Inside and Outside within Combinatorial Pyramids

Luc Brun† and Walter Kropatsch‡*

† GreYC -CNRS UMR 6072
ENSICAEN
6 Boulevard du Maréchal Juin
14045 Caen(France)
‡ Institute for Computer-aided Automation
Pattern Recognition and Image Processing Group
Vienna Univ. of Technology- Austria

† luc.brun@greyc.ismra.fr,‡krw@prip.tuwien.ac.at

Abstract. Irregular pyramids are made of a stack of successively reduced graphs embedded in the plane. Such pyramids are often used within the segmentation and the connected component analysis frameworks to detect meaningful objects together with their spatial and topological relationships. The graphs reduced in the pyramid may be region adjacency graphs, dual graphs or combinatorial maps. Using any of these graphs each vertex of a reduced graph encodes a region of the image. Using simple graphs one edge between two vertices encodes the existence of a common boundary between two regions. Using dual graphs and combinatorial maps, each connected boundary segment between two regions is associated to one edge. Moreover, special edges called loops may be used to differentiate a special type of adjacency where one region surrounds the other. We show in this article that the loop information does not allow to distinguish inside and outside of the loop by local computations. We provide a method based on the combinatorial pyramid framework which uses the orientation explicitly encoded by combinatorial maps to determine inside and outside with local calculus.

1 Introduction

An irregular pyramid [7] is defined as a stack of successively reduced graphs. The hierarchical representation provided by such pyramids allows to reduce the computational cost of many graph algorithms using reduced versions of the initial graph. This representation also provides a nice framework for graph algorithms based on a divide and conquer strategy. Finally, the irregular pyramids provide a global representation which may be used to add further constraints on many graph algorithms.

* WK was supported by the Austrian Science Foundation under grants P14662-INF and FSP-S9103-N04

Irregular pyramids have been widely used to encode partitions within the segmentation and the connected component analysis frameworks [7, 8]. The dual graph Pyramids introduced by Kropatsch [6] are defined as a stack of dual graphs successively reduced. Within such pyramids the mapping of a non surviving vertex to a surviving one is performed by the contraction of their common edge. The contraction of a graph reduces the number of vertices while maintaining the connections to other vertices. As a consequence self loops or multiple edges may occur, some of them being redundant in that they do not surround any part of the graph. Such edges surround thus "empty inside" and are called empty self loops. These redundant edges may be locally characterized in the dual of the graph and suppressed by a removal step.

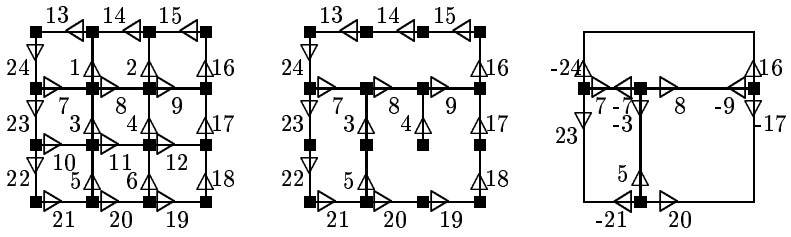
One particular type of adjacency between two regions is called the *includes* relationship. The includes relationship relates two regions, one is placed 'outside', the other is 'inside' and is surrounded by the outside region in the embedding (see e.g. the arrows in Fig. 3). A self-loop incident to the vertex encoding the outside surrounds the inside (Fig. 2(a)). The edge corresponding to the self-loop in the dual graph is a bridge connecting inside and outside region. Without orientation the exchange of inside and outside does not change the topology of the graph, the two graphs are indistinguishable. Such loops which are not present at the base level are created by the successive contraction and removal operations applied to build the pyramid.

A Combinatorial Pyramid is defined as a stack of successively reduced combinatorial maps. The reduction scheme used within the combinatorial pyramid framework is similar to the one used within dual graph pyramid. However, the formalisms of the combinatorial and dual graph pyramids are quite different. One of the main specific property of combinatorial maps is the explicit encoding of the orientation of the plane. The method presented in this paper uses the explicit encoding of the orientation by combinatorial maps to differentiate usual adjacency relationships from the includes ones.

The rest of this paper is structured as follows: We first present in section 2 the main properties of the combinatorial maps. Then we present in section 3 the combinatorial pyramid framework together with the main concepts used to compute inside relationships. Finally, we present the problem of the determination of the includes relationships in section 4 together with one method to determines the set of regions inside a given one. We conclude this last section with an experiment illustrating the usefulness of includes relationship.

2 Orientation in Combinatorial maps

A combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$ encodes a partition on an orientable surface without boundary. Combinatorial maps are used within the image processing and analysis framework to encode image's partitions. Using $2D$ images, combinatorial maps may be understood as a particular encoding of a planar graph where each edge is split into two half-edges called darts. Since each edge connects two vertices, each dart belongs to only one vertex. A $2D$ combinatorial map is



(a) $\overline{G}$ (b) $\overline{G}$ contracted (c) $\overline{G}$ reduced

Fig. 1. A dual of a combinatorial map (a) encoding a 3×3 grid with the contracted combinatorial map (b) obtained by the contraction of $K_1 = \alpha^*(1, 2, 10, 11, 12, 6)$. The reduced combinatorial map (c) is obtained by the removal of the empty self loops defined by $K_2 = \alpha^*(4)$ and the double edges defined by $K_3 = \alpha^*(13, 14, 15, 19, 18, 22) \cup \{24, -16, 17, -20, 21, -23, 3, -5\}$.

formally defined by the triplet $G = (\mathcal{D}, \sigma, \alpha)$ where $\mathcal{D}$ represents the set of darts and σ is a permutation on $\mathcal{D}$ whose cycles correspond to the sequence of darts encountered when turning counter-clockwise around each vertex. Finally α is an involution on $\mathcal{D}$ which maps each of the two darts of one edge to the other one. Given a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$, its dual is defined by $\overline{G} = (\mathcal{D}, \varphi, \alpha)$ with $\varphi = \sigma \circ \alpha$. The cycles of permutation φ encode the faces of the combinatorial map and may be interpreted as the sequence of darts encountered when turning clockwise around a face. In what follows, the cycles of α, σ and φ containing a dart d will be respectively denoted by $\alpha^*(d)$, $\sigma^*(d)$ and $\varphi^*(d)$. An introduction to combinatorial maps and combinatorial pyramids may be found in [4, 2].

Fig. 1(a) describes a dual combinatorial map $\overline{G} = (\{\blacksquare\}, \{\blacksquare - \blacksquare\})$ encoding a 3×3 4-connected planar sampling grid. Using this encoding the φ, σ and α cycles of each dart may be respectively understood as elements of dimensions 0, 1 and 2 and formally associated to a 2D cellular complex [4]. More precisely, each α cycle may be associated to a crack between two pixels. Each of the two darts of an α cycle corresponds to an orientation along the crack. For example, the cycle $\alpha^*(1) = (1, -1)$ is associated to the crack encoding the right border of the top left pixel of the 3×3 grid (Fig. 1(a)). The darts 1 and -1 define respectively a bottom to top and top to bottom orientation along the crack.

These results may be extended to any reduced combinatorial map encoding an image partition. In this case, each dart of the map should be interpreted as a sequence of oriented cracks encoding an oriented and connected boundary segment between two regions. Such a sequence is simply called a *segment*. The relationships between segments and cracks are as follows: Each oriented crack belongs to at most one segment. Moreover, if one oriented crack belongs to a segment associated to a dart d, the same crack with an opposite orientation belongs to the segment associated to $\alpha(d)$. For example, the dart 16 in Fig. 1 (c) is associated to the sequence of oriented cracks encoded by the darts 16.15.14.13.24 (Fig. 1(b)) while the dart -24 is associated to $-24. - 13. - 14. - 15. - 16$.

3 Connected boundary segments and orientation within the Combinatorial Pyramid framework

As in the dual graph pyramid scheme [6] (Section 1) a combinatorial pyramid is defined by an initial combinatorial map successively reduced by a sequence of contraction or removal operations. Contraction operations are encoded by contraction kernels. These kernels defined as a forest of the current combinatorial map may create redundant edges such as empty-self loops and double edges(Fig. 1(b)). Empty self loops (edge $\alpha^*(4)$ in Fig. 1(b)) may be interpreted as region's inner boundaries and are removed by an empty self loops removal kernel after the contraction step. The remaining redundant edges called double edges, belong to degree 2 vertices in $\overline{G}$ (e.g. $\varphi^*(13)$, $\varphi^*(14)$, $\varphi^*(15)$) in Fig. 1(b)) and are removed using a double edge removal kernel which contains all darts incident to a degree 2 dual vertex. Note that, any combinatorial map deduced from the application of a contraction kernel followed by the two removal kernels cannot contain empty self loops. No dart d of such a combinatorial map may thus satisfy the relationship : $\sigma(d) = \alpha(d)$. Further details about the construction scheme of a pyramid may be found in [2, 3].

As mentioned in Section 2, each dart of a reduced combinatorial map may be associated to a sequence of oriented cracks called a segment. Since each oriented crack is encoded by one dart in the base level combinatorial map G_0(Section 2), a segment may be equivalently defined as a sequence of darts belonging to G_0. Let us consider a combinatorial map $G_i = (\boldsymbol{\mathcal{D}}_i, \sigma_i, \alpha_i)$ defined at level i such that G_i does not contain any empty self loop. Given a dart d in $\boldsymbol{\mathcal{D}}_i$ the sequence $d_1 \dots d_n$ encoding the segment associated to d is defined by [2] :

$$d_1 = d\,, d_{j+1} = \varphi_0^m(\alpha_0(d_j)) \text{ and } \alpha_0(d_p) = \alpha_i(d). \tag{1}$$

where $\overline{G_0} = (\boldsymbol{\mathcal{D}}_0, \varphi_0, \alpha_0)$ is the dual of the initial combinatorial map and m is the minimal integer such that $\varphi_0^q(\alpha_0(d_j))$ survives at level i or belongs to a double edge kernel. This last condition is tested in constant time using the implicit encoding of combinatorial pyramids [2].

Note that, if G_0 encodes the 4-connected planar sampling grid, each φ_0 cycle is composed of at most 4 darts (Fig. 1(b)). Therefore, the computation of d_{j+1} from d_j requires at most 4 iterations and the determination of the whole sequence of cracks composing a boundary between two regions is performed in a time proportional to the length of this boundary.

Each oriented crack associated to an initial dart d_j may be encoded by the position of its starting point and one move. Using a 4-connected sampling grid these moves belong to $\{right, up, left, down\}$. Given a dart d of G_i, let us denote respectively by $Fm(d)$ and $Lm(d)$ the moves of the first and last oriented cracks of the segment associated to d. If $d_1 \dots d_p$ denotes the sequence of initial darts associated to d, we have $d_1 = d$ and $d_p = \alpha_0(\alpha_i(d))$ (equation 1). The two darts d_1 and d_p may thus be retrieved in constant time from d. Moreover, $Fm(d)$ and $Lm(d)$ are equal to the move of the oriented cracks respectively associated to d_1 and d_p. This correspondence between the oriented cracks and the

initial darts may be defined using any implicit numbering of the initial darts (see e.g. Fig. 1(a)). The values of $Fm(d)$ and $Lm(d)$ may thus be retrieved without additional memory requirement and in constant time using an appropriate numbering of the initial darts. For example, the first and last moves of the dart 16 in Fig. 1(c) are associated to the moves of the darts 16 and $\alpha_0(-24) = 24$ in $\overline{G_0}$ (Fig. 1(a)) and are respectively equal to *up* and *down*.

Given a dart d in G_i, and the sequence of darts $d_1 \dots d_p$ in G_0 encoding its segment, the properties of the segments (Section 2) together with the properties of the combinatorial pyramids [2] induce the two following properties:

$$\forall j \in \{1, \dots, p-1\} \quad move(d_j)^{-1} \neq move(d_{j+1}) \tag{2}$$

$$Lm(d) \neq Fm(\sigma_i(d))^{-1} \tag{3}$$

where $move(d_j)$ denotes the move of the oriented crack associated to d_j and $move(d_j)^{-1}$ is the opposite of the move of d_j (e.g. $right^{-1} = left$).

Equation 2 states that two successive moves within a segment cannot be opposite. This property is induced by the fact that one segment cannot contain twice a same crack with two orientations. Equation 3 states that the first move of the σ_i successor of a dart d cannot be the opposite of the last move of d. Otherwise, the dart d would be an empty self loop of G_i which is refused by hypothesis.

Given a dart d_1 in G_i, let us consider a sub-sequence $d_1 \dots . d_q$ of $\sigma_i^*(d_1)$ such that $d_q \neq \alpha_i(d_1)$. Let us also consider the sequence S of oriented cracks defined as the concatenation of the segments associated to $d_1 \dots . d_q$. The orientation of the sequence $d_1 \dots . d_q$ is then defined as the overall number of clockwise turns between the successive cracks along S. In order to measure such an orientation we define the angle between two successive oriented cracks as :

- +1 if the two oriented cracks define a clockwise 90° turns,
- -1 if the two oriented crack define a counter-clockwise 90° turns,
- 0 if the two oriented crack correspond to a same move,
- undefined if the two oriented cracks correspond to opposite moves.

Such angles may be easily encoded using a basic 4×4 array indexed by the Freeman's codes of the moves: right,up, left and down are numbered from 0 to 3. The angle between two moves $m1$ and $m2$ is denoted by $(m1, m2)\hat{}$. We have for example, $(right, right)\hat{} = 0$, $(right, up)\hat{} = -1$,$(right, down)\hat{} = +1$ and $(right, left)\hat{} = undefined$.

Given the angle between two successive oriented cracks we define the orientation of a dart as the sum of the angles between the oriented cracks along its associated segment. Given a dart d in G_i the orientation of d is thus defined by:

$$or(d) = \sum_{j=1}^{n-1} (move(d_j), move(d_{j+1}))\hat{} \tag{4}$$

where $d_1 \dots d_n$ is the the sequence of initial darts encoding the segment associated to d. Note that $(move(d_j), move(d_{j+1})\hat{}$cannot be undefined for any $j \in \{1, \dots, n-1\}$ (equation 2).

The orientation of a dart may be computed on demand using equation 4 or may be attached to each dart and updated during the construction of the pyramid. Indeed, let us consider two successive double darts d_1 and d_2 at one level of the pyramid. If d_1 survives at the above level its orientation may be updated by [1]:

$$or(d_1) = or(d_1) + or(d_2) + (Lm(d_1), Fm(d_2))\widehat{} \tag{5}$$

Note that this last formula may be extended to the removal of a sequence of successive double edge.

The dart's orientation may thus be computed by fixing the orientation of all initial darts to 0 and updating the dart's orientation using equation 5 during the removal of each double edge kernel.

Let us consider a sequence $d_1 \dots d_q$ in G_i such that $d_{j+1} = \sigma_i(d_j)$ for all j in $\{1, \dots, p-1\}$ and $d_q \neq \alpha_i(d_1)$. Its orientation is defined by:

$$or(d_1 \dots d_q) = \left(\sum_{j=1}^{q-1} or(d_j) + (Lm(d_j), Fm(d_{j+1}))\widehat{} \right) + or(d_q) \tag{6}$$

The quantity $(Lm(d_q), Fm(d_1))\widehat{}$ has to be added to $or(d_1 \dots d_q)$ if the sequence defines a closed boundary. Note that $(Lm(d_j), Fm(d_{j+1}))\widehat{}$ cannot be undefined for any $j \in \{1, \dots, q-1\}$ (equation 3). Moreover, one can show that if the sequence defines a closed boundary and if $Lm(d_q) = Fm(d_1)^{-1}$, then we should have $\alpha_i(d_q) = d_1$, which is refused by hypothesis.

Using the same notations and hypothesis than equation 6, one important result shown by Braquelaire and Domenger [1] states that the orientation of a sequence $d_1 \dots, d_q$ defining a closed boundary is equal to 4 if it is traversed clockwise and -4 otherwise. Moreover, this sequence corresponds to:

- a finite face of $\overline{G_i}$ and thus a region if its orientation is equal to -4,
- a set of faces of $\overline{G_i}$ connected by bridges and included in one face if the orientation is equal to 4. Such a set of faces is called an infinite face [1].

By construction each combinatorial map G_i of a combinatorial pyramid is connected and all faces of G_i but one define a finite face. The infinite face of a combinatorial map encodes the background of the image (denoted by E in Fig. 2). The above property is used in Section 4 to compute inside and outside adjacency relationships with local calculus.

4 Computing inside relationships

As in the dual graph pyramid framework, the inclusion relationships are encoded within the combinatorial pyramid framework by self-loops: One vertex v adjacent to a vertex w and surrounded by one loop of w encodes a region included in the region associated to w.

This property is illustrated in Fig. 2 where the image is partitioned into 4 regions. The region encoded by the vertex A includes the two regions B and C and is adjacent to the region D. The vertex E encodes the background of the image. These inclusion relationships are encoded in the combinatorial map G by the loop $(1, -1)$ which surrounds vertex B and the two nested loops $(1, -1)$ and $(3, -3)$ which surround vertex C. Note that each loop corresponds to a bridge in $\overline{G}$ (Fig. 2(b)).

Let us denote the combinatorial map represented in Fig. 2 by $G = (\mathcal{D}, \sigma, \alpha)$. The σ cycle of vertex A is equal to $\sigma^*(1) = (1, 2, 3, 4, -3, 5, -1, 6, 7, 8)$. If we suppose that the loop $(1, -1)$ does not surround the vertices B and C incident to the edges $\alpha^*(2, 3, 4, -3, 5)$ but the vertices D and E incident to $\alpha^*(6, 7, 8)$ we obtain the same σ orbit $\sigma^*(1)$. This last remark shows that inside relationships cannot be decided locally without additional information. Note that this problem is not specific to the combinatorial pyramid framework. Indeed the same example may be built within the dual graph pyramid framework leading to the same drawback.

Let us consider a combinatorial map $G_i = (\mathcal{D}, \sigma_i, \alpha_i)$ defined at the level i of a combinatorial pyramid and one loop $\alpha_i^*(d)$ of G_i such that $\sigma_i^*(d) = (d, d_2, \ldots, d_{k-1}, \alpha_i(d), d_{k+1}, \ldots, d_p)$. Let us additionally consider the two sequences of darts $C_1 = (d_2 \ldots, d_{k-1})$ and $C_2 = (d_{k+1}, \ldots, d_p)$ such that $\sigma_i^*(d) = (d, C_1, \alpha_i(d), C_2)$. The loop $\alpha_i^*(d)$ surrounds either the vertices incident to $\alpha_i^*(C_1)$ or $\alpha_i^*(C_2)$. In order to differentiate these two configurations we say that d is the starting dart of the loop in the former case and the ending dart in the later one. Note that since $\alpha_i^*(d)$ defines a bridge in $\overline{G_i}$ both C_1 and C_2 define closed boundaries. Moreover, since G_i does not contain redundant edges, we have $d_2 \neq \alpha_i(d_{k-1})$ and $d_{k+1} \neq \alpha_i(d_p)$. Using equation 6, we obtain after some calculus:

$$or(\sigma_i^*(d)) = or(C_1) + or(C_2) - 4 \Rightarrow or(C_1) = -or(C_2) \tag{7}$$

where $or(\sigma_i^*(d))$ denotes the orientation of the whole sequence of darts $(d, d_2, \ldots, d_p)$. Since this sequence defines a counter clockwise traversal of the face its orientation is equal to -4 (Section 3).

(a) G (b) $\overline{G}$

Fig. 2. *One partition composed of 4 regions with two included ones*

```
1   list starting_dart(combi_map G_i, dart d_1) {
2       list L=∅
3       stack P
4       for each dart d_k in σ_i^*(d) = (d_1, ..., d_p){
5           if(d_k is a loop) {
6               if(P is empty or α_i(d_k) is not on the top of the stack P)
7                   push d_k and or_k in P
8               else {// α_i(d_k) on top of the stack P
9                   let C_1 be the sequence of darts between α_i(d_k) and d_k
10                  computes or(C_1) using equation 8
11                  if(or(C_1) == 4)  L = L ∪ {α_i(d_k)} else L = L ∪ {d_k}
12              }
13      }
14      return L
15  }
```

Algorithm 1: *Determination of the starting darts of the loops*

Equation 7 may be interpreted as follows: The loop $\alpha_i^*(d)$ corresponds to a bridge in $\overline{G_i}$ the removal of which splits the combinatorial map into two connected components. The component encoding the including face is traversed counter-clockwise and have thus an orientation equals to -4. On the other hand the remaining component corresponds to the included regions and has an opposed orientation equals to 4. Therefore, given the orientations of C_1 and C_2, d is the starting dart of the loop if the orientation of C_1 is equal to 4. Otherwise $\alpha_i(d)$ is the starting dart of the loop and d the ending one.

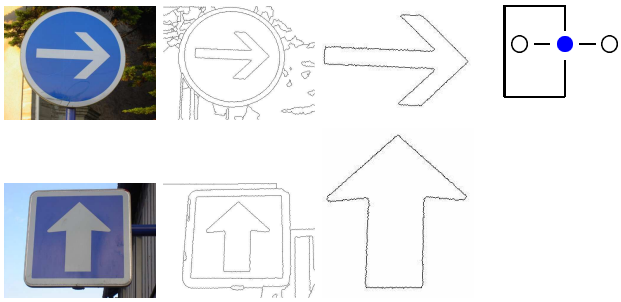
This last result is the basis of Algorithm 1 which traverses the σ_i cycle of a given vertex $\sigma_i^*(d_1) = (d_1, \ldots, d_p)$ and computes at each step the orientation of the sequence $d_1 \ldots, d_k$ denoted by or_k(equation 6).

Let us consider a dart d_k in $\sigma^*(d_1)$ such that $\alpha_i^*(d_k)$ corresponds to a loop and $\alpha_i(d_k) = d_j$ has been previously encountered ($j < k$). Then since the loops are nested d_j should be on the top of the stack. Using equation 6, if we denote by C_1 the sequence of darts between d_j and d_k, its orientation may be retrieved from or_k and or_j by the following formula:

$$or(C_1) = or_k - or_j - (Lm(d_j), Fm(d_{j+1}))\hat{} + (Lm(d_{k-1}), Fm(d_{j+1}))\hat{} \quad (8)$$

The darts d_j, d_{j+1} and d_{k-1} may be retrieved from the current dart d_k by: $d_j = \alpha_i(d_k)$; $d_{j+1} = \sigma_i(d_j)$ and $d_{k-1} = \sigma_i^{-1}(d_k)$. Given equation 8, the algorithm determines from the sign of $or(C_1)$ the starting dart of the loop between d_j and d_k (line 11). This starting dart is added to a list returned by the algorithm. Note that equation 8 is evaluated in constant time since or_k is the current orientation and or_j is retrieved from the stack.

Given the list of starting darts determined by Algorithm 1, the set of vertices included in $\sigma_i^*(d_1)$ is retrieved by traversing, the sequence $\sigma_i^*(d_1)$ from



(a) (b) (c) (d)

Fig. 3. Extraction of symbols within roadsigns using inside/outside information.

each starting dart to the corresponding ending one. By construction all darts encountered between the starting and ending darts of the loop encode adjacency relationships to included vertices. Note that in case of nested loops some loops may be traversed several times. Given a starting dart d, this last drawback may be avoided by replacing any encountered starting dart by its α_i successor during the traversal from d to $\alpha_i(d)$.

Our algorithm, is thus local to each vertex and the method may be applied in parallel to all the vertices of the combinatorial map G_i. Given a vertex $\sigma_i^*(d_1)$, the determination of its starting darts requires to traverse once $\sigma_i^*(d_1)$. Moreover, the determination of the inside relationships from the list of starting darts requires to traverse each dart of $\sigma^*(d_1)$ at most once. The worse complexity of our algorithm is thus equal to $\mathcal{O}(2|\sigma_i^*(d_1)|)$.

Fig. 3 illustrates one application of the inside/outside information to image analysis. The roadsign represented in Fig. 3(a) are composed of only two colors with one symbol inside a uniform background, the background itself being surrounded by one border with a same color than the symbol. In our example, the two roadsigns have a uniform blue background which includes one symbol representing a white arrow. The blue background is surrounded by a white border. In this application we wish to extract the sign of the roadsign using only topological and color information (and thus independently of the shapes of the symbol and the roadsign). Using only adjacency and color information, the symbol cannot be distinguished from the border of the roadsign since the border and the symbol have a same color and are both adjacent to the background of the roadsign (Fig. 3(d)). However, using inside/outside information, the symbol and the border may be distinguished since the background of the roadsign is adjacent to the border but includes the symbol. Our algorithm first builds a combinatorial pyramid using a hierarchical watershed algorithm [5]. Fig. 3(b) represents the top level of the hierarchies obtained from the two roadsigns. Using the top level combinatorial map of each pyramid our algorithm selects the k most blueish regions of the partition (k is fixed to five in our experiment). This last

step defines a set of canditate regions for the background of the roadsign. This background is then determined as the region whose included regions have the closest mean color from the color's symbol (equal to white in this experiment). Note that this step removes from the k selected candidates any regions which do not include another region. We thus explicit the a priori knowledge that the background of the roadsign should includes at least one region. The symbol is then determined as the set of regions included in the selected region (Fig. 3(c)). The symbol of the roadsign may thus be over segmented or composed of several disconnected regions. Finally, let us note that the includes information needs to be computed only on the k selected candidates for the roadsign's background. Within this experiment a global algorithm computing the includes information for all vertices would require useless calculus.

5 Conclusion

The method presented in this paper allows to get the set of regions inside in a given one. This method uses the orientation of the plane explicitly encoded by combinatorial maps and is particularly suited for algorithms using occasionally the inside information. Its worse complexity is equal to twice the number of edges incident to the vertex for which the inclusion relationships are computed. In our future work we plan to design a more global algorithm getting the inclusion information for all vertices.

References

1. J. P Braquelaire and J.P Domenger. Geometrical, topological, and hierarchical structuring of overlapping 2-d discrete objects. *Computers & Graphics*, 21(5):587–597, September 1997.
2. Luc Brun. *Traitement d'images couleur et pyramides combinatoires.* Habilitation à diriger des recherches, Université de Reims, 2002.
3. Luc Brun and Walter Kropatsch. Combinatorial pyramids. In Suvisoft, editor, *IEEE International conference on Image Processing (ICIP)*, volume II, pages 33–37, Barcelona, September 2003. IEEE.
4. Luc Brun and Walter Kropatsch. Receptive fields within the combinatorial pyramid framework. *Graphical Models*, 65:23–42, 2003.
5. Luc Brun, Myriam Mokhtari, and Fernand Meyer. Hierarchical watersheds within the combinatorial pyramid framework. In *Proc. of DGCI 2005.* IAPR-TC18, LNCS, 2005. to be published.
6. Walter G. Kropatsch. Building Irregular Pyramids by Dual Graph Contraction. *IEE-Proc. Vision, Image and Signal Processing*, Vol. 142(No. 6):pp. 366–374, December 1995.
7. P. Meer. Stochastic image pyramids. *Computer Vision Graphics Image Processing*, 45:269–294, 1989.
8. Annick Montanvert, Peter Meer, and Azriel Rosenfeld. Hierarchical image analysis using irregular tessellations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):307–316, APRIL 1991.