A Framework For Digital Topology

Eladio Dominguez[1]
Angel Frances[2]
Alberto Marguez[3]

Technical Report No. CS-92-31

June 1992

[1]Universidad de Zaragoza 50009 Zaragoza Sapin
[2]Universidad de Zaragoza 50009 Zaragoza Spain
[3]Brown University Computer Science Box 1910, Providence, RI 02912

A FRAMEWORK FOR DIGITAL TOPOLOGY

Eladio Domínguez*, Angel Francés*, Alberto Márquez°

*Dept. de Ingeniería Eléctrica e Informática. Facultad de Ciencias
Universidad de Zaragoza. 50009 Zaragoza (Spain)

°Department of Computer Science. Brown University
Providence, Rhode Island 02912-1910 (USA)

March 17, 1992

Abstract

The main goal of this paper is to show the functional architecture of a framework for Digital Topology. This architecture has four levels, called Device, Logical, Conceptual and Continuous Levels. In each one of them we can use several models according to a particular problem. The models in the Device Level represent the physical problem whereas the models in the Continuous Level are topological spaces which allows us to use the well-known results of continuous topology (actually, the stronger results of polyhedral topology). The other two levels are used to find a digital solution. The Logical Level is closer to the Device Level and it is used for processing, for writing algorithms and to show their correctness. The Conceptual Level is the nearest to the Continuous Level and it is used to translate results and notions from the Continuous Level to the Logical Level.

1 Introduction

The main purpose of Digital Topology is the study of topological properties of discrete objects which are obtained digitizing continuous objects. Digital Topology plays a very important role in computer vision, image processing and computer graphics. But a complete theoretical foundation of a consistent theory for digital spaces is still missing. Kong and Rosenfeld give in [8] a very good survey about this subject.

A digital image is an object in the computer screen where the smallest elements are pixels or, more abstractly, compact topological cells such that they only have intersection in their borders. So it is natural to abstract the screen model as a graph whose vertices corresponding to the pixels and the edges represent the adjacency between the pixels. This is the most extended point of view (Rosenfeld [13], [14], [15], Kong, Roscoe [7]). But the problem is that this kind of models are far from the Euclidean plane (or space). So, from this point of view, it is needed to develop the Digital Topology without using the well-known results of Euclidean Topology.

The best solution would be to find a graph that represents a proximity relationship between the pixels, in such a way that this graph is induced by a topological structure. But this is impossible as a consequence of the characterization of the graphs induced by a topology, proved by Domínguez and Francés [3]. From this result, the graphs used in Digital Topology, which represent the proximity between the pixels, are not induced by any topology.

Some authors (Kovalevsky [10], Ankeney, Ritter [1], Khalimsky [6]) consider either a special topology on the grid of the integer numbers or a combinatorial structure on the screen model. In some cases, these models have similar disadvantages as in the case that we have commented above. In other cases, they can use results coming from Euclidean

Topology but there are problems in the processing because the models are far from the physical objects.

Our idea is to present a framework in which context we can have the advantages of the known models. For this purpose we use two models which are strongly related. One of them is near to the computer screen, and it is used for processing; the other is near to the continuous model and it is used to translate continuous results to the digital model.

We present this framework as a general theory for the development of the Digital Topology corresponding to abstract topological spaces. Firstly, we show it solving a well known problem (Digital Curve Inclusion Problem) and later, we explain the functional architecture in a general context.

2 The problem

Suppose that we have the following situation: In a computer screen we have a binary digital image formed by a curve and a point which does not belong to the curve (as it is shown in Figure 1).

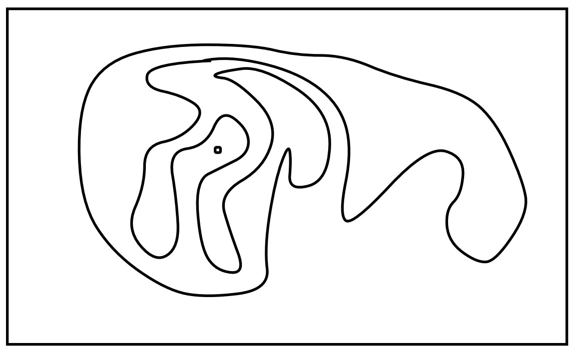


Figure 1: A digital image in the screen

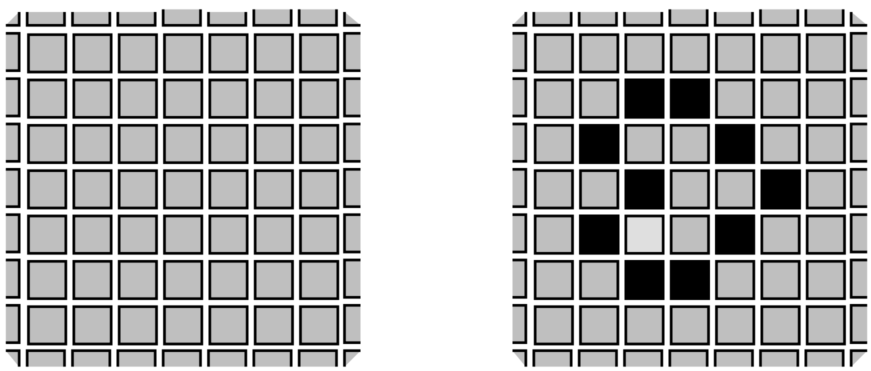
This image represents a topological situation in the Euclidean plane such that the curve is a Jordan curve. It is well-known that this kind of curves divide the plane in two regions with exactly one of them bounded. So the point in the image represents a point in one of the regions. Our problem is to decide whether or not the point belongs to the bounded region. That is, we want to solve the following

Digital Curve Inclusion Problem. Given a simple closed curve C in the screen and a pixel p which does not belong to the curve, determine whether or not p is internal to C.

But now, our problem is that the digital image is a discrete object. So we can not use the continuous techniques.

3 The mathematical models

We consider that the screen model is an infinite matrix S of pixels with the shape showed in Figure 2.a where each pixel can have two states represented by a dotted or small black square. So the curve of our image is represented taking some points in black. For the sake of simplicity, we consider that the curve in our problem is that one pictured in Figure 2.b.

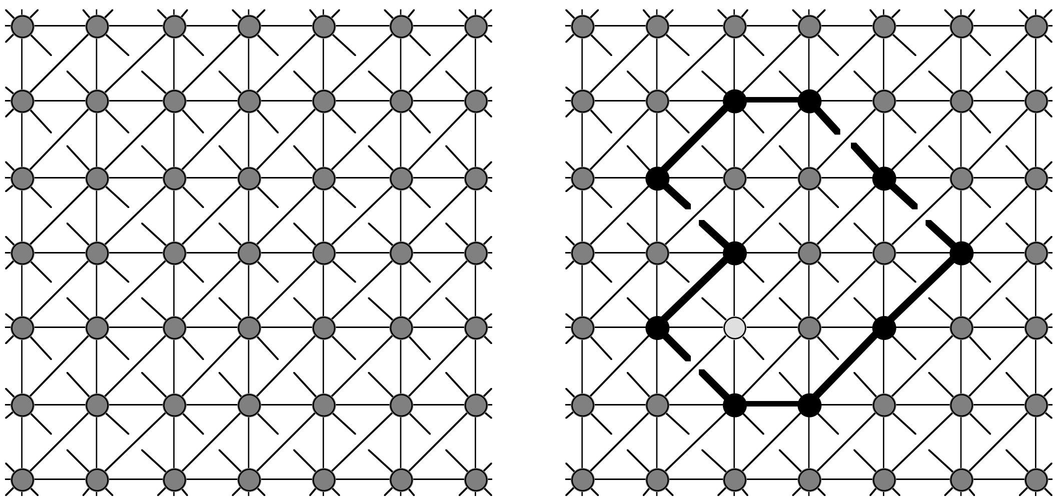


a. Screen model b. A curve C

Figure 2

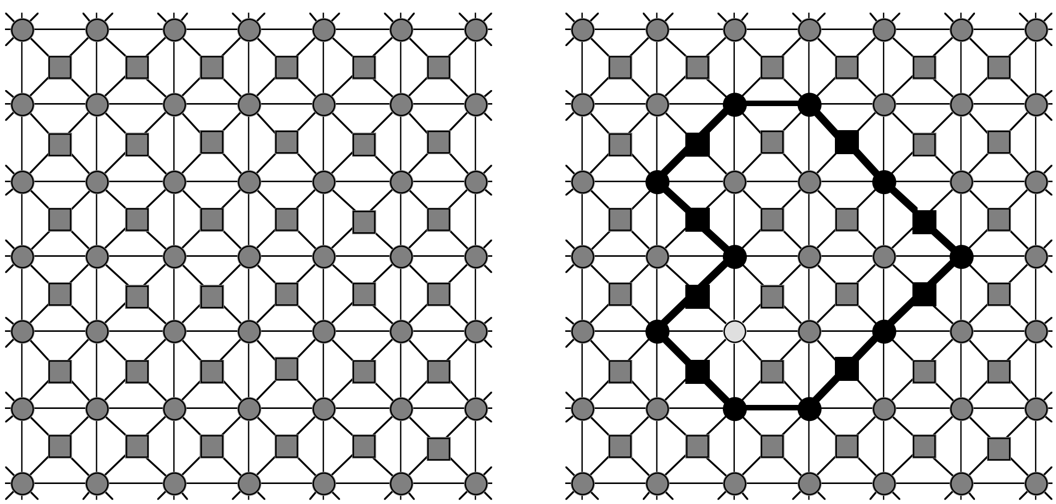
Since our problem has a topological nature, it is natural to represent the screen model with the graph E_8 represented in the Figure 3.a. For the sake of simplicity we will suppose that the vertex set of E_8 is $\mathbb{Z}^2$, the set of all pairs of integer numbers. The vertices of the graph represent the pixels and two vertices are adjacent if and only if their corresponding pixels are contiguous in the obvious sense. In order to solve our problem, this mathematical model represents the Logical Level. In it the curve becomes the subgraph induced by the vertices corresponding to black pixels.

Because E_8 is not planar, it is well-known that we can not represent in it the topology of the Euclidean plane (see Rosenfeld [13]). To solve that problem we flatten out this graph in a natural way and we get the planar graph E_8^* represented in the Figure 4.a. In this graph there are two different kinds of vertices. Some of them represent the pixels and the others, called middle points, represent a degree of nearness between the corresponding pixels; in fact it is the diagonal nearness. Observe that this graph is a triangulation of the Euclidean plane. That makes up the Conceptual Level to solve our problem.



a. E_8 b. The curve C

Figure 3



a. E_8^* b. The curve C^*

Figure 4

On the other hand, we define a digital image (or digital subspace) of one of these graphs as an induced subgraph; that is, a subgraph which contains an edge if and only if it contains the two vertices of the edge. We represent the set of digital images in E_8 and E_8^* by $\mathcal{O}(E_8)$ and $\mathcal{O}(E_8^*)$, respectively.

Now, we have a natural transformation π: $\mathcal{O}(E_8) \longrightarrow \mathcal{O}(E_8^*)$ defined as follows: "Given a digital image C in E_8, $\pi(C) = C^*$ is the subgraph generated by the vertices in C and the middle vertices defined by two diagonal vertices in C (see Figures 3.b and 4.b)." In a natural way we have a transformation π^*: $\mathcal{O}(E_8^*) \longrightarrow \mathcal{O}(E_8)$. Given a curve D^* in E_8^*, $\pi^*(D^*) = D$ is the subgraph generated by the vertices in D^* that are not middle vertices. Also we have a transformation j: $\mathcal{O}(E_8^*) \longrightarrow \mathcal{L}(\mathbb{R}^2)$ induced by the embedding of E_8^* in the Euclidean plane $\mathbb{R}^2$, where $\mathcal{L}(\mathbb{R}^2)$ is the set of polygonal subspaces.

In this way, we have the architecture represented by the following diagram

$$\begin{tikzcd} \mathcal{O}(E_8) \arrow[r, "i"] \arrow[d, shift right, "\pi"'] & \mathcal{O}(S) \\ \mathcal{O}(E_8^*) \arrow[r, "j"] \arrow[u, shift right, "\pi^*"'] & \mathcal{L}(\mathbb{R}^2) \end{tikzcd}$$

where $\mathcal{O}(S)$ is the set of digital images in the screen model S and i is the 1-1 transformation considered at the beginning of the section.

4 The Digital Jordan Curve Theorem

Observe that, through the transformation i, we can interpret the curve C as a simple digital curve in E_8 and the pixel p as a vertex in the graph E_8. It is obvious that the image by π of a simple closed digital curve C in E_8 is a simple closed digital curve C^*. And, also, the image by the embedding j of C^* is a polygonal Jordan curve C' in $\mathbb{R}^2$. On the other hand, $\pi(p) = p^* \in E_8^* - C^*$ and $j(p^*) = p' \in \mathbb{R}^2 - C'$ (see Figures 3, 4). In this way, we have translated our initial Inclusion Problem for the Screen Model to analogous problems in E_8, E_8^* and $\mathbb{R}^2$. Now then, it is well-known that the solution to this problem in $\mathbb{R}^2$ is the Polygonal Jordan Curve Theorem. So that, our goal is to translate this theorem, by mean of the transformations j and π^*, to E_8 in order to find a solution to the Inclusion Problem in this model.

In the literature, there exist several equivalent statements to the Polygonal Jordan Curve Theorem. Here, we consider one of them that is appropriated to find an algorithm solving this problem and to prove its correctness. The base of this statement is the notion of transversal intersection between a half-line, which is parallel to the axis OX, and a polygonal curve.

Definition 4.1. Let D be a polygonal curve in $\mathbb{R}^2$ whose set of vertices $\{p_0,..., p_n\}$ is counter-clockwise ordered. Let $r_q = \{(x,y);\ y = y_q \text{ and } x \geq x_q\}$ be a half-line, where $q = (x_q,y_q)$. There exists a *transversal intersection* between D and r_q if one of the following situations occurs:

a) r_q intersects the edge defined by the vertices p_i and p_{i+1} in only a point $p \notin \{p_i, p_{i+1}\}$.

b) there exists $i \in \{0,\ldots, n\}$ such that for some $k \geq 0$

1) $y_i = y_{i+1} = \ldots = y_{i+k} = y$

2) $y_{i-1} > y$ and $y_{i+k+1} < y$ or $y_{i-1} < y$ and $y_{i+k+1} > y$

3) $x_j \geq x$ for every $i \leq j \leq i+k$.

With this definition we can state the following well-known result.

Polygonal Jordan Curve Theorem. Let D be a simple closed polygonal curve in $\mathbb{R}^2$, then $\mathbb{R}^2$ - D has two connected components (one of them bounded and the other one unbounded). Moreover, a point $q \in \mathbb{R}^2$ - D belongs to the bounded component if and only if $\#(r_q \pitchfork D) \equiv 1$ (mod. 2), where $\#(r_q \pitchfork D)$ represents the number of transversal intersections between D and r_q .

It is important to point out that the result we want to translate to E_8 is applied to polygonal curves C' coming from a digital curve C in E_8 and half-lines $r_{p'}$, where p' belongs to $\mathbb{Z}^2$. In such a way, it is easy to observe that the transversal intersections between one of such a half-line $r_{p'}$ and one of such a curve C' is never in the case a) of the definition above. That is, this kind of intersections has always a vertex of the curve. So that, we can translate the notion of transversal intersection to E_8 and E_8^* in such a way that

$$\#(r_{p'} \pitchfork C') = \#(r_{p^*} \pitchfork C^*) = \#(r_p \pitchfork C)$$

On the other hand, in the Conceptual Level, represented by E_8^*, we can consider the natural notion of connectness induced by the graph structure. This notion coincides with the connectness notion induced by the topology of the Euclidean plane through the embedding j. So we can consider the connected components of E_8^* - C^*. Since E_8^* is a triangulation of $\mathbb{R}^2$, it is not difficult to prove that the number of connected components of E_8^* - C^* and $\mathbb{R}^2$ - C' agree; even more, each component K^* of E_8^* - C^* is the finite triangulation of a component K' of $\mathbb{R}^2$ - C' in the following sense:

1) Given K^* there is one and only one component K' of $\mathbb{R}^2$ - C' such that $K^* \subset$ K'.

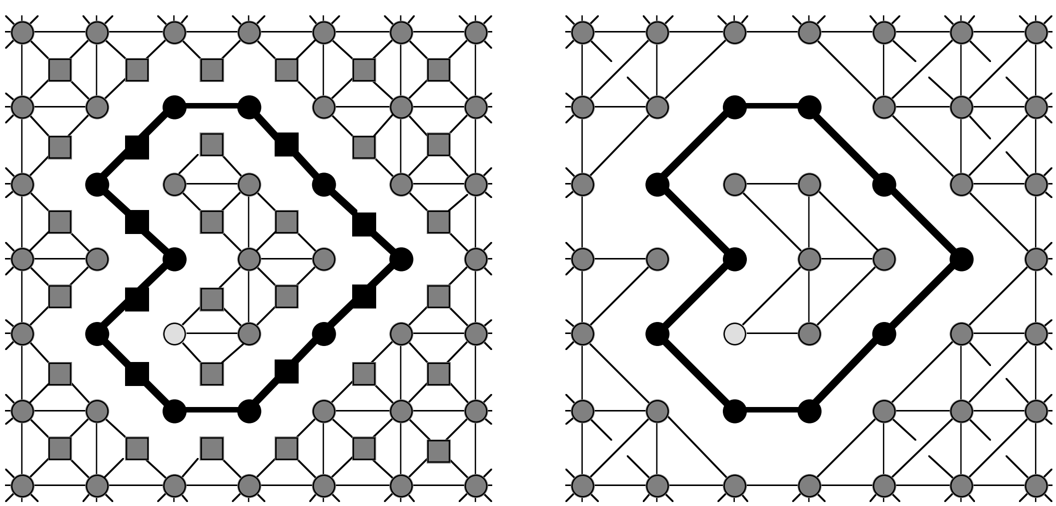
2) K^* is generated by the vertices of E_8^* in K'.

These properties show us that the components of $E_8^* - C^*$ represent the components of $\mathbb{R}^2 - C'$.

Now we can translate the Jordan Curve Theorem to E_8^* in the following way.

Jordan Curve Theorem in $\mathbf{E_8^*}$. Let C^* be a simple closed digital curve in E_8^*, then $E_8^* - C^*$ has two connected components (one of them bounded and the other one unbounded). Moreover, a point $p^* \in E_8^* - C^*$ belongs to the bounded component if and only if $\#(r_{p^*} \pitchfork C^*) \equiv 1$ (mod. 2).

Finally, we need to consider an appropriated notion of component in E_8. Let C be a simple closed digital curve in E_8. We call a *top-component* of E_8 - C to the image by π^* of a connected component of $E_8^* - C^*$. Observe that, in general, a top-component of E_8 - C does not coincide with a connected component of E_8 - C.



a. Components of $E_8^* - C^*$ b. Top-components of E_8 - C

Figure 5

In this way we have proved

Digital Jordan Curve Theorem. Let C be a simple closed digital curve in E_8, then E_8 - C has two connected top-components (one of them bounded and the other one unbounded). Moreover, a point $p \in E_8$ - C belongs to the bounded top-component if and only if $\#(r_p \pitchfork C) \equiv 1 \pmod{2}$.

5 The Digital Inclusion Problem

Up to now we have showed how to use our framework in order to find results in the Logical Level by using continuous results. At this point, using the model in this level we have to find an algorithm to decide whether a point p is or not in the bounded region that a digital closed curve determines. The goal of this section is to present an algorithm which solves this problem and to prove its correctness (this algorithm is presented as an example of how to use our framework and it is not pretended to be an optimal algorithm).

As we saw in previous sections, the Digital Jordan Curve Theorem gives us a method to solve this decision problem. To be exact, given a closed digital curve C in E_8 and a point p $\in E_8$ - C it is sufficient to calculate the number of transversal intersections between C and the digital half-line r_p.

However, we consider more appropriated to describe the algorithm by the relation between the points in the curve and the half-line. To do that, let C = $\{p_0,\ldots, p_n\}$ be a non-closed digital curve and let p be a point called problem point. Suppose that p = (x,y) and $p_i = (x_i,y_i)$, for every $i \in \{0,\ldots, n\}$. The next result classifies the points in the curve C with regard to their position relative to the half-line r_p.

Proposition 5.1. If $p \notin C$ and $y \neq y_0$ then every point p_i in C satisfies one and only one of the following properties.

P1) there exists $k_i \in \{0,\ldots, i\}$ such that $x_j < x$ for every $j \in \{k_i, k_{i+1},\ldots, i\}$.

P2) there exists $k_i \in \{0,\ldots, i\}$ such that $y_{k_i} > y$ and $x_j \geq x$, $y_j \geq y$, for every $j \in \{k_i, k_{i+1},\ldots, i\}$.

P3) there exists $k_i \in \{0,\ldots, i\}$ such that $y_{k_i} < y$ and $x_j \geq x$, $y_j \leq y$, for every $j \in \{k_i, k_{i+1},\ldots, i\}$.

Proposition 5.2. Let C be a non-simple closed digital curve in E_8 and let $p \in E_8$ - C be a problem point in the conditions of the proposition above. For every transversal intersection between C and the half-line r_p there exists one and only one point p_i satisfying the property P2 (respectively, P3) such that p_{i+1} satisfies P3 (respectively, P2). Such a point is called the exit point of the transversal intersection.

Proof. Suppose that C has a transversal intersection with r_p in the set $\{p_j, p_{j+1},\ldots, p_j, p_{j+k}\}$. It is straightforward from the definition of transversal intersection that either every point p_i satisfies P2 or every point p_i satisfies P3, $j \leq i \leq j+k$, and that p_{j+k+1} verifies the property P3 (respectively, P2). So, p_{j+k} is the exit point of this transversal intersection. ■

From now on, when we deal with several curves $C_1,\ldots, C_n$ and only one half-line r_p, if confusion is not possible, we will speak of an exit point of C_i instead of the exit point of a transversal intersection between C_i and r_p. And, when we have only a curve then we will speak of its exit points.

If C is a closed digital curve it is also possible to classify its points with regard to their relative position to the half-line r_p. Moreover, the condition on the problem point ordinate and the first point in the curve ordinate is not a true restriction when C is closed, because all the sets $\{p_k, p_{k+1},\ldots, p_n, p_0,\ldots, p_{k-1}\}$ with $k \in \{0,\ldots, n\}$ represent the same curve. Furthermore, when C is a closed digital curve, the point p_n could be the exit point of a transversal intersection as we establish in the following definition.

Definition 5.3. Let C be a simple closed digital curve in E_8 and let $p \in E_8$ - C be a problem point. p_n is an exit point if it verifies the property P2 (respectively, P3) and $p_{n+1} = p_0$ satisfies P3 (resp., P2).

With this definition it is possible to generalize the proposition above to closed digital curves, and then to prove the next result.

Corollary 5.4. Let C be a simple closed digital curve. Then, a point $p \in E_8$ - C belongs to the bounded component of E_8 - C if and only if the number of exit points of C is odd.

Algorithm

```
function INTERNAL((p0,...,pn): CURVE;
                        p: POINT);

var COUNTER,i: integer; UP, DOWN: boolean;

begin
    COUNTER:=0; UP:=false; DOWN:=false;
    ANALYZE(p0,p,COUNTER,UP,DOWN);
    i:=0;
    while i≤n+1 do begin
    ANALYZE(pi,p,COUNTER,UP,DOWN);
    i:=i+1;
    end;
    if COUNTER is odd then
        return "the point is in the bounded
                component"
    else return "the point is not in the bounded
                component"
end;
```

```
procedure ANALYZE(p',p: POINT;
    var COUNTER: integer; var UP, DOWN:boolean);
begin
    if x'≥x then
        if y'>y then begin
            UP:=true;
            if DOWN then begin
                COUNTER:=COUNTER+1;
                DOWN:=false end
        end
        else if y'<y then begin
                DOWN:=true;
                if UP then begin
                    COUNTER:=COUNTER+1;
                    UP:=false end
            end
    else begin
        UP:=false; DOWN:=false end
end;
```

Figure 6

This result gives us the idea to design our algorithm. In order to make easier the proof of its correctness we have divided the algorithm into two subalgorithms. The main is the function INTERNAL which decides whether the problem point p = (x,y) is or not in the bounded region of E_8 - C, where the input curve is $C = \{p_0, \ldots, p_n\}$, with $p_i = (x_i, y_i)$ for every $i = 0, \ldots, n$. The procedure ANALYZE decides which points of C are exit point (see Figure 6).

Three variables are used in the algorithm:

• COUNTER, with integer type, which is used to count the number of exit points of C.

• UP and DOWN, with Boolean type, which are used to represent the property P1, P2 or P3 that every point in the curve verifies.

Now, we are going to prove the correctness of our algorithm. To do that we use the specification technique; that is, we consider that the algorithm inputs verify a predicate, called precondition or input assertion, and the algorithm outputs satisfy another predicate called postcondition or output assertion. The last one must express the solution to the problem.

In our case, the precondition is $\{p \notin C \wedge y \neq y_0\}$ where $\wedge$ is the conjunction logic connector.

In order to make easier the proof, it is convenient also to consider some other predicates placed between the sentences of the algorithm. Two consecutive predicates are the precondition and the postcondition for the lines of the algorithm placed between them. In Figure 7 we have inserted those assertions as comments.

We have used the following notations.

• C_i represents the digital curve whose points are $\{p_0, \ldots, p_i\}$, $i = -1, \ldots, n+1$. Observe that C_{-1} represents the empty curve and $C_{n+1} = C$.

• $P(i) \in \{P1, P2, P3\}$ is the property that the point $p_i \in C$ satisfies.

• Q(i) is a shorter form for the assertion

$$(\neg\, UP \wedge \neg\, DOWN \Leftrightarrow P(i) = P1) \wedge (UP \wedge \neg\, DOWN \Leftrightarrow P(i) = P2) \wedge$$

$$(\neg\, UP \wedge DOWN \Leftrightarrow P(i) = P3),$$

where $\neg$ stands for the logic negation.

Algorithm Specification

```
function INTERNAL((p_0,...,p_n): CURVE; p: POINT);

var COUNTER,i: integer; UP, DOWN: boolean;

begin      {p ∉ C ∧ y ≠ y_0}
    COUNTER:=0; UP:=false; DOWN:=false;
{p ∉ C ∧ y ≠ y_0 ∧ COUNTER = 0 ∧ ¬ UP ∧ ¬ DOWN}
    ANALYZE(p_0,p,COUNTER,UP,DOWN);
{p ∉ C ∧ COUNTER = #ep(0) ∧ Q(0)}
    i:=0;
{p ∉ C ∧ COUNTER = #ep(i-1) ∧ Q(i-1) ∧ i = 1}
    while i≤n+1 do begin
{p ∉ C ∧ COUNTER=#ep(i-1) ∧ Q(i-1) ∧ i ≤ n+1}
    ANALYZE(p_i,p,COUNTER,UP,DOWN);
{p ∉ C ∧ COUNTER = #ep(i) ∧ Q(i)}
    i:=i+1;
{p ∉ C ∧ COUNTER = #ep(i-1) ∧ Q(i-1)}
    end;
{COUNTER = #ep(n+1)}
    if COUNTER is odd then
        return "the point is in the bounded component"
    else return "the point is not in the bounded
                component"
end;
```

```
procedure ANALYZE(p',p: POINT;
                        var COUNTER: integer;
                        var UP, DOWN:boolean);
begin
    if x'≥x then
        if y'>y then begin
            UP:=true;
            if DOWN then begin
                COUNTER:=COUNTER+1;
                DOWN:=false
            end
        end
        else
            if y'<y then begin
                DOWN:=true;
                if UP then begin
                    COUNTER:=COUNTER+1;
                    UP:=false
                end
            end
    else begin
        UP:=false; DOWN:=false
    end
end;
```

Figure 7

• #ep(i) is the number of exit points of C_i. Observe that p_n is an exit point of C depending on the behaviour of $p_0 = p_{n+1}$. So that, our goal is to calculate #ep(n+1).

With this technique, to prove the correctness of an algorithm is to prove that if a state of the algorithm variables satisfies a predicate, which is the precondition for some sentences, the execution of these sentences gives a state of the variables satisfying the postcondition. All this work is laborious but a routine. So, we only give a sketch of the proof.

ANALYZE Algorithm

Every time this algorithm runs, the number of exit points of C_i is found, for i = 0,…, n+1. We have to distinguish two cases in the execution of ANALYZE. The inputs for the general one are the point p_i, the problem point p, the number of exit points of C_{i-1} and the property P(i-1) verified by p_{i-1}. So that its precondition in this case is

$$\{p \notin C \wedge \text{COUNTER} = \#ep(i-1) \wedge Q(i-1)\}.$$

But when i = 0, because C_{-1} represents the empty curve, neither there exist exit points of C_{-1} nor it make sense to speak of the property verified by p_{-1}. Then, when ANALYZE runs for the first time its inputs are the point p_0 and the problem point p, and the parameters COUNTER, UP and DOWN have the values 0, false and false, respectively. In this case the precondition is

$$\{p \notin C \wedge y \neq y_0 \wedge \text{COUNTER} = 0 \wedge \neg\, \text{UP} \wedge \neg\, \text{DOWN}\}.$$

In any case the ANALYZE output assertion is

$$\{p \notin C \wedge \text{COUNTER} = \#ep(i) \wedge Q(i)\},$$

for every i = 0,…, n+1, and the proof of its correctness is straightforward.

INTERNAL Algorithm

Now, we have to prove that if the inputs of INTERNAL verify the precondition

$$\{p \notin C \wedge y \neq y_0\}$$

then this algorithm computes the number of exit points of C; that is, the final state of its variables satisfy the postcondition

$$\{\text{COUNTER} = \#\text{ep}(n+1)\}.$$

Suppose that the inputs satisfy the precondition. Then, when the variables are initialized it is obviously true the predicate

$$\{p \notin C \wedge y \neq y_0 \wedge \text{COUNTER} = 0 \wedge \neg\, \text{UP} \wedge \neg\, \text{DOWN}\},$$

which is the precondition needed by ANALYZE in its first execution. So that, the predicate

$$\{p \notin C \wedge \text{COUNTER} = \#\text{ep}(i\text{-}1) \wedge Q(i\text{-}1) \wedge i = 1\}$$

is satisfied when this subalgorithm runs and the variable i is set to 1.

The next steep in the proof is to prove that any iteration of the loop satisfies some invariant assertion and, of course, the loop ends. In one hand, it is easy to observe that this invariant assertion is

$$\{p \notin C \wedge \text{COUNTER} = \#\text{ep}(i\text{-}1) \wedge Q(i\text{-}1)\};$$

on the other hand, the loop stops in any case because $n > 0$, the variable i is set to 1 in the beginning and it is increased by one in every iteration.

Finally, i is equal to n+2 when the loop stops and then the output assertion is verified.

The following result summarizes the solution to the Digital Inclusion Problem.

Theorem 5.5. Whether a problem point $p \in E_8$ is internal to a simple closed digital curve C with N points can be determined in O(N) time, without preprocessing.

6 The functional architecture

Our framework has four levels, called Device, Logical, Conceptual and Continuous Levels.

In the Device Level we represent the objects in a computer screen (typically digital images). This level has a very few degree of abstraction and we only represent the physical aspects of the objects.

A second level of abstraction is obtained in the Logical Level. We consider in it the proximity aspects of the objects and so, we can study some properties of topological nature. The main function of this level is to be the support for writing the algorithms and to prove their correctness.

In general, the level above is far from the mathematical model in which we have a solution for our problem. So we need the Conceptual Level as an interface between the level above and the Continuous Level. To realize this interface it is necessary to translate: 1) Objects and properties from the Logical Level to the Conceptual Level and vice versa; 2) Objects and properties from the Conceptual Level to the Continuous Level; 3) Properties of objects in the Continuous Level to properties of objects in the Conceptual Level.

Finally, the Continuous Level is used to find a continuous solution. Observe that, actually, the objects and concepts obtained from the Logical Level are inside the Polyhedral Topology rather than the Continuous Topology and so we can use the more powerful tools of this field. The objects of our physical problem have been translated by consecutive abstractions from the Device Level. Now we must find a continuous solution in this level by using the well-known results of Polyhedral Topology and we translate it to the Logical Level across the Conceptual Level.

When we have a concrete problem we must choose specific models in each level and functions which can support the functionality that we have described. Specifically, suppose that these chosen models are D, L, C and S for the Device, Logical, Conceptual and Continuous Level, respectively. Let $\mathcal{O}(D)$, $\mathcal{O}(L)$, $\mathcal{O}(C)$ and $\mathcal{O}(S)$ be the sets of the objects (i.e., substructures in a mathematical sense) of these models. So we have the following functional architecture

$$\begin{tikzcd} \mathcal{O}(L) \arrow[r, "i"] \arrow[d, shift right, "\pi"'] & \mathcal{O}(D) \\ \mathcal{O}(C) \arrow[r, "j"] \arrow[u, shift right, "\pi^*"'] & \mathcal{O}(S) \end{tikzcd}$$

We represent our physical objects in the model D and we translate it to the model L by the function i. If we have in L enough knowledge to solve the problem we do not need to use the rest of the models; when we have the solution, we interprete it in D by the function i. Otherwise, we translate the objects to the model C. If we can find in it a solution we translate it to the model L by the function π^*. But if even in C we can find a solution, we translate the objects to the model S where we can apply all of the quite powerful tools and results of Polyhedral Topology and, if we find a solution, we translate it to L by the functions j and π^*.

From a theoretical point of view, the particular structures that we need in the levels depend on the problem we want to solve. But, in general, there is a basic structure for a wide range of problems. For example, the basic structure to the Plane Digital Topology is the one shown in the paragraphs above. But sometimes it is necessary to enlarge this structure. In [5] we enlarge the model E_8^*, used in the Conceptual Level to prove the Digital Jordan Theorem, by another model E_8^Δ in order to prove the Digital Schoenflies Theorem. This graph E_8^Δ is connected to the above one by two functions

$$\mho(E_8^*) \rightleftarrows \mho(E_8^\Delta)$$

which have the same functional properties that π and π*. Also there is a function

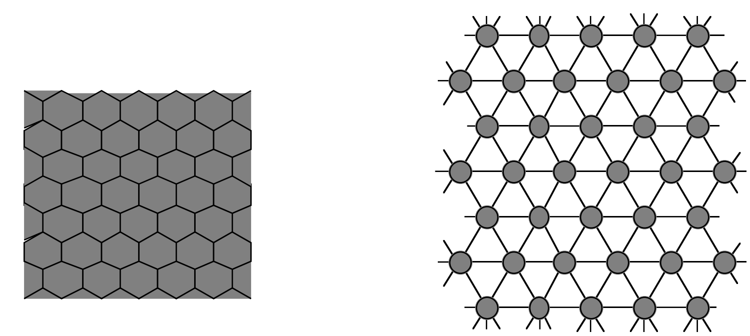
$$\mho(E_8^\Delta) \longrightarrow \mho(\mathbb{R}^2)$$

which is compatible with j: $\mho(E_8^*) \longrightarrow \mho(\mathbb{R}^2)$.

The idea is that the Logical Level has very few information about the connectivity degrees between the digital cells. So, if it is required by a problem, it is necessary to enlarge the basic structure of the Conceptual Level in order to represent all the connectivity degrees needed. This necessity grows with the dimension of the topological space.

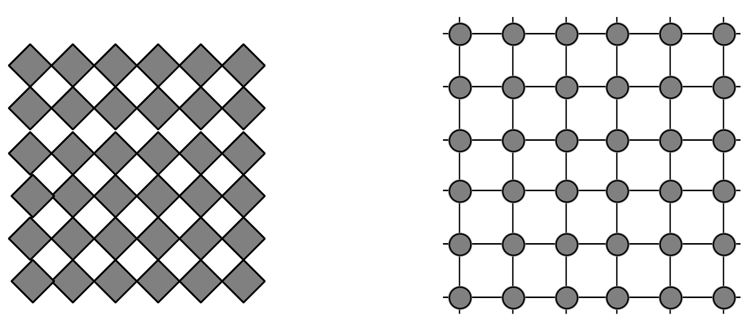
7 Final remarks

In this paper, we have developed a general framework that allows to use very powerful tools and results (those of Polyhedral Topology) for Digital Topology. The models used in the architecture depend on the Screen Model. For example, if our Screen Model is represented by Figure 8.a, the graph used as Logical Model is pictured in Figure 8.b (called hexagonal graph). In this case, each pair of cells has the same connectivity degree and the graph is planar, so we choose the same graph to represent the Conceptual Level. Observe that, in this case, this model verifies the Jordan Curve Theorem. The proof is the same as in the case of the graph of the 8-adjacencies.



a. Screen model b. Graph E_6

Figure 8



a. Screen model b. Graph E_4

Figure 9

There are models which do not verify the Jordan Curve Theorem. For example, we can consider that the graph E_4, represented by Figure 9.b, which is the logical model of the screen model represented by Figure 9.a. This graph is planar, so we must consider the same graph in the Conceptual Level. Thus the top-connection is equivalent to the 4-connection. Now, the vertices of a minimal cycle on this graph define a closed simple digital curve but its complement is connected. In fact, the only digital curves which verify the Jordan Curve Theorem are those whose embedding in the graph E_8 is a digital curve.

Very frequently, some authors have shown a proof of the Digital Jordan Curve Theorem in order to prove the consistency of their theories. That is why we also have chosen it for our framework. In the literature there are several proofs of this theorem using different techniques. The first author who gave a proof was Rosenfeld who presented two versions in a serie of papers ([13] [14],[16]). One is taking an 8-curve (i.e., a curve in the graph E_8) and proving that its complement has two 4-connected components (i.e., connected by arcs in the graph E_4 of the 4-adjacencies). The other is taking a 4-curve (i.e., a curve in the graph E_4) and proving that its complement has two 8-connected components (i.e., connected by arcs in E_8).

It is easy to observe that the theorem presented in this paper includes both versions. This is a direct consequence of the following property. Given a closed digital curve C in E_8, then:

1) If C is an 8-curve, the top-components of E_8 - C coincide with the 4-connected components.

2) If C is an 4-curve, the top-components of E_8 - C coincide with the 8-connected components.

In next papers we will develop our framework in depth. In [4] we present the mathematical structure which is used in the Logical and Conceptual Levels. And in [5] we present the basic properties of our architecture in order to develop the Plane Digital Topology.

Acknowledgements. We would like to thank Julio Rubio for his very useful comments on an earlier draft of this paper. We also acknowledge the partial assistance of the Diputación General de Aragón (D.G.A) (for the work of Angel Francés) and the Dirección General de Investigación Científico y Técnica (DGICYT) and the Junta de Andalucía (for the work of Alberto Márquez).

References

[1] L.A. Ankeney and G.H. Ritter, Cellular Topology and its Applications in Image Processing, *Int. Journ. Compt. Inf. Sciences* 12, 1983, 433-455.

[2] B. Bollobás, *Graph Theory: an introductory course*, Graduate Texts in Math., vol 63, Springer-Verlag, 1979.

[3] E. Domínguez, A. Francés, Characterization of the Graphs Induced by a Topology, to appear.

[4] E. Domínguez, A. Francés, A. Márquez, Digital Spaces: A Mathematical Model for Digital Image Processing, to appear.

[5] E. Domínguez, A. Francés, A. Márquez, The Topology of the Euclidean Digital Plane, to appear.

[6] E. Khalimsky, Computer graphics and Connected Topologies on finite ordered sets, *Topology and Appl*. 36, 1990, 1-17.

[7] T.Y. Kong, A.W. Roscoe, A Theory of Binary Digital Pictures, *Comput. Vision Graphics Image Process*. 32, 1985, 221-243.

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

[9] R. Kopperman, P.R. Meyer, R.G. Wilson, A Jordan Surface Theorem for three-dimensional Digital Spaces, *Discrete & Computational Geometry* 6, 1991, 155-161.

[10] E. Kovalevsky, The topology of cellular complexes as applied to image processing, in *Computer Analysis of Images and Patterns*, Proc. II Int. Conference CAIP'87 on Automatic Image Processing, 1987, pp. 162-173.

[11] F.P. Preparata, M.I. Shamos, *Computational Geometry: an introduction,* Texts and Monographs in Computer Science, Springer-Verlag, 1985.

[12] G.X. Ritter, Topology of Computer Vision, *Topology Proceedings* 12, 1987, 117-158.

[13] A. Rosenfeld, Connectivity in Digital Pictures, J*ourn. Assoc. Compt. Mach.* 17, 1970, 146-160.

[14] A. Rosenfeld, Arcs and Curves in Digital Pictures, *Journ. Assoc. Compt. Mach.* 20, 1973, 81-87.

[15] A. Rosenfeld, Adjacency in Digital Pictures, *Information and Control* 26, 1974, 24-33.

[16] A. Rosenfeld, Digital Topology, *Amer. Math. Monthly* 86, 1979, 621-630.