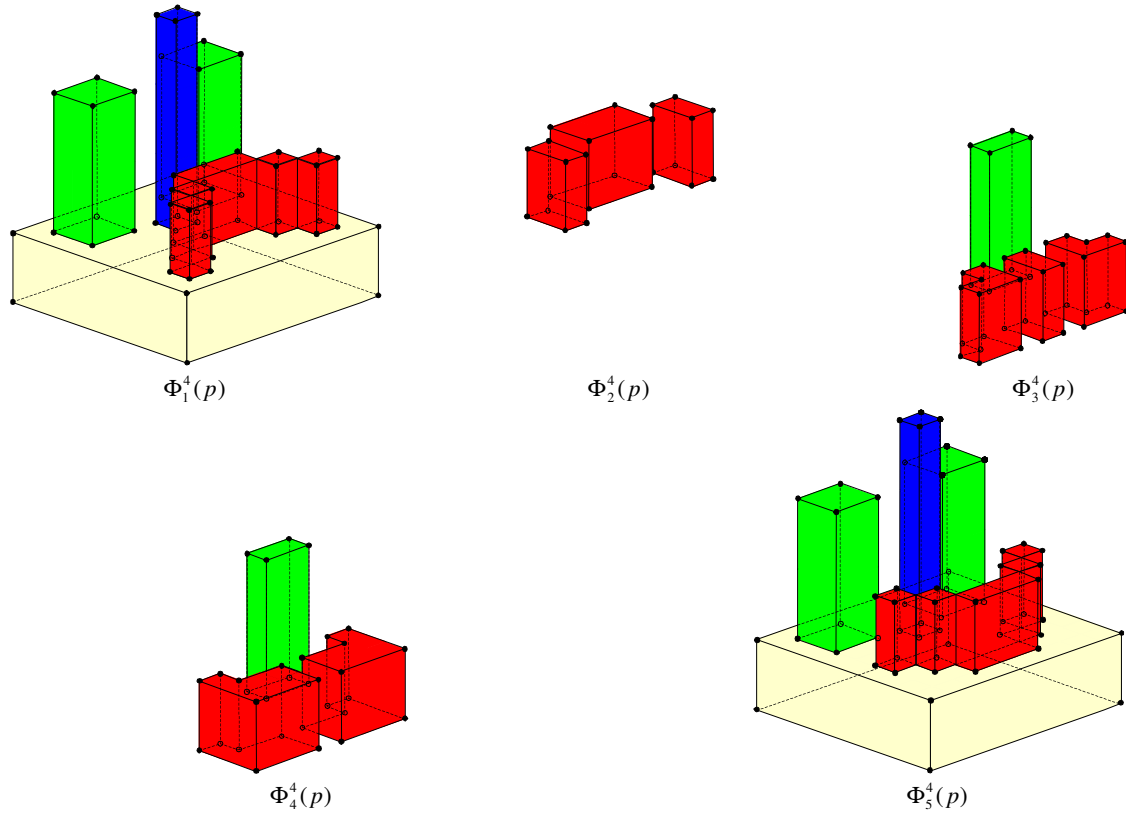


Figure 7.6. The 3D couplets of the 4D-OPP p that represents a color 2D-animation (from **Figure 7.1**. Their extreme vertices are shown).

The **Algorithm 7.1** shows the procedure for converting a set of frames in an animation to a 4D-OPP that codifies it. Such 4D-OPP is represented through a 4D-EVM.

Input: A sequence of frames associated to a color 2D animation.

Output: The 4D-EVM corresponding to the polytope that codifies frames in the input animation.

Procedure GenerateEVM-movie(Movie animation)

```

    EVM evmMovie          // The EVM that will store and codify the input animation.
    EVM Fcurr, Fprev
    real t                // The amount of time that current processed frame is displayed.
    Fprev = InitEVM( )
    for each frame in animation do
        Frame f = animation.nextFrame( )
        t = animation.getDisplayingTime( )
        // Frame f is extruded towards 3rd dimension and its 3D-EVM is computed.
        Fcurr = GetEVMfromExtrudedFrame(f)
        // We perform the Xor between the current and previous 3D frames.
        hvl = MergeXor(Fcurr, Fprev)
        // Amount of time t associated to frame Fcurr is attached to the current 3D couplet.
        SetCoord(hvl, t)
        // A new 3D couplet is attached to the polytope that codifies the input animation.
        PutHvl(hvl, evmMovie)
        Fprev = Fcurr
    end-of-for
    return evmMovie
end-of-procedure

```

Algorithm 7.1. Codifying a Color 2D-animation through a 4D-OPP and the EVM.

By representing a given color 2D-animation using a 4D-OPP p and its 4D-EVM we have the following characteristics [Pérez-Aguila03d]:

- The sequence of the projections of sections in p corresponds to the sequence of 3D frames, i.e., $\pi_4(S_k^4(p)) = F_k$.
- Computation of 3D frames: From **Theorem 5.17** we have that $\pi_4(S_k^4(p)) = \pi_4(S_{k-1}^4(p)) \otimes^* \pi_4(\Phi_k^4(p))$. Because p is expressed through the EVM then by **Corollary 5.8** the 3D-EVM of the frame F_k is computed by $EVM_3(F_k) = EVM_3(F_{k-1}) \otimes EVM_3(\pi_4(\Phi_k^4(p)))$.
- 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 red-green-blue components.

In the **Figure 7.7** are presented the sequences of extended faces of the 3D frames F_k for the 2D animation presented in **Figure 7.1**.

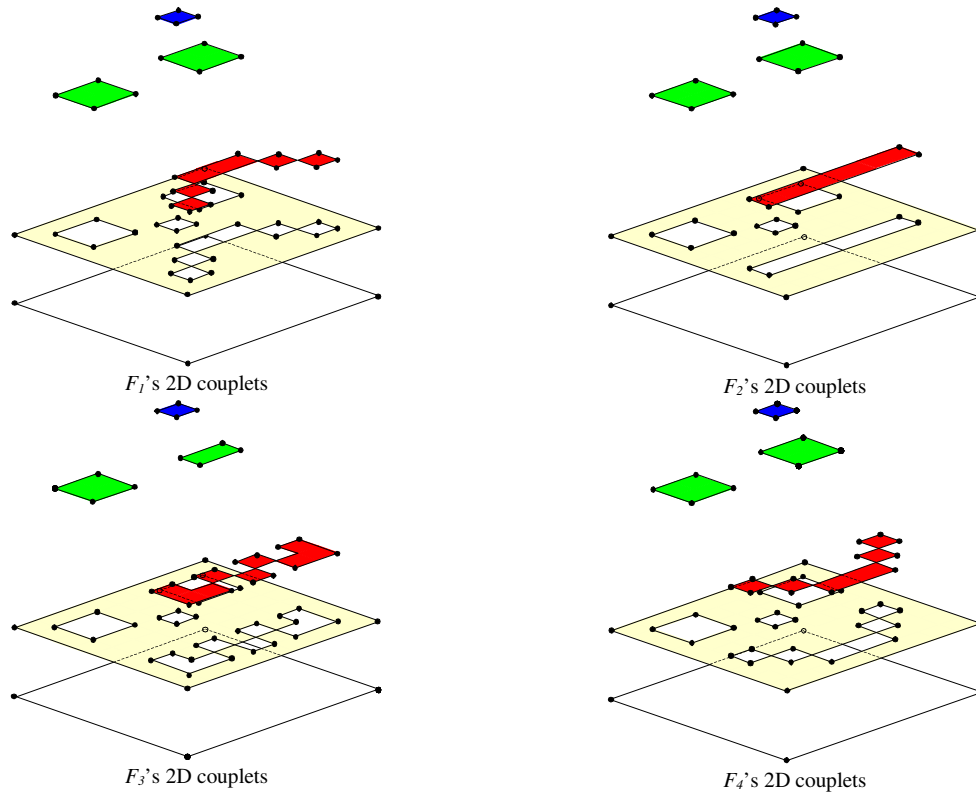


Figure 7.7. The sequences of extended faces (the polygons to display) of the 3D frames that represent a color 2D-animation (from **Figure 7.1**).

The **Algorithm 7.2** applies the above ideas in order to extract animation colored 2D frames from a 4D-OPP and display them. Basically it extracts the 3D couplets perpendicular to X_4 -axis and computes the sections that correspond to the extrusion to 3D space of the animation's 2D frames. When the extrusion of a frame is obtained then its 2D couplets perpendicular to X_3 -axis are extracted. Such 2D couplets are the polygons to draw and their filling color is assigned according to their common X_3 coordinate in the 3D frame. A 2D couplet and its color are processed through the procedure *Display* in the algorithm.

Input: A 4D-EVM p that represents a color 2D-animation.

```

Procedure playEVM-movie(EVM p)
    EVM hvl // Current 3D couplet in p.
    EVM Fprev, Fcurr // Previous and current 3D frames in the animation.
    EVM hvlF // Current 2D couplet in Fcurr. It contains the polygons to display.
    int color // The color to apply to the polygons to be displayed.
    Fprev = InitEVM( )
    hvl = ReadHvl(p)
    while (Not (EndEVM(p)))
        Fcurr = GetSection(Fprev, hvl) // We get the next 3D frame.
        // We proceed to display the current frame in the animation.
        while (Not (EndEVM(Fcurr)))
            /* Get the common coordinate of the vertices in the next 2D couplet to be
               extracted. */
            Color = getCurrentCoord(Fcurr)
            hvlF = ReadHvl(Fcurr)
            /* Polygons in the 2D couplet are displayed and filled according to
               variable color. */
            Display(hvlF, color)
        end-of-while
        Fprev = Fcurr
        hvl = ReadHvl(p) // Read next 3D couplet.
    end-of-while
end-of-procedure

```

Algorithm 7.2. Displaying a color 2D-animation represented through a 4D-OPP and the EVM.

The application we implemented (see **Figure 7.8**) performs the conversion taking from an input MPEG codified video some frames according to a time interval established by the user. The resolution of the frames corresponds to 320×240 pixels.



Figure 7.8. The GUI of the application developed for converting color 2D-animations to the 4D-EVM.

The **Table 7.1** shows some specific frames of an animation represented through the 4D-EVM. We selected the frames in such table according to the distinct takes that the movie sequence contains. Its time length was 79 seconds. We selected a frame at intervals of 0.15 seconds; therefore our animation had a total of 284 frames. The 4D-OPP that represented such set of selected frames has 22,208,664 extreme vertices.

Frame 1	Frame 20	Frame 30	Frame 60
Frame 90	Frame 120	Frame 150	Frame 180
Frame 210	Frame 240	Frame 270	Frame 284

Table 7.1. A sequence from “Star Wars” used for testing its conversion and codification through the 4D-EVM.

A total of 22,208,664 extreme vertices were required to codify it.

The **Table 7.2** shows another sequence of colored 2D frames. In this case we considered a movie sequence whose time length was 117 seconds. From its corresponding MPEG file we selected 430 frames. A frame was extracted from the original movie each 0.15 seconds. The size of the 4D-EVM corresponding to its codification as a 4D-OPP required 17,175,392 extreme vertices. As can be noted, in the sequence shown in **Table 7.1** we required 22,208,664 extreme vertices for representing 284 animation frames. The reason behind this behavior was yet identified in [Aguilera98]: Since $\pi_4(S_k^4(p)) = \pi_4(S_{k-1}^4(p)) \otimes^* \pi_4(\Phi_k^4(p))$ (see **Theorem 5.17**) then,

$EVM_3(F_k) = EVM_3(F_{k-1}) \otimes EVM_3(\pi_4(\Phi_k^4(p)))$, i.e., the regions at couplets $\Phi_k^4(p)$ represent the regions of a previous frame F_{k-1} that need to be modified in order to update it to the following frame F_k . In other words, a couplet perpendicular to X_4 -axis $\Phi_k^4(p)$ only stores the differences between consecutives 3D frames F_{k-1} and F_k . The way the frames change through time has impact over the number of extreme vertices in the couplets associated to the 4D-OPP that represents the animation. The exemplified animation from **Table 7.1** contains at least seven takes, or cuts, in its sequence. The frames between cuts contain slight differences, while frames before and after a cut have more considerable visual changes. On the other hand, the animation in **Table 7.2** is a sequence where there are no cuts. That is, the whole sequence is performed in only one take. Visual differences between frames along time are slight; hence, the transition between frames in the second animation is more uniform than transitions between frames in the first animation. Another point of impact is the number of colors in the scene. The second animation contains in all frames a black background, that is, a constant black region that does not change between frames is always present. These situations obviously have impact in the number of extreme vertices. See in **Appendix I** the cardinalities of the sets of extreme vertices in the couplets perpendicular to X_4 -axis in the referred animations. The mean cardinality in the first animation indicates that there are 77,839 extreme vertices per frame; while the mean cardinality in the second animation established 39,939 extreme vertices per frame.



Table 7.2. A sequence from “Star Wars”. The movie’s well known opening titles sequence was used for testing its conversion and codification through the 4D-EVM. A total of 17,175,392 extreme vertices were required to codify it.

7.1.1. Representing Color 3D-Animations through 5D-OPP's and the EVM

The managing of a color 3D-animation’s m frames can be performed in analogous way to the procedure described in the above section (we assume that each 3D frame is defined through a voxelization):

- The red-green-blue components of each voxel are integrated into a single value. Each voxel is 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 **Corollary 5.9**:

$$EVM_4(F_k) = \bigotimes_{i=1}^{npr} EVM_4(pr_i \in xf_k)$$

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

$$EVM_5(p) = \bigotimes_{k=1}^m EVM_5(hyperprism_k)$$

- The sequence of sections of p corresponds to the sequence of 4D frames, i.e., $\pi_5(S_k^5(p)) = F_k$.
- Computation of 4D frames: From **Theorem 5.17** we have that $\pi_5(S_k^5(p)) = \pi_5(S_{k-1}^5(p)) \otimes^* \pi_5(\Phi_k^5(p))$ then by **Corollary 5.8** $EVM_4(F_k) = EVM_4(F_{k-1}) \otimes EVM_4(\pi_5(\Phi_k^5(p)))$.
- Displaying the 3D colored animation: Each 3D couplet perpendicular to the X_4 axis of each 4D frame F_k contains the volumes to display. The colors to apply to those volumes are referred through the X_4 coordinate that contains the integrated red-green-blue components.

7.2. Application 2: A Procedure for Comparing Color 2-Dimensional Images Through their Extrusions to the 5-Dimensional Colospace

This section presents a procedure for comparing color 2-Dimensional images through their extrusions to the 5-Dimensional colospace ([Aguilera05], [Aguilera05b] & [Pérez-Aguila05b]). Some key operations to perform our comparison process include the computation of 5D hypervolumes and 5D Boolean operations (intersection and union). Finally, it describes an application of the proposed procedure under the context of the comparison and classification of a volcano's fumaroles.

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 [Huttenlocher93], [Jurisica00] or [Pass96]. In the next sections we will describe our proposed method for comparing color 2D-images which can be resumed in the following way ([Aguilera05], [Aguilera05b] & [Pérez-Aguila05b]):

- Extruding color 2D-images towards the 5D colospace (**Section 7.2.1**).
- Computing the 5D hypervolumes of the extruded images (**Section 7.2.2**).
- Determining if two color 2D-images are "initially similar" (**Section 7.2.3**).
- Computing the intersection between two extruded images (**Section 7.2.4**).
- Determining if two color 2D-images are similar (**Section 7.2.5**).

Section 7.2.6 presents the algorithm to perform the proposed comparison method. And finally **Section 7.2.7** describes a specific application under the context of the classification of a volcano's fumaroles and some considerations to be taken in account for this specific topic.

Our methodology for comparing images will be described under the context of the EVM in order to show the applicability of the model in this type of procedures.

7.2.1. A New Metric Over \mathbb{R}^+

Before going any further we will define a function that has a paramount role in our comparison method. In the **Theorem 7.1** we will show that such function is in fact a metric over \mathbb{R}^+ .

Definition 7.1 [Pérez-Aguila05c]: Let $x, y \in \mathbb{R}^+$. Let ρ be the function described as

$$\rho(x, y) = \begin{cases} 1 - \frac{x}{y} & \text{if } x < y \\ 1 - \frac{y}{x} & \text{if } y < x \\ 0 & \text{if } x = y \end{cases}$$

Theorem 7.1 [Pérez-Aguila05c]: Let $x, y \in \mathbb{R}^+$. Therefore $\rho(x, y)$ is a metric over \mathbb{R}^+ .

Proof:

Let $x, y, z \in \mathbb{R}^+$. See **Appendix J** for the referred Lemmas and Properties.

- We will show that $\rho(x, y) = \rho(y, x)$.
 - If $x = y \Rightarrow \rho(x, y) = 0 = \rho(y, x)$.
 - If $x < y \Rightarrow \rho(x, y) = 1 - x/y = \rho(y, x)$.
 - If $y < x \Rightarrow \rho(x, y) = 1 - y/x = \rho(y, x)$. $\therefore (\forall x, y \in \mathbb{R}^+)(\rho(x, y) = \rho(y, x))$
- By **Definition 7.1**: $(\forall x \in \mathbb{R}^+)(\rho(x, x) = 0)$.
- By **Definition 7.1** if $\rho(x, y) = 0 \Rightarrow x = y$.
- By **Property J.1**, $(\forall x, y \in \mathbb{R}^+)(\rho(x, y) \geq 0)$.
- We will show that $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$ (Triangle's inequality):
 - If $x = y = z \Rightarrow \rho(x, z) = 0 \leq \rho(x, y) + \rho(y, z) = 0$
 - If $x = z, x \neq y \Rightarrow \rho(x, z) = 0 \leq \rho(x, y) + \rho(y, z)$
 - If $x = y, x \neq z \Rightarrow \rho(x, z) = \rho(y, z)$
 - If $y = z, x \neq y \Rightarrow \rho(x, z) = \rho(x, y)$
 - If $x < y < z \Rightarrow$ By **Lemma J.1**, $\rho(x, z) < \rho(x, y) + \rho(y, z)$
 - If $x < z < y \Rightarrow$ By **Lemma J.2**, $\rho(x, z) < \rho(x, y) + \rho(y, z)$
 - If $z < x < y \Rightarrow$ By **Lemma J.3**, $\rho(x, z) < \rho(x, y) + \rho(y, z)$
 - If $z < y < x \Rightarrow$ By **Lemma J.4**, $\rho(x, z) < \rho(x, y) + \rho(y, z)$
 - If $y < x < z \Rightarrow$ By **Lemma J.5**, $\rho(x, z) < \rho(x, y) + \rho(y, z)$
 - If $y < z < x \Rightarrow$ By **Lemma J.6**, $\rho(x, z) < \rho(x, y) + \rho(y, z)$ $\therefore (\forall x, y, z \in \mathbb{R}^+)(\rho(x, z) \leq \rho(x, y) + \rho(y, z))$

$\therefore \rho$ is a metric over \mathbb{R}^+ . □

7.2.2. Extruding Color 2D-images Towards the 5D Colourspace

The color 2D-images are extruded towards the 5D colourspace: 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 [Duffin94]. This way the extrusion of each pixel will be a 5D hyperprism h_j and m will indicate the total number of hyperprisms obtained for a color 2D-image. It is recommended to avoid zero values for components R, G and B in order to obtain for each pixel its corresponding 5D hyperprism.

7.2.3. Computing the 5D Hypervolume of the Extruded Images

Let H the set of 5D hyperprisms for a color 2D-image. According to **Section 6.6.3**, since our extruded pixels are quasi-disjoint convex orthogonal polytopes then the 5D Extreme Vertices Model associated to the set H is given by:

$$EVM_5(H) = EVM_5\left(\bigcup_j h_j\right) = \bigotimes_j EVM_5(h_j)$$

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 [Aguilera05]. Because H is now represented through the EVM then we have that HV can be computed through **Algorithm 6.5**:

$$HV = Content(EVM_5(H), 5)$$

7.2.4. Determining if Two Color 2D-images are “Initially Similar”

Let $EVM_5(H_a)$ and $EVM_5(H_b)$ be the associated EVM's to 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 an 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 according to:

$$Q_{a,b}(HV_a, HV_b) = \rho(HV_a, HV_b)$$

Where ρ is the metric we defined in **Section 7.2.1**.

Let ε_1 be an arbitrary value such that $0 \leq \varepsilon_1 \leq 1$. Then, we will propose that two images a and b are “initially similar” [Aguilera05] (a second comparison will be considered in **Section 7.2.6**) 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):

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

For example, consider the images presented in **Figure 7.9** and $\varepsilon_1 = 0.05$ (both images have a size of 320×240 pixels). Then HV_a (according to our implementation) is $5,146,844 u^5$ (where u^5 stands for 5D hypercubical units) and HV_b is $4,996,787 u^5$. Therefore $Q_{a,b}(HV_a, HV_b) \approx 0.029$ which implies that $Q_{a,b} \leq \varepsilon_1$ and thus the images are “initially similar”.

The images from **Figure 7.10** were classified, according to our proposed procedure, as not “initially similar”. Let $\varepsilon_1=0.05$, 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 the condition $Q_{a,b} \leq \varepsilon_1$ is not fulfilled.



Image a



Image b

Figure 7.9. Two images classified as “initially similar” (see text for details; images obtained from the CENAPRED [Cenapred06]).



Image a



Image b

Figure 7.10. Two images classified as not “initially similar” (see text for details; images obtained from the CENAPRED [Cenapred06]).

7.2.5. Computing the Intersection Between Two Extruded Images

Now we will compute the intersection between $EVM_5(H_a)$ and $EVM_5(H_b)$ (the corresponding sets of 5D hyperprisms, represented through the EVM, for color 2D-images a and b) which were classified as “initially similar”. This process can be achieved by considering a call to **Algorithm 6.4** in the following way:

$$EVM_5(H_c) = EVM_5(H_a \cap^* H_b) = \text{BooleanOperation}(EVM_5(H_a), EVM_5(H_b), \text{IntersectionOperator}, 5)$$

As shown above, 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 [Pérez-Aguila05b]. In the **Figure 7.11** the color 2D-image, that is the result of intersecting the 5D colorspace’s extrusions of images presented in **Figure 7.9**, is shown.

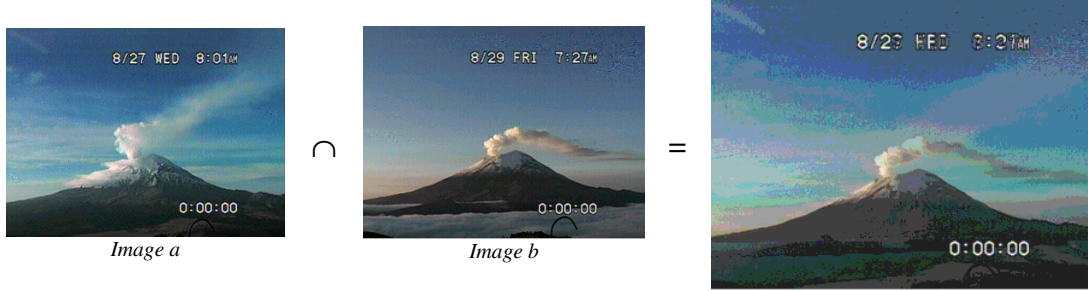


Figure 7.11. Computing the intersection between the 5D colorspace’s extrusions of two color 2D-images “initially similar” (from **Figure 7.9**; images a and b obtained from the CENAPRED [Cenapred06]).

7.2.6. Determining if Two Color 2D-images are Similar

We will compute the 5D hypervolume HV_c 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 [Aguilera05]. 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 [Aguilera05]:

$$Q_{a,c} = \rho(HV_a, HV_c)$$

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:

$$Q_{b,c} = \rho(HV_b, HV_c)$$

Let ϵ_a be an arbitrary assigned value such that $0 \leq \epsilon_a \leq 1$. 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 $0 \leq \epsilon_b \leq 1$ 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 [Aguilera05]:

$$\begin{aligned} Q_{a,c} &\leq \epsilon_a \\ Q_{b,c} &\leq \epsilon_b \end{aligned}$$

For example, consider the images and their intersection presented in **Figure 7.11**. Since **Section 7.2.4**, 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 $\epsilon_a = \epsilon_b = 0.30$, then we have, that through our procedure, both $Q_{a,c} \leq \epsilon_a$ and $Q_{b,c} \leq \epsilon_b$ are satisfied. Therefore, color 2D images a and b in **Figure 7.9** are classified as similar.

7.2.7. The Algorithm

The whole proposed procedure (**Sections 7.2.2 to 7.2.6**) for comparing two color 2D-images can be summarized through the **Algorithm 7.3**.

Input: Two-dimensional color images a and b .

The arbitrary value e_1 such that $0 \leq e_1 \leq 1$.

The arbitrary value e_a such that $0 \leq e_a \leq 1$.

The arbitrary value e_b such that $0 \leq e_b \leq 1$.

Output: True, if and only if input images a and b are similar. Otherwise, it is returned the false value.

Procedure ImagesAreSimilar(Image a , Image b , real e_1 , real e_a , real e_b)
/ We get the EVM's associated to the sets of 5D hyperprisms (extruded pixels) for images a and b */*
 EVM H_a = getEVMforExtrudedPixelsFromImage(a)
 EVM H_b = getEVMforExtrudedPixelsFromImage(b)
/ We calculate the 5D hypervolumes of the sets of the hyperprisms in H_a and H_b .*
 real H_{va} = Content(H_a , 5) *// Call to Algorithm 6.5.*
 real H_{vb} = Content(H_b , 5) *// Call to Algorithm 6.5.*
/ We compute the numeric difference between the 5D hypervolumes of the sets of hyperprisms. */*
 real $Q_{ab} = \rho(H_{va}, H_{vb})$
/ 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 ($Q_{ab} \leq e_1$) **then**
/ We get the EVM corresponding to the set of 5D hyperprisms which is the intersection between the 5D hyperprisms in H_a and H_b . */*
 EVM H_c = BooleanOperation(H_a , H_b , IntersectionOperator, 5) *// Call to Algorithm 6.4*
/ It is computed the 5D hypervolume of the intersection between sets of hyperprisms H_a and H_b . */*
 real H_{vc} = Content(H_c , 5) *// Call to Algorithm 6.5*
/ It is computed the proportion of hypervolume in H_a not included in H_c .*
 real $Q_{ac} = \rho(H_{va}, H_{vc})$
/ It is computed the proportion of hypervolume in H_b not included in H_c .*
 real $Q_{bc} = \rho(H_{vb}, H_{vc})$
/ If both proportions of hypervolume not included in H_c are less or equal than the allowed proportion indicated by input values e_a and e_b then */*
if ($(Q_{ac} \leq e_a)$ **and** ($Q_{bc} \leq e_b$)) **then**
 return true *// The images are similar.*
end-of-if
return false *// The images are not similar.*
end-of-if
return false *// The images are not "initially similar".*
end-of-procedure

Algorithm 7.3. Determining if two images are similar through their extrusions to the 5D colorspace.

7.2.8. An Application for Comparing Volcano's Fumaroles

The above proposed method for comparing images has been used in an experimental application related to the Popocatepetl volcano (located in the limits of Puebla state in México; and active and under monitoring since 1997) in order to evaluate its fumaroles ([Pérez-Aguila03d] & [Aguilera05b]).

We selected a total of 76 images from *CENAPRED* archives [Cenapred06] which compose a case base. These images represent some of the Popocatepetl volcano fumaroles during the year 2005. The selected images have a resolution of 640×480 pixels and 24-bits color under the format JPG. The objective of our application is given an input image, to obtain a subset of the 76 selected images which are classified as similar to that input image. Four inputs are expected from the user (see **Figure 7.12**):

- An image which will be compared with those in the case base.
- The value e_1 , i.e. the allowed difference between the 5D hypervolumes of two images.
- The value e_a , i.e. the allowed difference between the extrusion of the input image and the intersection between the extrusions of the input image with each one of the stored cases.
- The value e_b , i.e. the allowed difference between a stored case and the intersection between the extrusions of the input image with that stored case.

Given this set of inputs, the system retrieves those similar images of the volcano from the set of stored images.

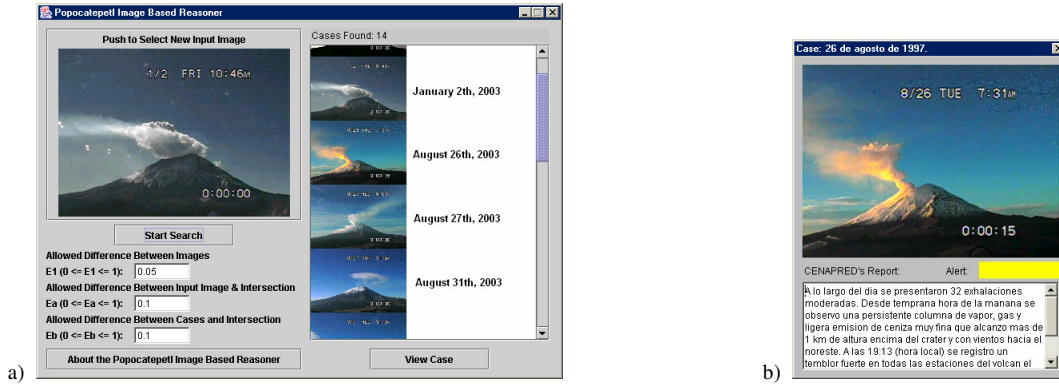


Figure 7.12. a) The graphical user interface which shows an example with a set of specific inputs and the corresponding results provided by the system. b) A retrieved case's detailed information.

In order to normalize the images in the case base, they were scaled down to a resolution of 320×240 pixels and segmented through a process of Multilevel Thresholding [Kurmyshev02]. Through this procedure we converted each 24-bits color image (with 16,777,216 possible colors or 256 possible values for each one of its color components) to a 4,096 colors image (where red, green and blue components have each one only 16 possible values). The following is an example of the function of Multilevel Thresholding over one of the color components (where $g(x,y)$ is the value of green component for a pixel in (x,y) and $G(x,y)$ is its new assigned value):

$$G(x, y) = 16 \left\lfloor \frac{g(x, y)}{16} \right\rfloor + 8$$

Since we are interested in comparing images by considering only the volcano's fumaroles, we will try to eliminate some "noise" that could have impact on the retrieval process. We will consider the volcano's silhouette as a source of "noise" because it has been observed that some images present that silhouette covered by snow while others don't present that situation and therefore the volcano's silhouette is visualized as a darker region (For example, see **Figure 7.13**).

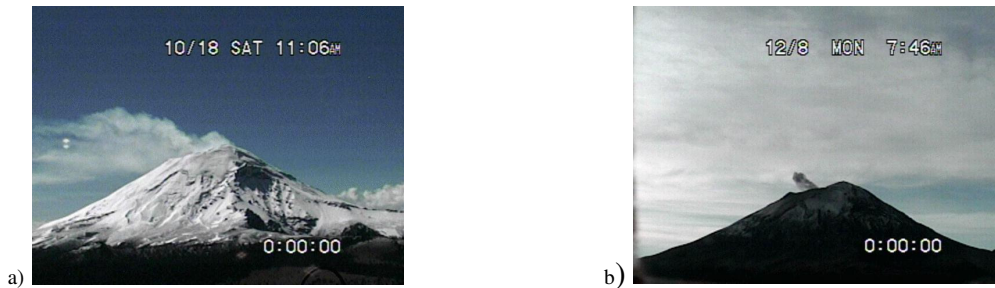


Figure 7.13 Two possible situations related with the visualization of a volcano's silhouette. a) The silhouette covered by snow. b) The silhouette visualized as a dark region (Images obtained from the CENAPRED [Cenapred06]).

As commented in **Section 7.2.4**, the 5D hypervolume of an extruded lighter (darker) pixel will have a greater (lesser) value. The situation related to the silhouette's visualization is closely linked to the final 5D hypervolume of an extruded image, because two images with similar fumaroles could have different silhouettes and therefore their total 5D hypervolumes could have important differences that could classify them as not "*initially similar*". We will eliminate this "noise" by assigning to all our selected images in the cases base (and the input image) the same volcano's silhouette [Pérez-Aguila05b]. This will be performed by computing the union of each one of the 76 extruded images with an image (also extruded to 5D colorspace) that contains a completely white volcano's silhouette and a black background (see **Figure 7.14**).

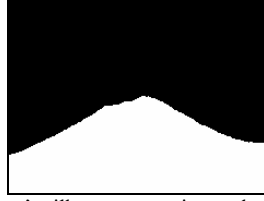


Figure 7.14. The volcano's silhouette to assign to the images in the case base.

The union of two extruded images can be performed in direct way by considering **Algorithm 6.4**. Let S be the set of 5D hyperprisms that represent the extrusion of the image in **Figure 7.14**, and let H be the set of 5D hyperprisms corresponding to one of the 76 images in our case base. Hence we should to operate them according to

$$EVM_5(H \cup^* S) = BooleanOperation(EVM_5(S), EVM_5(H), UnionOperator, 5)$$

Independently of the silhouette's coloration in the original images, the union of them with image of **Figure 7.14** will assign the white silhouette because all its color components are the highest possible, while the black background of **Figure 7.14** won't affect the original images' background because its color components are the lowest possible (see two examples in **Figure 7.15**).

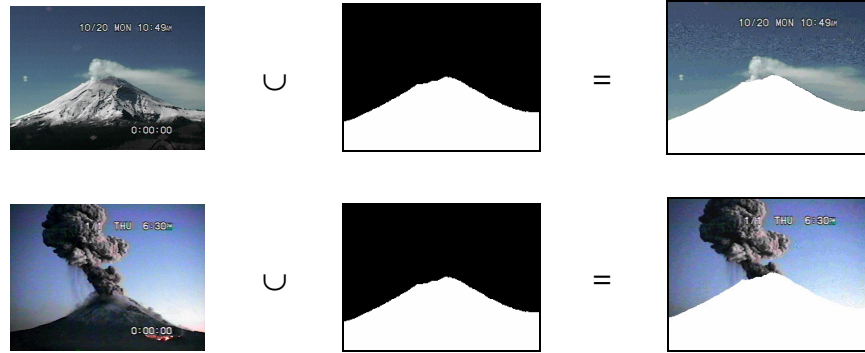


Figure 7.15. Two examples of the assignation of the same volcano's silhouette by performing the union of images

7.3. Application 3: Incorporating the nD-EVM to Image Based Reasoning

Case Based Reasoning has been used to attack some problems in different domains, such as the diagnosis and interpretation of diseases ([Oliver00], [Schmidt95] & [Schmidt99]) or the developing of automated tutors for games like chess [Lazzeri96]. In this section we will describe the incorporation of the nD-EVM to Image Based Reasoning, a paradigm described by Jurisica & Glasgow, which combines Case Based Reasoning and Image Processing and Analysis in order to compare images [Jurisica00]. The combination of those areas arises from the fact that not always the information is, according to Jurisica & Glasgow, symbolic; however, sometimes that information is associated to multimedia content (images, for example). This section is organized as follows: in **Sections 7.3.1**, **7.3.2** and **7.3.3** we will describe Image Based Reasoning's main aspects and in **Section 7.3.4** we will propose some extensions to the paradigm by taking in account our nD-EVM.

7.3.1. Images Storing

In [Jurisica00] two ways for storing an image in a case base are defined:

- **Explicit Storing:** An image is stored as a bitmap that contains all its associated visual information.
- **Implicit Storing:** Only some useful descriptors of the image are stored. For example, by transforming the raster image in a vector representation and by selecting only some of its polygons, lines, etc.

7.3.2. Image's Retrieval

In the images' domain one of the most important aspects to consider is the selection of the set of the most important characteristics in order to determine the similarity between two images and thus to achieve the retrieval process. Independently of the process for comparing two images, it is necessary to consider the previous preparation that they require. Jurisica & Glasgow summarize that previous preparation by considering the images' segmentation in order to reduce their complexity and to identify some objects in them:

- Region Oriented Segmentation: Which is based in the searching of regions with similar coloring. An example is the Multilevel Thresholding [Kurmyshev02].
- Edge Oriented Segmentation: Which is based in the searching of abrupt changes in the coloration that could indicate the presence of an edge between two or more objects.

Basically, in [Jurisica00] the comparison between two images is performed by using as similarity metric the number of geometric transformations required to make the images equivalent (as commented previously, in **Section 7.3.4** we will propose an additional metric).

7.3.3. The Process of the Image Based Reasoning

The process followed by an Image Based Reasoner system can be summarized as follows (See **Figure 7.16**):

- Input. An image represented through a bitmap.
- Originally in [Jurisica00] a way for indexing is not proposed, therefore all the images in the case base are candidates to be similar to the input image.
- If the images in the case base are stored explicitly, then each one must be considered for the segmentation process; otherwise, only the input image is segmented.
- The pair of images (the input one and each image in the case base), both in a vector representation, will be compared. The similarity metric proposed in [Jurisica00] has as principle the number of geometric transformations applied to the input image in order to make it similar to each one of the images in the case base.
- Output: The images in the case base which are similar to the input image.

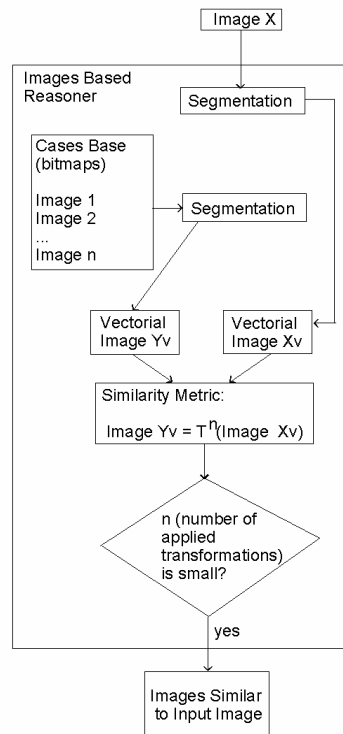


Figure 7.16. A schematic view of an Image Based Reasoner according to Jurisica & Glasgow [Jurisica00].

7.3.4. New Options for Retrieval in Image Based Reasoning

Now we will propose a second similarity metric: the comparison of images through their extrusions to the 5D colorspace (**Section 7.2**). The output of the procedure described in **Section 7.2.4** is a set of stored images classified as “*initially similar*” to the input. Since our process requires computing 5D geometric and Boolean operations under the 5D-EVM, implicit storing seems to be the most suitable format for image storage. For this format we consider the following descriptors (See **Section 7.2.3**):

- The EVM of the set of 5D hyperprisms associated with the original image.
- The total 5D hypervolume of the set of 5D hyperprisms.

Jurisica & Glasgow did not propose an indexing scheme associated with the images [Jurisica00]. This means that in order to select the most appropriated images stored in the case base it will imply to consider all of them. However, our proposed comparison procedure has to verify, as one of its first steps, whether or not two images are “*initially similar*” (see **Section 7.2.4**). This verification can be achieved by comparing the 5D hypervolumes associated to both images. The input image’s 5D hypervolume can be immediately computed after its extrusion to the 5D colorspace. Finally, we will select only as candidate cases those that are “*initially similar*” to the input image. See in **Figure 7.17** the schematic view of an Image Based Reasoner by considering the proposed options.

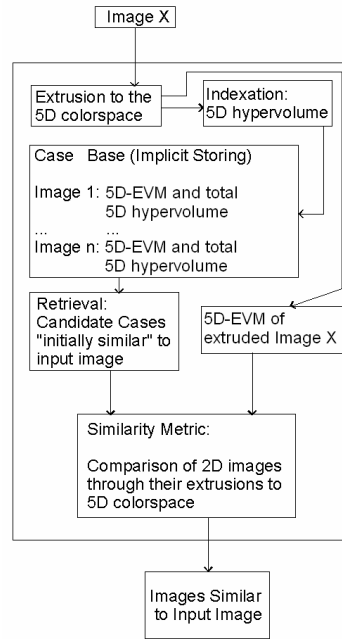


Figure 7.17. A schematic view of an Image Based Reasoner by considering the images’ comparison through their extrusions to 5D colorspace.

7.4. Application 4: Manipulating “Real World” 3D Datasets with the nD-EVM

In this section we will describe some results related to the conversion from voxelizations to our specific implementation of nD-EVM when $n = 3$. Such voxelizations correspond to “real world” datasets taken from the MoViBio Research Group [MoViBio06], and the University of Tübingen’s Project VolRen [VolRen06]. As commented in the **Section 6.6.3**, a 3D voxelization is a set of black and white cells where each cell is a convex orthogonal polytope. The set of black cells represents an nD-OPP p whose vertices coincide with some of the black cells’ vertices. Each of these vertices may be common to (surrounded by) up to 8 black cells. So, according to **Theorem 5.1**, if a vertex is surrounded by an odd number of black cells then it is an Extreme Vertex. Thus, a 3D voxelization to the 3D-EVM conversion algorithm is as simple as collecting every vertex that belongs to an odd number of cells, and discarding the remaining vertices. The **Tables 7.3, 7.4** and **7.5** show the measures we obtained when we converted 3D voxelizations, taken from the mentioned research groups, to our implementation of the nD-EVM when $n = 3$. In our measures we take in account the following aspects:

- Each cell in the source 3D voxelizations has only two possible values: 1 if the (black) cell is occupied, 0 if the (white) cell is empty. That is, we will deal with binary voxelizations.
- The evaluations were performed in a computer with Intel Celeron Processor at 2.40 Ghz and 256 megabytes in RAM memory. This equipment was isolated from network connections, virus scanners and utilities for the management and maintenance of files. This isolation has the objective of avoiding as possible the execution of additional processes that could affect the execution time of our algorithms.
- The algorithms were implemented using the Java Programming Language under the Software Development Kit 1.5 provided by Sun Microsystems.
- As commented in **Section 6.1.1** our EVM’s are stored and managed through trie trees. Our implemented algorithms consider this aspect.
- The measured execution times are expressed in seconds.

- We report the following execution times:
 - Time for computing the conversion from 3D voxelization to 3D-EVM.
 - Time for computing the content of the 3D-OPP (**Section 6.3**) expressed under the 3D-EVM.
 - Time for computing the boundary content of the 3D-OPP (**Section 6.4**) expressed under the 3D-EVM.
 - Time for computing the 2D sections, starting from the 2D couplets (**Section 5.5.3**) perpendicular to X_1 -axis, of the resulting 3D-OPP.

The descriptions corresponding to the set of objects being modeled in each voxelization are given in **Tables 7.3, 7.4** and **7.5**, as well as the total number of voxels in each representation.

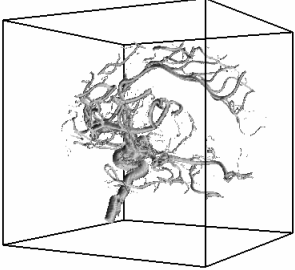
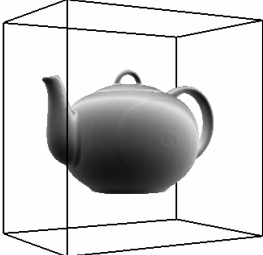
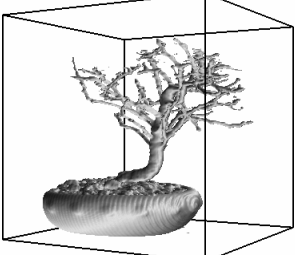
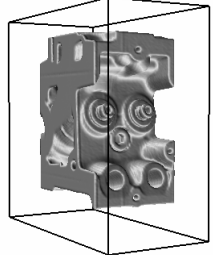
	<p>Aneurism Voxelization size: $(256 \times 256 \times 256) \equiv 16,777,216$ voxels Description: Rotational computer arm x-ray scan of the arteries of the right half of a human head. A contrast agent was injected into the blood and an aneurism is present.</p> <p>EVM size: 265,520 Time for conversion from Voxelization to EVM: 48.389 s Content: $168,948 u^3$ Time for computing content: 8.272 s Boundary content: $363,000 u^2$ Time for computing boundary content: 16.154 s Time for computing 2D sections: 8.312 s</p>
	<p>Teapot Voxelization size: $(256 \times 256 \times 178) \equiv 11,665,408$ voxels Description: Computed generated teapot.</p> <p>EVM size: 302,872 Time for conversion from Voxelization to EVM: 799.269 s Content: $4,932,534 u^3$ Time for computing content: 13.479 s Boundary content: $841,104 u^2$ Time for computing boundary content: 24.415 s Time for computing 2D sections: 12.488 s</p>
	<p>Bonsai Voxelization size: $(256 \times 256 \times 256) \equiv 16,777,216$ voxels Description: Computed tomography of a bonsai tree.</p> <p>EVM size: 641,462 Time for conversion from Voxelization to EVM: 858.905 s Content: $3,412,818 u^3$ Time for computing content: 33.709 s Boundary content: $2,098,246 u^2$ Time for computing boundary content: 71.322 s Time for computing 2D sections: 36.282 s</p>
	<p>Engine Voxelization size: $(256 \times 256 \times 128) \equiv 8,388,608$ voxels Description: Computed tomography of two cylinders of an engine block.</p> <p>EVM size: 644,060 Time for conversion from Voxelization to EVM: 699.145 s Content: $5,198,360 u^3$ Time for computing content: 30.504 s Boundary content: $1,320,558 u^2$ Time for computing boundary content: 65.864 s Time for computing 2D sections: 29.944 s</p>

Table 7.3. Results from the conversion of 3D voxelizations (*Aneurism*, *Teapot*, *Bonsai* and *Engine*) to the 3D-EVM.

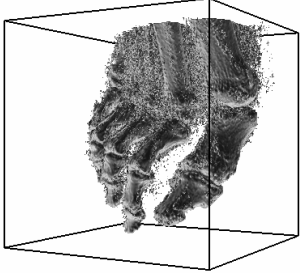
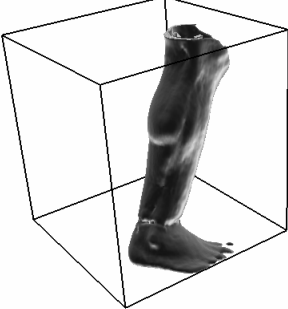
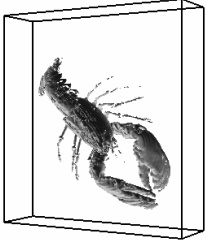
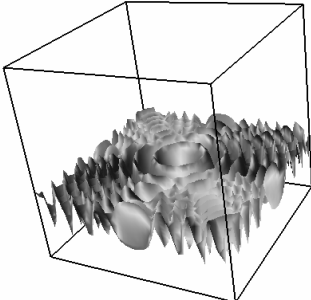
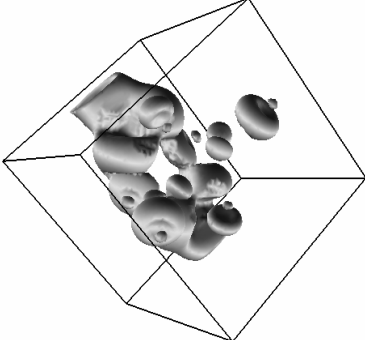
	<p>Foot Voxelization size: (256 × 256 × 256) ≡ 16,777,216 voxels Description: Rotation computer arm x-ray scan of a human foot. EVM size: 658,450 Time for conversion from Voxelization to EVM: 1,187.187 s Content: 4,854,701 u³ Time for computing content: 26.358 s Boundary content: 1,078,416 u² Time for computing boundary content: 54.479 s Time for computing 2D sections: 28.020 s</p>
	<p>Leg of statue Voxelization size: (341 × 341 × 93) ≡ 10,814,133 voxels Description: Computed tomography of a leg of a bronze statue. EVM size: 1,604,538 Time for conversion from Voxelization to EVM: 480.902 s Content: 2,499,595 u³ Time for computing content: 136.847 s Boundary content: 3,244,280 u² Time for computing boundary content: 367.579 s Time for computing 2D sections: 130.257 s</p>
	<p>Lobster Voxelization size: (301 × 324 × 56) ≡ 5,461,344 voxels Description: Computed tomography of a lobster contained in a block of resin. EVM size: 41,566 Time for conversion from Voxelization to EVM: 335.722 s Content: 3,831,352 u³ Time for computing content: 0.921 s Boundary content: 263,910 u² Time for computing boundary content: 1.582 s Time for computing 2D sections: 0.751 s</p>
	<p>Marschner/Lobb Voxelization size: (41 × 41 × 41) ≡ 68,921 voxels Description: Simulation of high frequencies where 99% of the sinusoids are right below the Nyquist frequency. EVM size: 872 Time for conversion from Voxelization to EVM: 1.332 s Content: 68,637 u³ Time for computing content: 0.010 s Boundary content: 11,070 u² Time for computing boundary content: 0.020 s Time for computing 2D sections: 0.001 s</p>
	<p>Neghip Voxelization size: (64 × 64 × 64) ≡ 262,144 voxels Description: Simulation of the spatial probability distribution of the electrons in a high potential protein molecule. EVM size: 12,998 Time for conversion from Voxelization to EVM: 5.208 s Content: 121,586 u³ Time for computing content: 0.250 s Boundary content: 34,930 u² Time for computing boundary content: 0.451 s Time for computing 2D sections: 0.170 s</p>

Table 7.4. Results from the conversion of 3D voxelizations (*Foot*, *Leg of statue*, *Lobster*, *Marschner/Lobb* and *Neghip*) to the 3D-EVM.

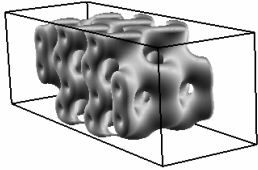
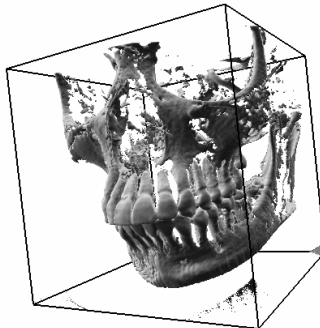
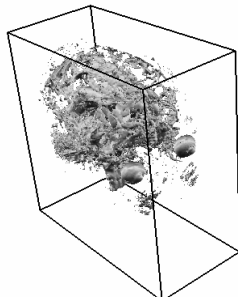
	<p>Silicium</p> <p>Voxelization size: $(98 \times 34 \times 34) \equiv 113,288$ voxels</p> <p>Description: Simulation of a silicium grid.</p> <p>EVM size: 1,164</p> <p>Time for conversion from Voxelization to EVM: 1.412 s</p> <p>Content: $66,163 \text{ u}^3$</p> <p>Time for computing content: 0.010 s</p> <p>Boundary content: $11,274 \text{ u}^2$</p> <p>Time for computing boundary content: 0.020 s</p> <p>Time for computing 2D sections: 0.001 s</p>
	<p>Skull</p> <p>Voxelization size: $(256 \times 256 \times 256) \equiv 16,777,216$ voxels</p> <p>Description: Rotational computer arm x-ray scan of a human skull.</p> <p>EVM size: 1,302,134</p> <p>Time for conversion from Voxelization to EVM: 2,626.757 s</p> <p>Content: $14,834,427 \text{ u}^3$</p> <p>Time for computing content: 87.907 s</p> <p>Boundary content: $2,656,000 \text{ u}^2$</p> <p>Time for computing boundary content: 245.873 s</p> <p>Time for computing 2D sections: 78.723 s</p>
	<p>CSF</p> <p>Voxelization size: $(256 \times 256 \times 124) \equiv 8,126,464$ voxels</p> <p>Description: Dataset corresponding to the Cerebro-Spinal-Fluid in a human head.</p> <p>EVM size: 86,570</p> <p>Time for conversion from Voxelization to EVM: 798.458 s</p> <p>Content: $8,115,182 \text{ u}^3$</p> <p>Time for computing content: 2.293 s</p> <p>Boundary content: $323,810 \text{ u}^2$</p> <p>Time for computing boundary content: 3.395 s</p> <p>Time for computing 2D sections: 2.283 s</p>

Table 7.5. Results from the conversion of 3D voxelizations (*Silicium*, *Skull* and *CSF*) to the 3D-EVM.

From **Tables 7.3, 7.4** and **7.5** can be observed, in first place, that the processing time for the conversion from the 3D voxelizations to the 3D-EVM was the largest from all the considered tasks. In this situation model *Skull* (**Table 7.5**) required 2,626.757 seconds (almost 45 minutes) for its conversion while, on the other hand, the shortest time for conversion corresponds to the model *Marschner/Lobb* (**Table 7.4**) with 1.332 seconds. It is interesting to observe that the model *Skull* does not contain the maximum number of extreme vertices in its corresponding EVM. The 3D model *Leg of Statue* (**Table 7.4**) was represented with 1,604,538 extreme vertices while *Skull* was represented with 1,302,134 extreme vertices, however, *Leg of Statue* required 480.902 seconds for its conversion to the 3D-EVM (approximately 8 minutes). The model *Marschner/Lobb* has both the lowest number of extreme vertices (872) and the lowest processing time for its conversion (1.332 seconds).

Now, let p be a 3D-OPP expressed under a voxelization with size $(x_1Size \times x_2Size \times x_3Size)$ and with $EVM_3(p)$ as its corresponding EVM. Consider the ratio

$$\frac{x_1Size \cdot x_2Size \cdot x_3Size}{Card(EVM_3(p))}$$

As can be seen, the idea behind such ratio is to express the number of times the quantity of voxels in the original representation of the object p is greater than the number of extreme vertices in its corresponding representation through the 3D-EVM. For example, consider model *CSF* (**Table 7.5**). Its source voxelization has size $(256 \times 256 \times 124)$ which implies that we require to store 8,126,464 voxels. The 3D-EVM associated to *CSF* has 86,570 extreme vertices (see **Table 7.5**). Hence, our proposed ratio gives us the value 93.87 which implies that the number of stored voxels that belong to the original representation of the object is precisely 93.87 times greater than the number of obtained extreme vertices. The **Table 7.6** shows the ratio Number-of-voxels/Number-of-Extreme-Vertices for the models described in **Tables 7.3, 7.4** and **7.5**. According to the results we obtained, the number of voxels in the model *Lobster* is 131.38 times greater than the cardinality of its corresponding 3D-EVM. In fact, model *Lobster* has the largest ratio from our twelve tested models. On the other hand, the model *Leg of Statue* (**Table 7.5**) has a number of voxels which is 6.73 times greater than the number of extreme vertices required for representing it. The value shared by our ratio depends on the topology and geometry of the objects being modeled, but it shows to us the conciseness, related to storing requirements, when we represent such objects through the EVM.

Object p	Voxelization Size (Number of voxels)	Card($EVM_3(p)$) (Number of extreme vertices)	$\frac{x_1Size \cdot x_2Size \cdot x_3Size}{Card(EVM_3(p))}$
<i>Aneurism</i>	16,777,216	265,520	63.18
<i>Teapot</i>	11,665,408	302,872	38.51
<i>Bonsai</i>	16,777,216	641,462	26.15
<i>Engine</i>	8,388,608	644,060	13.02
<i>Foot</i>	16,777,216	658,450	25.47
<i>Leg of Statue</i>	10,814,133	1,604,538	6.73
<i>Lobster</i>	5,461,344	41,566	131.38
<i>Marschner/Lobb</i>	68,921	872	79.03
<i>Neghip</i>	262,144	12,998	20.16
<i>Silicium</i>	113,288	1,164	97.32
<i>Skull</i>	16,777,216	1,302,134	12.88
<i>CSF</i>	8,126,464	86,570	93.87

Table 7.6. The ratio Number-of-voxels/Number-of-Extreme-Vertices for the 3D voxelizations shown in **Tables 7.3, 7.4** and **7.5**.

The importance behind a “real world” 3D dataset is the information can be obtained about it. As commented in the introduction of this section, we computed for each model shown in **Tables 7.3, 7.4** and **7.5**, through its corresponding 3D-EVM, the volume (u^3), the area of its boundary (u^2), and its sections. The 3D model *Leg of Statue* (that with the highest number of extreme vertices) required the maximum processing times for computing both its volume as the area of its boundary: 136.847 seconds and 367.579 seconds respectively. Conversely, the 3D model *Marschner/Lobb* (that with the lowest number of extreme vertices) required the minimum processing times for computing both its volume as the area of its boundary: 0.01 seconds and 0.02 seconds respectively. In the case related to the computing of sections, we found again that *Leg of Statue* required the maximum processing time: 130.257 seconds. A “tie” was found in the minimum processing time for sections corresponding to 3D models *Marschner/Lobb* and *Silicium* (**Table 7.5**): their sections were computed in only one millisecond.

Computing sections associated to the 3D models expressed under the 3D-EVM has an essential role in the analysis of these 3D datasets. Consider for example the Computed tomography shown in **Table 7.7**. In this case, we show the selected 13 sections from a patient’s head in order to inspect internal regions in his brain.

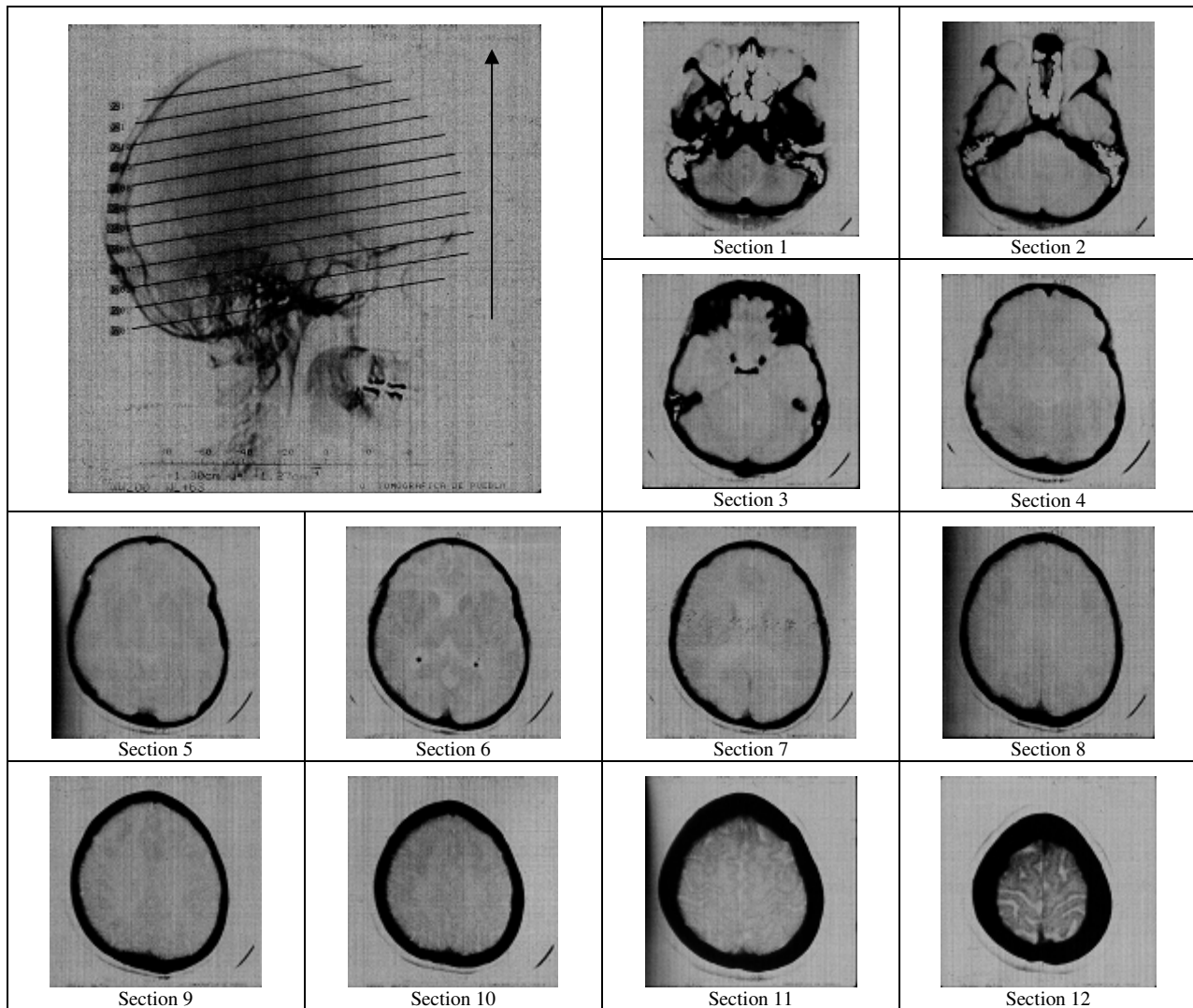


Table 7.7. A computed tomography which shows some sections of a patient's head.

If our 3D models are represented through the 3D-EVM then, according to the procedures mentioned in **Section 5.5.3**, their sections will describe to us the interior of the modeled objects with the objective to perform the appropriate analyses according to the application. The **Table 7.8** shows some sections corresponding to the 3D model *Bonsai* originally presented in **Table 7.3**. The model has a total of 641,462 extreme vertices and the required time for processing its 256 sections perpendicular to X_1 -axis was 36.282 seconds. The **Table 7.8** shows 18 of these 2D sections which allow observing the interior of the original 3D object. The original 3D voxelization has incomplete the bowl where the tree stands. Such incompleteness can be observed in the additional perspective of the model that we present in **Table 7.8** (the perspective used in **Table 7.3** does not permit to observe this characteristic).

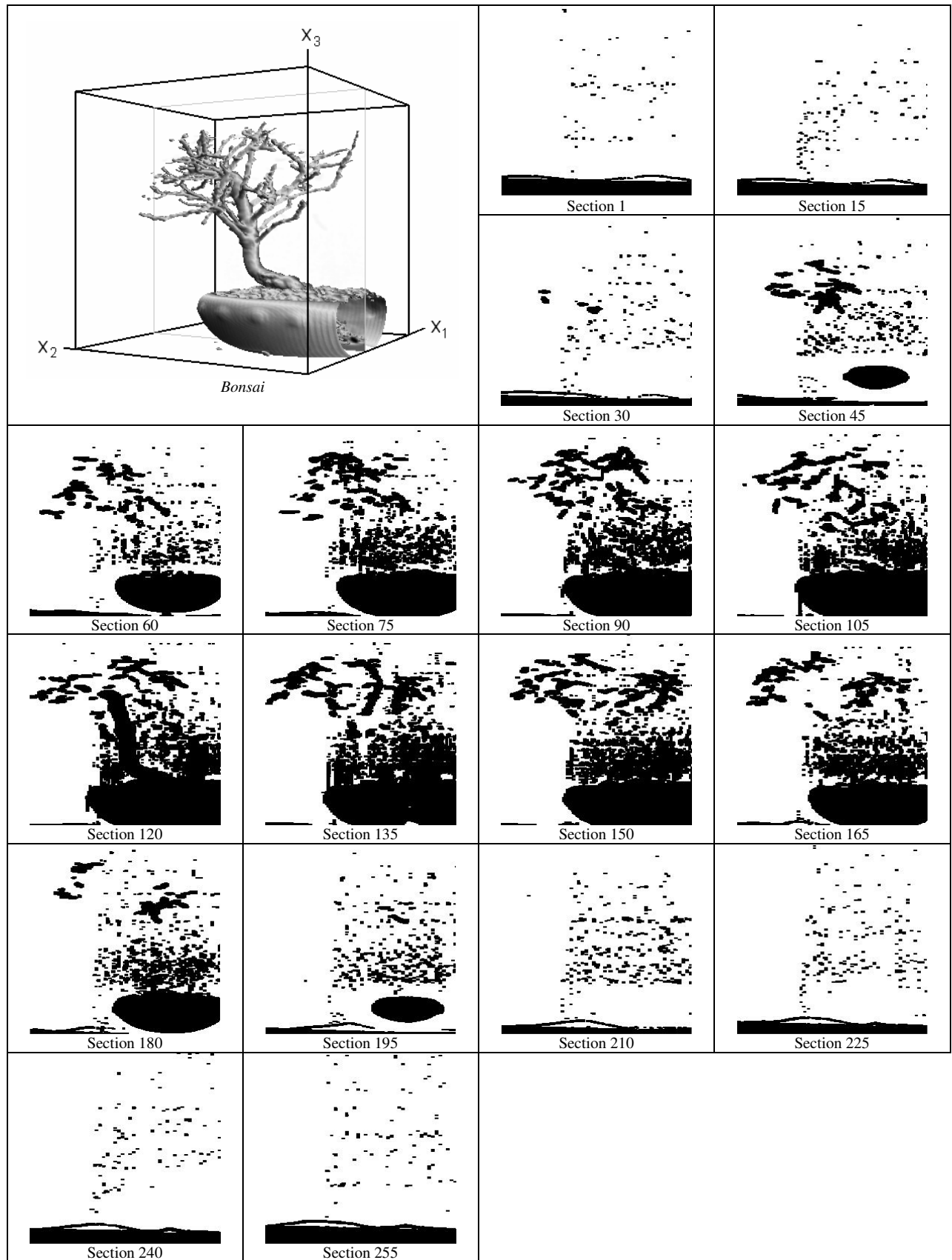


Table 7.8. Visualizing some sections perpendicular to X_1 -axis of the 3D-OPP *Bonsai* (Table 7.3) expressed in the 3D-EVM.

7.5. Application 5: Collision Detection

This application was originally proposed in [Zhou91]. It can be considered under our context in the following way: If p and q are two (4D polytopes) 3D polyhedra representing the motion of two (solids) polygons (See **Figure 7.18**), then these two (solids) polygons collide if and only if $p \cap q \neq \emptyset$. Furthermore, if $p \cap q \neq \emptyset$ then the time coordinate values of the intersection indicate the precise range time of the collision.

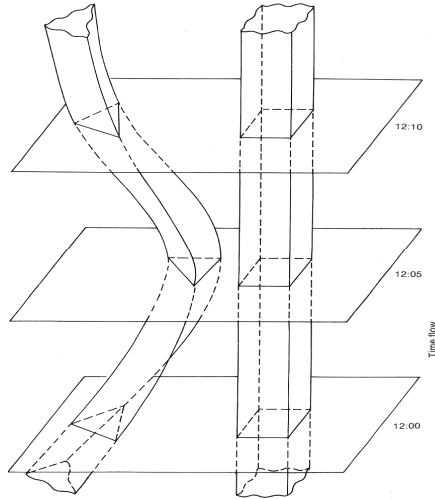


Figure 7.18. Two polyhedra representing the motion of a triangle and a square. In this specific case the intersection between such polyhedra is empty which implies that no collision occurred. Picture taken from [Rucker84].

In this application we will concern to the representation of the motion of an object in 3D space through a 4D polytope. Such motion is described by the 4D polytope under the Space-Time geometry: each instant in the motion will have associated points of the form (x_1, x_2, x_3, t) where the fourth dimension will be associated to time. We will proceed to represent the motion of a 3D-OPP as follows:

- 1) Let m be the number of instants that describe the motion of an object. Let $state_k$ be the state associated to the object at the instant t_k , $1 \leq k \leq m$. Each 3D-OPP $state_k$ describes the position of the object in 3D space and its geometry at the instant t_k . For example, **Figure 7.19** shows three states associated to the motion of an object at three consecutive instants ($m = 3$). In this specific example, we have that the geometry of the object changes along time.

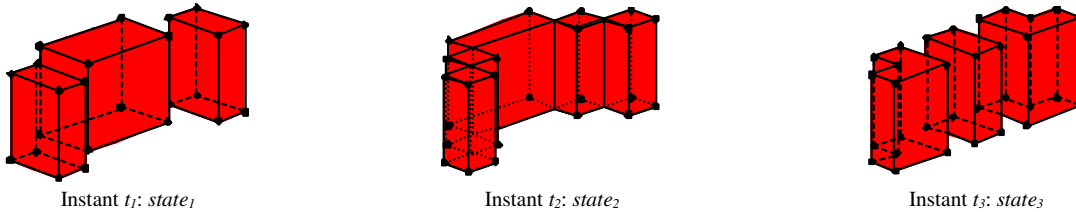


Figure 7.19. Three states associated to the motion of an object in 3D space (See text for details).

- 2) Each 3D-OPP $state_k$, $1 \leq k \leq m$, is extruded towards the fourth dimension and a 4D hyperprism $hyperstate_k$ is obtained. The bases of $hyperstate_k$ are the 3D-OPP $state_k$ and its length is proportional to the time that the object preserves its current state. Consider for example, in **Figure 7.20**, the 4D hyperprisms associated to the states described in **Figure 7.19**. In **Figure 7.20** can be observed that hyperprism $hyperstate_3$ shows that the object's state at instant t_3 is preserved more time than the other two states (represented by $hyperstate_1$ and $hyperstate_2$).

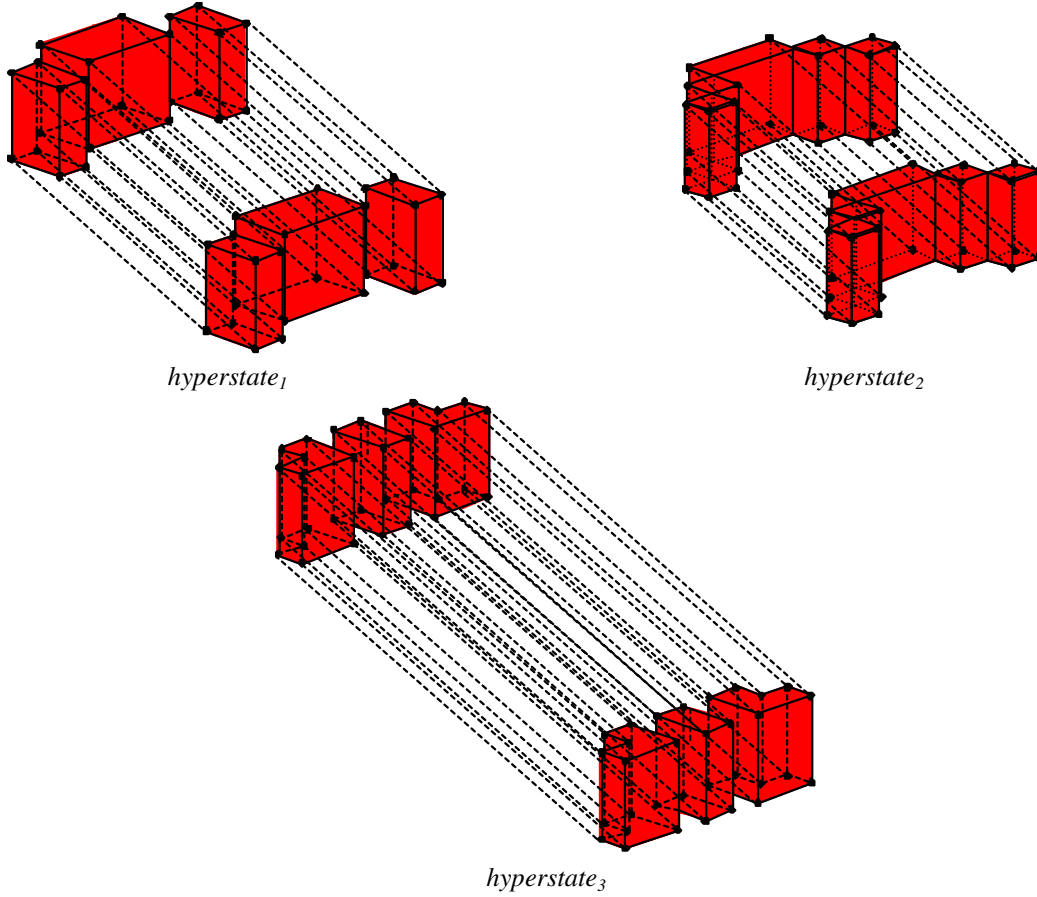


Figure 7.20. The three 4D hyperprisms associated to the motion of an object in 3D space (See text for details).

- 3) Let p be defined as

$$p = \bigcup_{k=1}^m \text{hyperstate}_k$$

Then p is a 4D-OPP that represents the motion of an object in 3D space. If each hyperstate_k is described through the 4D-EVM then

$$EVM_4(p) = EVM_4\left(\bigcup_{k=1}^m \text{hyperstate}_k\right) = \bigotimes_{k=1}^m EVM_4(\text{hyperstate}_k)$$

- 4) **Detecting Collisions:** Let p and q be two 4D-OPP's that describe the motion of two objects in the 3D space. We assume that p and q were composed under the same time scale and both are expressed through the 4D-EVM. Therefore, p and q collide if and only if:

$$EVM_4(p \cap^* q) = \text{BooleanOperation}(EVM_4(p), EVM_4(q), \text{IntersectionOperator}, 4) \neq \emptyset$$

Let r be a 4D-OPP resulting from the situation where $EVM_4(p \cap^* q) = EVM_4(r) \neq \emptyset$, that is, r describes the intersection between polytopes p and q which at their time describe motion of two objects in 3D space. Because r is expressed through the 4D-EVM, then the range time of the collision can be obtained by analyzing the set of np_4 3D-couplets from r which are perpendicular to X_4 -axis (See **Definition 5.14**). Because the extreme vertices in $EVM_4(r)$ that lie in each 3D-couplet of r have a common X_4 -coordinate, then we will have a set VX_4 of distinct X_4 -coordinates present in r (See **Definition 5.11**). In fact, $\text{Card}(VX_4) = np_4$. The set VX_4 provides us the precise range time of the collision between our 3D objects represented through 4D-OPP's p and q .

7.6. Conclusions

In this chapter we have experienced the modeling of certain applications under the context of the nD-EVM. In particular we considered:

- Classification of Color 2D-Images.
- Representation and manipulation of Color 2D and 3D Animations.
- Enhancing Image Based Reasoning.
- Manipulation of “real world” 3D datasets.
- Collision detection of 3D objects.

In each one of these applications we have described the way the algorithms and operations defined under the nD-EVM can be applied in order to solve the required tasks by these applications. The concise way those tasks can be expressed and performed under the EVM lead us to consider a wide range of additional applications to be expressed under our model.