Rooted Kernels and Labeled Combinatorial Pyramids *

Jocelyn Marchadier†, Luc Brun‡ and Walter Kropatsch†

† Pattern Recognition and Image Processing Group, Vienna University of Technology

Favoritenstraß e 9/1832, A-1040 Vienna, Austria

‡ LERI, Universite de Reims

IUT Léonard de Vinci

Rue des Crayères, 51100 Reims

e-mail: ‡{jm, krw}@prip.tuwien.ac.at, † luc.brun@univ-reims.fr

Abstract

An irregular pyramid consists of a stack of successively reduced graphs. Each smaller graph is deduced from the preceding one using contraction or removal kernels. A contraction (resp. removal) kernel defines a forest of the initial (resp. dual) graph, each tree of this forest being reduced to a single vertex (resp. dual vertex) in the reduced graph. A combinatorial map encodes a planar graph thanks to two permutations encoding the edges and their orientation around the vertices. We present in this article a new definition of contraction and removal kernels which allows to encode the different values attached to a given vertex, dual vertex or edge along the pyramid.

1 Introduction

Irregular pyramids are widely used in image segmentation and image analysis frameworks to encode a hierarchy of partitions [10, 12, 8]. These pyramids are defined as a stack of successively reduced graphs. Each graph is built from the graph below by selecting a set of vertices named surviving vertices and mapping each non surviving vertex to a surviving one [10, 12, 8]. If the initial graph is planar its reduced versions are also planar. Moreover, given an image, if each vertex of the initial graph is associated to one pixel, the set of initial vertices mapped to a surviving vertex defines a region of the image.

The boundaries between two adjacent regions are encoded by the edges of the reduced graphs. Using simple graphs (without multiple edges between vertices nor self-loops) multiple boundaries between two regions are mapped into only one edge. This drawback may be

*This work was supported by the Austrian Science Foundation (FWF) under grants P14445-MAT and P14662-INF.

overcome by using the Dual graph reduction scheme [12]. Within this framework, the reduction operation is performed in two steps: First, the contraction of a set of edges identifies a set of vertices. The set of contracted edges, named a contraction kernel, has to form a forest of the initial graph. Each tree of this forest is then reduced to a single vertex in the reduced graph. The contraction of the edges contained in a contraction kernel may create redundant edges such as empty self-loops or double edges [12]. These redundant edges are characterized in the dual of the graph and removed by a set of edge removals encoded by a removal kernel. This removal kernel is a forest of the the dual graph and each tree of this forest is reduced to a single vertex in the dual graph. Using such a reduction scheme each edge in the reduced graph corresponds to one boundary between two regions. Moreover, inclusion relationships may be differentiated from adjacency ones in the dual graph.

Combinatorial Pyramids [3, 6] may be defined as a stack of successively reduced combinatorial maps. These pyramids inherit all the useful properties from the dual graph pyramids with the addition that they also preserve the local orientation of edges around vertices and faces. Moreover, each combinatorial map of a combinatorial pyramid may be associated to a topological data structure called a numbered simplicial set [14]. Each combinatorial map of a combinatorial pyramid may thus be interpreted using a well known topological model and a robust theoretical background from combinatorial topology [1].

Using either Dual Graphs or Combinatorial pyramids each tree of a contraction kernel is reduced to a single vertex in the reduced graph. However, the trees composing a kernel are non rooted ones. Therefore, if each vertex is identified by a unique label, the label of the vertex which represents the tree in the reduced graph is selected randomly from the labels of the tree in the graph below. This last property may be a severe drawback of the pyramid construction scheme if we want to attach geometrical properties to vertices. Moreover, the same problem holds for a removal kernel. For example, in one of our project we have to detect manufactured objects made of long lines intersecting at corners. Our segmentation algorithm should thus detect these lines and preserve dual vertices encoding their intersection. The use of an oriented kernel will allow us to preserve the label of these dual vertices and thus to retrieve their geometrical embedding from the implicit association between the label of the initial dual vertices and their coordinates in the initial image.

The notion of rooted kernels presented in this paper allows us to overcome the above drawback. Moreover, using rooted kernels the attributes associated to a given element (vertex, edge or dual vertex) may be stored for each level of the pyramid where this element survives and retrieved efficiently from the base level combinatorial map.

The remaining of this paper is as follows: We present in section 2 the combinatorial map model and a consistent labeling of its permutations. In section 3 we define the contraction and removal operations within the combinatorial map framework. The notion of rooted kernel is finally presented in Section 4 together with its main properties.

2 Consistent labeling of Combinatorial Maps

Combinatorial maps and generalized combinatorial maps define a general framework which allows to encode any subdivision of nD topological spaces orientable or non-orientable with or without boundaries. An exhaustive comparison of combinatorial maps with other boundary representations such as cell-tuples and quad-edges is presented in [13]. Recent trends in combinatorial maps apply this framework to the segmentation of 3D images [2] and the encoding of hierarchies [3].

The remaining of this paper will be based on 2D combinatorial maps which we simply call combinatorial maps. A combinatorial map may be seen as a planar graph encoding explicitly the orientation of edges around a given vertex. Figure 1(a) demonstrates the derivation of a combinatorial map from a plane graph $G = (V, E)$ corresponding to a 3×3 pixel grid. First the edges of E are split into two half edges called *darts*, each dart having its origin at the vertex it is attached to. The fact that two half-edges (darts) stem from the same edge is recorded in the reverse permutation α. A second permutation σ encodes the set of darts encountered when turning counterclockwise around a vertex.

A combinatorial map is thus defined by a triplet $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, where $\boldsymbol{\mathcal{D}}$ is the set of darts and σ, α are two permutations defined on $\boldsymbol{\mathcal{D}}$ such that α is an involution:

$$\forall d \in \boldsymbol{\mathcal{D}} \quad \alpha^2(d) = d \tag{1}$$

If the darts are encoded by positive and negative integers, the involution α may be implicitly encoded by the sign (Figure 1(a)).

Given a dart d and a permutation π, the π-cycle of d denoted by $\pi^*(d)$ is the series of darts $(\pi^i(d))_{i \in \mathbb{N}}$ defined by the successive applications of π on the dart d. The σ and α cycles of a dart d will be respectively denoted by $\sigma^*(d)$ and $\alpha^*(d)$.

Given a combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, its dual is defined by $\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha)$ with $\varphi = \sigma \circ \alpha$. The cycles of the permutation φ encode the set of darts encountered when turning around a face of G. Note that, using a counter-clockwise orientation for permutation σ, each dart of a φ-cycle has its associated face on its right (see e.g. the φ-cycle $\varphi^*(1) = (1, 8, -3, -7)$ in Figure 1(a)).

Figure 1(a) illustrates the encoding $G = (\{\bullet\}, \{\bullet - \bullet\})$ of a 3×3 4-connected discrete grid by a combinatorial map. The dual combinatorial map $\overline{G} = (\{\blacksquare\}, \{\blacksquare - \blacksquare\})$ is shown in Figure 1(b). Each vertex of the initial combinatorial map (Figure 1(a)) encodes a pixel of the grid. The σ-cycle of one vertex encodes its adjacency relationships with neighboring vertices. Let us consider the dart 1 in Figure 1(a) (see also Figure 1(c)). Turning counterclockwise around the central vertex we encounter the darts $1, 13, 24$ and 7. We have thus $\sigma(1) = 13$, $\sigma(13) = 24$, $\sigma(24) = 7$ and $\sigma(7) = 1$. The σ-cycle of 1 is thus defined as $\sigma^*(1) = (1, 13, 24, 7)$. The permutation α being implicitly encoded by the sign in Figure 1, we have $\alpha^*(1) = (1, -1)$.

Using our construction scheme, each vertex of the initial combinatorial map is associated to one pixel and thus to an open square centered on the coordinates of the pixel (see Figure 1(b)). In the same way, each vertex of the dual combinatorial map may be associated to a corner of a pixel. The top-left corner of the pixel represented in Figure 1(d) is for example encoded by the φ cycle : $\varphi^*(24) = (24, -13)$. Moreover, each dart may be understood in this combinatorial

map as an oriented crack, i.e. as a side of a pixel with an orientation. For example, the dart 1 in Figure 1(b) encodes the right side of the upper-left pixel oriented from bottom to top (Figure 1(d)). The φ, α and σ cycles of a dart may thus be respectively understood as elements of dimensions 0, 1 and 2.

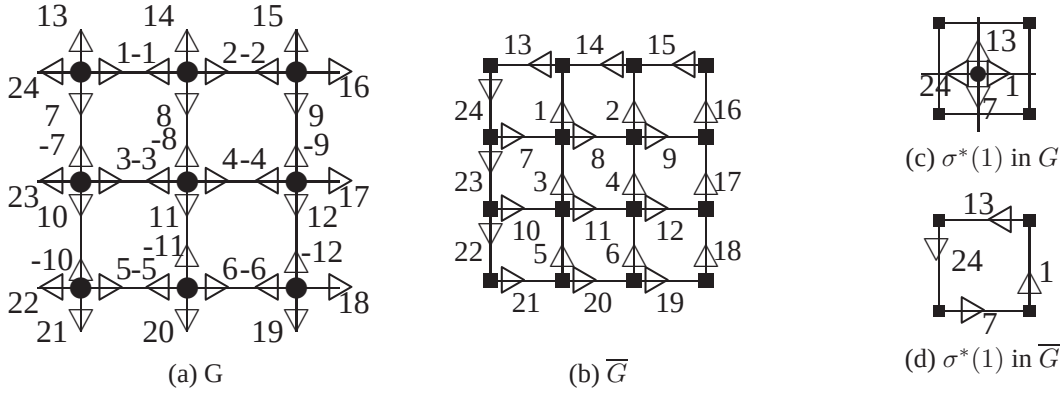

Figure 1: A 3×3 grid encoded by a combinatorial map G(a) and its dual $\overline{G}$(b). The σ-cycle $\sigma^*(1)$ encoding the top-left pixel is represented in the initial combinatorial map G (c) and its dual $\overline{G}$ (d).

A cellular complex [11] $C = (E, B, dim)$ is defined by a set E of abstract elements called cells, one bounding relation between cells B and one function dim which associates a dimension to each cell. The functions B and dim must satisfy the following relation :

$$\forall (e, e') \in E^2 \quad (e, e') \in B \Rightarrow dim(e) < dim(e')$$

Intuitively, this relationship impose that a cell of dimension 2 (a region) must be bounded by cells of dimensions 1 (edges) or 0 (points).

A precise description of the relationships between a cellular complex [11], a numbered simplicial set [14] (Section 1) and a combinatorial map is beyond the scope of this paper and the reader is referred to [14] for a more precise study of these relationships. We can however, associates informally the elements described by a combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ to a 2D cellular complex in the following way:

$$C = (E_0 \cup E_1 \cup E_2, B, dim) \tag{2}$$

with $E_0 = \{\varphi^*(d),\ d \in \boldsymbol{\mathcal{D}}\}, E_1 = \{\alpha^*(d),\ d \in \boldsymbol{\mathcal{D}}\}, E_2 = \{\sigma^*(d),\ d \in \boldsymbol{\mathcal{D}}\}$ and $dim(v) = i$ for all $v \in E_i$, $i \in \{0, 1, 2\}$. Moreover, if we denote the permutations φ, α and σ by π_0,π_1 and π_2, a couple $(\pi_i^*(b), \pi_j^*(b'))$ belongs to B if and only if $i < j$ and $\pi_i^*(b) \cap \pi_j^*(b') \neq \emptyset$.

Using Figures 1(c) and 1(d), the $2D$ cell encoding the top left pixel is encoded by the σ-cycle $\sigma^*(1) = (1, 13, 24, 7) \in E_2$ (Figure 1(c)). This $2D$ cell is bounded by the $1D$ cells defined by $\{\alpha^*(1)$,$\alpha^*(13)$, $\alpha^*(24)$,$\alpha^*(7)\} \subset E_1$ (Figure 1(d)) and we have for example $(\alpha^*(13), \sigma^*(1)) \in B$. Finally, each $1D$ cell $\alpha^*(d)$ is bounded by the two $0D$ cells $\varphi^*(d)$

and $\varphi^*(\alpha(d))$. The $1D$ cell $\alpha^*(24)$ is for example, bounded by $\varphi^*(24) = (24, 13) \in E_0$ and $\varphi^*(-24) = (-24, 7, 23) \in E_0$. More generally, within the segmentation framework, the 2D cells of a cellular complex are associated to regions while 1D cells are associated to the region's boundaries and 0 cells are associated the intersections of different region's boundaries.

The cells are respectively encoded explicitly and implicitly within the cellular complexes and combinatorial maps formalisms. This last point constitutes one of the main practical difference between both encodings. The combinatorial map formalism allows to avoid an explicit encoding of all the cells. However, if an application needs to associate values to the cells of a given dimension this implicit encoding may become explicit using labeling functions:

Definition 1. Labeling function

A labeling function l associated to a permutation π defined on $\mathcal{D}$ is a function from $\mathcal{D}$ to a set of labels $\mathcal{L}$ constants on each cycle of π:

$$\forall (d, d') \in \mathcal{D}^2 \quad l(d) = l(d') \Leftrightarrow \pi^*(d) = \pi^*(d')$$

Note that a labeling function on a permutation π may be simply defined by setting an order on the cycles of π: All the darts of the first cycle are labeled by 0, then all the darts of the second cycle by 1 until the last cycle of π. Let us denote by l_0, l_1 and l_2 three labeling functions defined in this way on the permutations φ, α and σ of a combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$.

A labeling function associates a unique value to each cycle of a permutation. Since the $0, 1$ and 2 cells of a combinatorial map are respectively defined by the cycles of the permutations φ, α and σ, the labeling functions l_0, l_1 and l_2 associate a unique label to each cell and encode thus respectively the cells of dimensions 0, 1 and 2 of a combinatorial map.

Let us consider the labeling function l_2 which maps all the darts of a cycle $\sigma^*(d)$ to a unique label which identifies the vertex and thus the associated region. The number of darts of a cycle $\sigma^*(d)$ is equal to the number of boundaries surrounding the associated region. The definition of a labeling function within the combinatorial map framework is thus quite different from a connected component labeling which maps all the pixels of a given region to an unique value. Moreover, one should note that all the labeling functions may not need to be encoded. For example, within the segmentation framework, if an application uses only region's features, only the labeling function l_2 associated to 2D cells and thus to regions needs to be encoded.

3 Combinatorial Pyramids

As in the dual graph pyramid scheme [12] (Section 1) a combinatorial pyramid is defined by an initial combinatorial map successively reduced by a sequence of contraction or removal operations. In order to preserve the number of connected components of the initial combinatorial map, we forbid the removal of bridges and the contraction of self-loops. Such contractions may be avoided by using a contraction kernel defined as a forest of the initial combinatorial map. Given a contraction kernel defined by a set K of darts to be contracted, the set of surviving darts denoted by $\mathcal{SD}$ is equal to $\mathcal{D} - K$ where $\mathcal{D}$ denotes the initial set of darts. In the same way, the removal of bridges may be avoided by using a removal kernel defined as a forest

1. set both d and $\alpha(d)$ as belonging to K in the boolean array of darts.
2. Given $r_1 = root(l_2(d))$ and $r_2 = root(l_2(\alpha(d)))$ select randomly one of the root (say r_1) as being the root of the merged tree. The union of the two trees is performed by setting $Father[r_2]$ to r_1.

Figure 2: Update of our data structures after the insertion of the edge $\alpha^*(d)$ in the kernel K.

of the dual combinatorial map. Note that, while a contraction kernel is application dependent a removal kernel may be automatically defined from one combinatorial map. Indeed, within our reduction scheme a contraction kernel specifies a set of regions to be merged while a removal kernel is restrained to the removal of double edges and empty self-loops defining redundant boundaries between the merged regions.

Several methods [6] have been proposed to define a contraction kernel. However, in all these methods a contraction kernel K may be encoded using both a boolean array of darts to encode the edges which belong to K and an array $Father$ of vertices to encode the father of each contracted vertex within its tree. Each vertex is initially its own father and this array is initialized to $Father[v] = v$ for each vertex. Using union-find techniques [6, 7], the root of each tree may be determined from the array $Father$ in almost constant time [7].

Let us denote by $root(l_2(d))$, the root of a tree containing the vertex $l_2(d)$. The addition of an edge $\alpha^*(d)$ connecting two vertices $l_2(d)$ and $l_2(\alpha(d))$ to the contraction kernel K merges the two trees containing respectively $l_2(d)$ and $l_2(\alpha(d))$. The update of our data structures during the addition of this edge is performed using the two steps described in Figure 2.

Note that using a sequential algorithm, the construction scheme described in Figure 2 selects randomly the root of each tree according to the order defined during the traversal of the set of darts and the choice of the root made for each contracted dart.

The creation of the reduced combinatorial map $G' = (\mathcal{SD}, \sigma', \alpha)$ from a contraction or a removal kernel K is performed in parallel using the following formula [6]:

$$\forall d \in \mathcal{SD} \quad \sigma'(d) = \begin{cases} \sigma^n(d) & \text{if } K \text{ is a removal kernel} \\ \varphi^n(\alpha(d)) & \text{if } K \text{ is a contraction kernel} \end{cases} \tag{3}$$

with $n = Min\{p \in \mathbb{N}^* \mid \varphi^p(\alpha(d)) \in \mathcal{SD}\}$, if K is a contraction kernel and $n = Min\{p \in \mathbb{N}^* \mid \sigma^p(d) \in \mathcal{SD}\}$ is K is a removal kernel.

Let us assume that the kernel K is a contraction kernel and that only the labeling function l_2 is defined on the combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$. The construction of the reduced combinatorial map $G' = (\boldsymbol{\mathcal{D}}', \sigma', \alpha)$ and the labeling function l_2' associated to σ' is then performed as follows:

For each dart $d \in \boldsymbol{\mathcal{D}}$

- if $d \in \mathcal{SD} = \boldsymbol{\mathcal{D}} - K$ we set
  1. the value of $\sigma'(d)$ using equation 3,

2. the label $l'_2(d)$ of d to $root(l_2(d))$.

- if $d \in K$, we merge the values associated to $l_2(d)$ and $l_2(\alpha(d))$ to the one associated to $root(l_2(d))$. These two vertices are then removed from the vertex set.

The adaptation of the above steps to the cases where K is a removal kernel or where other labeling functions are defined on G may be easily defined using the following remarks:

1. The contraction operation merges 2D cells (vertices of G) without removing any 0 cell (vertices of $\overline{G}$). Conversely, the removal operation, merges 0 cells without removing any 2D cell.

2. If the labeling function l_1 is defined on G, the removal or the contraction of an edge $\alpha^*(d)$ implies the removal of the label $l_1(d)$.

The above steps and remarks allow us to reduce a combinatorial map using either contraction or removal kernels. Starting from an initial combinatorial map, encoding for example a 4-connected sampling grid, we can apply a sequence of contraction or removal operations defined by kernels to obtain a sequence of reduced combinatorial maps. Such a sequence constitute an explicit encoding of the combinatorial pyramid. However, we have shown [5] that the whole pyramid may be encoded by storing:

1. For each dart of the initial combinatorial map, the maximal level to which it survives in the pyramid plus one.

2. the operation (contraction or removal) applied at each level.

This encoding constitutes an implicit encoding of the pyramid. This encoding provides a compact representation of the pyramid which allows to retrieve efficiently most of its features [5]. For example, within the segmentation framework, the retrieval of the combinatorial map encoding a partition at a given level of the pyramid is performed in a time proportional to the length of the boundaries associated to this partition.

Let us consider a contraction kernel K and a tree $\mathcal{T}$ of K. Using our construction scheme, the label of the vertex corresponding to $\mathcal{T}$ in the reduced combinatorial map is selected randomly among the labels of the vertices of $\mathcal{T}$. This last property does not allow us to ensure that two reconstructions of a given level of the pyramid from the implicit encoding will provide combinatorial maps with identical labels of vertices.

This last point may be a drawback if we want to compute the different values associated to a given vertex along the pyramid.We propose in the following section, a more efficient encoding scheme based on the notion of kernel of darts rather than the usual notion of kernel of edges.

4 Rooted Kernels

Using a non rooted tree, a kernel is defined as a set of edges encoding a forest of a combinatorial or its dual. Therefore, given any dart d of a kernel K, the other dart of the same edge: $\alpha(d)$ should also belong to K (see Figure 2, step 1).

(a) non rooted tree

(b) rooted tree

Figure 3: A rooted and a non rooted tree of a kernel.

Given a kernel K and one edge $\alpha^*(d)$ to be contracted, the basic idea of a rooted kernel is to adds only one of the darts d and $\alpha(d)$ to K. The orientation of the edge provided by the darts encodes a father-child relationship between the two vertices connected by $\alpha^*(d)$ (Figure 3). If K is a contraction kernel, we assume that a dart d belonging to K encodes the fact that $\sigma^*(d)$ is a child of $\sigma^*(\alpha(d))$ in the hierarchy. For example, in Figure 3, the vertices $\sigma^*(1), \sigma^*(-3)$ and $\sigma^*(-2)$ are the child of $\sigma^*(-1)$ which contains the dart $-1 = \alpha(1), 3 = \alpha(-3)$ and $2 = \alpha(-2)$. In the same way given a removal kernel K, a dart $d \in K$ encodes the fact that $\varphi^*(d)$ is a child of $\varphi^*(\alpha(d))$.

Given a kernel K, a rooted kernel K' associated to K must contain only one of the two darts of each edge $\alpha^*(d) \subset K$. This last condition is however not sufficient to insure that a rooted kernel defines a valid hierarchy. We must additionally requires that each vertex belonging to a rooted tree has at most one father within this tree. This condition may be expressed as follows within the combinatorial map formalism:

- If K is a rooted contraction kernel: $\forall d \in K \quad |\sigma^*(d) \cap K| \leq 1$ where $|.|$ denotes the cardinal of the set.
- If K is rooted removal kernel: $\forall d \in K \quad |\varphi^*(d) \cap K| \leq 1$

The root of the tree is defined as the vertex having children within the tree but no father. In other word, if K is a rooted contraction kernel, a vertex $\sigma^*(d)$ is the root of a tree if and only if:

$$|\sigma^*(d) \cap K| = 0 \text{ and } \exists d' \in K \mid \alpha(d') \in \sigma^*(d)$$

For example, using Figure 3, the rooted contraction kernel is defined by $K = \{1, -2, -3, 4, 5\}$ and we have $\sigma^*(-2) \cap K = \{-2\}$. The vertex $\sigma^*(-2)$ belongs thus to one tree of the contraction kernel but is not the root of this tree since $|\sigma^*(-2) \cap K| \neq 0$. On the other hand, $\sigma^*(-1) \cap K = 0$ while $\alpha(1) = -1 \in \sigma^*(-1)$ with $1 \in K$. The vertex $\sigma^*(-1)$ is thus the root of one tree of K. Note that that same considerations hold with a rooted removal kernel by simply replacing the permutation σ by the permutation φ.

A rooted kernel may thus be considered as an usual kernel on which we have selected a root for each tree of the forest and selected the darts encoding the father-child relationships

accordingly. Therefore, all results obtained for usual kernels [4, 3] may be easily adapted for rooted ones. We have especially the following properties:

1. Any rooted kernel K may be decomposed into a family of rooted kernels $K_0, ..., K_n$.
2. Given a rooted contraction (resp. removal) kernel K_1 defined on a combinatorial map G, if G_1 denotes the reduced combinatorial map deduced from G and K_1 and K_2 denotes a rooted contraction (resp. removal) kernel K_2 defined on G_1, the set of darts $K = K_1 \cup K_2$ is a rooted contraction (resp. removal) kernel of G.

These two properties allow us to decompose a kernel into several sub kernels or on the contrary to group several kernels into one. The size of the kernels used in the pyramid is often application dependent. For example, using the reduction scheme defined by Haximusa [9], the depth of the trees of a contraction kernel is at most one. Using such a reduction scheme the reduction of a combinatorial map requires the application of many small kernels. The above properties show us that the contracted edges encoded by these kernels may be grouped into a single equivalent contraction kernel whose reduction will provide the same reduced combinatorial map. Moreover the use of rooted kernels allows us to ensure that the labels of the vertices, dual vertices and edges will be the same using either a sequence of small kernels or an equivalent one. We can therefore use this property to decrease the height of the pyramid by removing intermediate levels. On the contrary, a single reduction step performing too many operations simultaneously may be decomposed into several kernels which induce several levels in the pyramid.

The implicit encoding of a combinatorial pyramid with rooted kernels is performed by attributing *signed* levels to darts: By convention all the darts with a level $+i$ or $-i$ belong to a non rooted kernel defined at level i while only the darts whose level is equal to $+i$ belong to the corresponding rooted kernel. The absolute value of the level of a dart corresponds thus to the usual level defined within the implicit encoding of a pyramid. The sign attribute added to the function level allows us to distinguish rooted from non rooted kernels.

5 Conclusion

In this paper, we have defined the notion of rooted kernels and proposed an encoding of such kernels within the combinatorial pyramid framework. Rooted kernels provide an elegant tool to fix the label of the cells at any level of the pyramid. Such a feature may be a decisive advantage in any application which requires an encoding of the different attributes associated to a cell (vertex, edge or dual vertex) along the pyramid. The advantages of the combinatorial pyramid framework such as the encoding of the adjacency and inclusion relationships between regions at any levels of the pyramid are preserved by this new framework.

References

[1] Max K. Agoston. *Algebraic Topology*. Marcel Dekker, 1976.

[2] Yves Bertrand, Guillaume Damiand, and Christophe Fiorio. Topological map: Minimal encoding of 3d segmented images. In Jean Michel Jolion, Walter Kropatsch, and Mario Vento, editors, 3^{rd} *Workshop on Graph-based Representations in Pattern Recognition*, pages 64–73, Ischia(Italy), May 2001. IAPR-TC15, CUEN.

[3] L. Brun and Walter Kropatsch. The construction of pyramids with combinatorial maps. Technical Report 63, Institute of Computer Aided Design, Vienna University of Technology, lstr. 3/1832,A-1040 Vienna AUSTRIA, June 2000.

[4] Luc Brun and Walter Kropatsch. Pyramids with combinatorial maps. Technical Report PRIP-TR-057, PRIP, TU Wien, 1999.

[5] Luc Brun and Walter Kropatsch. Labeled pyramids with combinatorial maps. Technical Report PRIP-TR-82, PRIP, TU Wien, 2002.

[6] 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.

[7] C. Fiorio and J. Gustedt. Two linear time union-find strategies for image processing. Technical Report 375/1994, Technische Universitat Berlin, 1994.

[8] R. Glantz and R. Englert. Representation of image structure by a pair of dual graphs. In Walter Kropatsch and J.-M. Jolion, editors, 2^{nd} *IAPR-TC-15 Workshop on Graph-based Representations*, volume 126, pages 155–163, Haindorf, Austria, May 1999. Österreichische Computer Gesellschaft.

[9] Yll Haxhimusa, Roland Glantz, Maamar Saib, Georg Langs, and Walter G. Kropatsch. Reduction Factors of Pyramids on Undirected and Directed Graphs. In Horst Wildenauer and Walter G. Kropatsch, editors, *Computer Vision - CVWW'02, Computer Vision Winter Workshop*, pages pp. 29–38, Wien, Austria, February 2002. PRIP, TU Wien.

[10] Jean Michel Jolion and Annick Montanvert. The adaptative pyramid: A framework for 2d image analysis. *Computer Vision, Graphics, and Image Processing*, 55(3):339–348, May 1992.

[11] V.A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 46:141–161, 1989.

[12] 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.

[13] P. Lienhardt. Topological models for boundary representations: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991.

[14] Pascal Lienhardt. N dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry & Applications*, 4(3):275–324, 1994.