Generalized Map Pyramid for Multi-level 3D Image Segmentation

Carine Grasset-Simon and Guillaume Damiand

SIC - Université de Poitiers
bât. SP2MI, Bvd M. et P. Curie
BP 30179, 86962 Futuroscope Chasseneuil Cedex - France
{simon, damiand}@sic.univ-poitiers.fr

Abstract. Graph pyramids are often used to represent an image with various levels of details. Generalized pyramids have been recently defined in order to deal with images in any dimension. In this work, we show how to use generalized pyramids to represent 3D multi-level segmented images. We show how to construct such a pyramid, by alternating segmentation and simplification steps. We present how cells to be removed are marked: by using an homogeneous criterion to mark faces and the cell degree to mark other cells. When the pyramid is constructed, the main problem consists in retrieving information on regions. In this work, we show how to retrieve two types of information. The first one is the low level cells that are merged into a unique high level cell. The second one is the inter-voxel cells that compose a given region.

Keywords: Irregular image pyramid, Inter-voxel elements, Generalized map, Hierarchical segmentation.

1 Introduction

To segment an image consists in partitioning this image into homogeneous connected regions given a criterion. A classical approach to region segmentation is the *split-and-merge* method and all its variants: a *bottom-up* approach [1, 2] consists in taking small regions and merging them into bigger and bigger regions; a *top-down* approach [3, 4] is the opposite one, starting from big regions and cutting them into smaller and smaller regions; a *mixed* approach [5, 6] consists in combining the two previous ones.

For bottom-up approaches, it is important to be able to extract information on regions (for example mean, variance, ...), and to be able to retrieve adjacent regions of a given region. Graph based structures allow to retrieve such information, and this is why they are often used in image processing. However such structures have several drawbacks: they do not represent all the topological information and all the cells (the volumes representing the regions of the image and the faces, edges and vertices composing the region boundaries).

To solve these problems, structures based on combinatorial maps have been defined [7, 8]. These structures have several advantages over graphs:

- they represent topological information, such as multi-adjacency or inclusion relations;
- they represent all the cells of the represented objects, and not only the regions as in the region adjacency graph;
- they allow to retrieve inter-voxel elements that composed the regions of the image and thus allow to compute geometrical features;
- they allow to compute topological characteristics of image regions.

Moreover, it is often necessary to represent a same image with different segmentation levels. For that, classical structures are extended in hierarchical structures [9,10] in order to be able to represent a same object with different resolutions. In this work, we use a 3-G-map pyramid in order to deal with a 3D multi-level segmented image. Considered images are in grey level, and the segmentation method is a bottom-up approach based on a very simple criterion that uses the squared error.

The 3-G-map pyramid used in this work is similar to the one presented in [11]. With this structure, we represent various partitions of a same image as well as links between the levels in order to be able to run through the pyramid. Moreover, each cell and each adjacency and incidence relation are represented for each different level. This information allows multi-level operations, such as for example a local modification of a region at a given level, with propagations on neighbor levels in order to keep the coherence of the structure.

In order to compute topological or geometrical features on regions of the image, it is often necessary to retrieve:

- which regions of a fine segmentation (a segmentation composed of many small regions) were merged in a unique region at a higher level in the pyramid;
- inter-voxel elements in the initial image that composed a given cell of a given region in the pyramid (e.g. the voxels of a region or the surfels of a face).

In this work, we show that this information can be retrieved in the 3-G-map pyramid by using the notion of generalized cells (particular cases of generalized orbits defined in [12]). The main result of this work is the definition of the algorithms which allow to retrieve all the information concerning a given region.

This paper is organized as follows. Section 2 provides some recalls about pyramids of n -dimensional generalized maps and about the notion of generalized cells. In section 3 we examine the construction of the n -G-map pyramid representing different segmentation levels of a same image in grey level. In section 4 we show how to retrieve voxels and inter-voxel elements which compose a region at a higher level. Conclusion and further issues are discussed in section 5.

2 Recall: Pyramid of n -Dimensional Generalized Maps and Generalized Cells

An n -dimensional generalized map (n -G-map) allows to represent the topology of n -dimensional objects. For example a 3-G-map can represent the topology of

a 3D image. An n -G-map is a set of abstract elements, called darts, together with $n + 1$ involutions defined on this set of darts (an involution f on a finite set S is a one to one mapping from S onto S such as $f = f^{-1}$). Each involution α_i represents an adjacency relation between i -dimensional cells (c.f. Fig. 1 and definition in [13]).

The different *cells* of an image such as pointel, linel, surfel and voxel (or in general vertex, edge, face and volume corresponding to 0, 1, 2, and 3-dimensional cells) are implicitly represented as subset of darts by using the orbit notion. Intuitively, an orbit $\langle f_1, \dots, f_k \rangle (d)$ is the set of darts that we can reach by a breadth first search algorithm starting from d and using any application f_i or f_i^{-1} . Each i -cell is defined by a particular orbit in n -G-maps, using all the involutions except α_i (see [13] for definition of i -cells).

The *degree* of an i -cell c is the number of distinct $(i + 1)$ -cells incident to c . For example, the degree of a vertex v is the number of edges incident to v . The *local degree* of an i -cell c is similar to the degree but computed locally without to run through the $(i + 1)$ -cells incident to c . So if an incident $(i + 1)$ -cell is incident to c twice, it is considered as two $(i + 1)$ -cells when we compute the local degree of c .

The operation of *cell removal* (defined in [14]) removes simultaneously various cells in an n -G-map. These cells can be removed if they respect two preconditions: they have to be disjoint, and the local degree of each cell is two. For removing an i -cell c , we delete the darts which form this cell and for each surviving neighbor dart of c , we redefine the value of involution α_i applied to c (see Fig. 1) in order to jump over the removed cells.

A *pyramid of n -G-maps* is a hierarchical data structure composed of several n -G-maps, where each map is a reduction deduced from the previous map (cf. de-

Fig. 1. Example of a 3-G-map pyramid composed of four levels. At each level darts are represented by black segments. Two darts linked by α_i are in the same j -cell (with $j \neq i$) but in two adjacent i -cells. In this pyramid, level 0 is composed of 2 volumes, level 1 is obtained by removing the face between the two volumes, level 2 is obtained by removing edges e_1, e_2, e_3 and e_4 , and level 3 is obtained by removing degree two vertices v_1, v_2, v_3 and v_4 . In level 2, the edges incident to d_1 and d_2 are adjacent and so linked by α_1 . This link is deduced from those of level 1 which allow to go from d_1 to d_2 by jumping removed edge e_3 . This principle is recursive. Generalized face $CG_{(2,0,3)}(d)$ (resp. $CG_{(2,1,3)}(d)$ and $CG_{(2,2,3)}(d)$) corresponding to the grey face incident to dart d at level 3 corresponds to the set of grey faces at levels 0 (resp. 1 and 2).

finition in [11]). In the particular case of a region growing segmentation method, each n -G-map is deduced from the previous level by applying the general operation of cell removal. The choice of cells to remove depends on the application and is the result of an external process. Fig. 1 illustrates an example of a 3-G-map pyramid composed of four levels. An important property of this pyramid is that a one to one mapping exists between the surviving darts of a level (the darts which are not marked to remove) and the darts of the following level.

With an n -G-map pyramid it is possible to represent geometrical information by adding attributes on cells such as vertex coordinates, or region colors. . . Then, it is necessary to be able to retrieve this information which is generally kept in the initial level.

The notion of *generalized cell* (particular case of generalized orbit defined in [12]) allows to retrieve the set of the i -cells of a level which are concerned in the formation of an i -cell c given at a higher level. This set of i -cells is composed of:

- i -cells which have been merged into c by removing incident $(i - 1)$ -cells;
- i -cells incident to these $(i - 1)$ -cells, which have been removed.

We note $CG_{(i,l',l)}(d)$ the generalized i -cell at level l' that corresponds to the i -cell incident to dart d at level l . This generalized cell is a union of i -cells at level l' (see Fig. 1). The generalized cells of dimension 0, 1, 2 or 3 are respectively named generalized vertices, edges, faces or volumes.

3 Presentation of the Pyramid

3.1 Choice of the Structure

In this work, we use a pyramid of 3-G-maps in order to realize a multi-level segmentation of a 3D image. Indeed this structure have several advantages.

First, n -G-maps are defined for any dimension of the space, their definition is homogeneous for all dimensions and so the operations defined above them are generic (in particular cell removals). They allow to represent all the cells of an image (not only the regions and their boundaries) as well as the incidence, adjacency and inclusion relations.

Second, a pyramidal structure allows to keep in memory various segmentation levels of the same image and so to work at the best level according to each operation. Since the levels of such a structure are linked between them, it allows to work simultaneously at various levels or to retrieve the set of regions of a fine segmentation which have been merged into a region of a coarse segmentation.

To facilitate the retrieval of information in the pyramid, it is important to simplify each segmentation level. This type of simplification is often used in 2D with dual graph pyramids [15] and combinatorial pyramids [16]. To add a new segmentation level, we propose to use three different steps:

- first, the merge of similar adjacent regions of the previous level;
- second, a first simplification of the boundary of each region by merging the adjacent faces incident to two same regions;

- third, the second simplification of the boundary of each region by merging the adjacent edges incident to two same faces.

In the pyramid, these three steps are achieved by removing respectively the faces, edges or vertices between the elements to merge. They are successively applied and as many times as necessary. To represent a new segmentation level, three pyramid levels are constructed and noted level 0, 1 and 2 (mod 3).

3.2 Construction of the Pyramid

The construction of a new 3-G-map is achieved by applying a removal kernel (see [17] for all algorithms constructing a pyramid) based on the following principle:

- first, the cells to remove are marked. The choice of these cells depends on the level we want to add. For example, to construct a level 0 (mod 3), the faces separating two homogeneous regions according to a segmentation criterion are marked;
- second, the new G-map is constructed by copying each surviving dart of the previous level and by linking them by taking into account disappeared darts.

For all the levels, the method used to add a new level is the same. The unique difference existing between the levels is the criterion used to mark the cells.

Marking Faces to Remove. Two adjacent regions have to be merged if their union is homogeneous according to a criterion. In our application, the homogeneity of a region R is measured with the squared error which corresponds to the sum of the square distance of each grey level to the mean grey level ν of R . This criterion can be formulated with the moments of order zero, one and two of a region:

$$EQ(R) = M_2(R) - M_0(R)\nu(R)^2$$

where $M_0(R)$ is the number of voxels contained in R , $M_1(R)$ is the sum of voxel grey levels of R , $M_2(R)$ is the sum of squared voxel grey levels of R , and $\nu(R) = \frac{M_1(R)}{M_0(R)}$. Since the moments of the union of two regions can be incrementally computed from the moments of both regions, the squared error can be efficiently updated.

Two regions have to be merged if their union is homogeneous i.e. the squared error of their union is inferior to a threshold T . Otherwise, their union is non-homogeneous and both regions should not be merged.

To obtain coarser segmentations when we go up in the pyramid, the threshold have to be increased with the levels ($T^l > T^{l-1}$ with l the level we want to add). It is possible to compute a new threshold by multiplying the previous threshold by a constant k ($T^l = T^{l-1} \times k$), or to use a different formula like $T^l = (T^{l-1})^2$ (if $T^{l-1} > 1$) depending on how we want the segmentation to evolve.

To merge all homogeneous regions, the algorithm scans twice the 3-G-map:

- it considers each face of the G-map and marks it to remove if it separates two similar regions according to the homogeneity criterion;

- it considers each not marked face and marks it if it separates two regions which will be merged in the next level. This step is necessary to avoid inner faces (i.e. faces inside a region).

Since face removal can lead to volume disconnection, a region inclusion tree is added to each level (see [18, 19] for more details on disconnection problems and possible solutions).

Marking Edges to Remove. In the first step of the boundary simplification, the edges separating two faces between the same two regions (i.e. local degree two edges) are removed. Edge removal can lead to face disconnection and object disappearance (for example in Fig. 2-b the removal of edge e_3 leads to a face disconnection, and in Fig. 2-d the removal of edge e_5 removes the representation of the cube). A way to solve this problem is not to mark an edge if its removal leads to a disconnection or a disappearance. Algo. 1 considers successively each edge and marks it if:

- its local degree is two;
- its removing does not lead to disappearance;
- its removing does not lead to disconnection.

The two first points are realized by a direct test (achieved in $\mathcal{O}(1)$), and the third point is realized by running through an orbit (cost $\mathcal{O}(f)$ with f the number of darts of the face). Note that this last test can be optimized by using a union-find tree.

When an edge is marked to remove, the algorithm reconsiders possibly incident edges. For example, in Fig. 2 we want to simplify the boundary of the region represented by the cube. If we consider edge e_3 before e_4 , we cannot mark it since its removal leads to a face disconnection. Then, when we consider e_4 , it is marked to remove, and we can see that edge e_3 can now be marked to remove without face disconnection. This is the reason why e_3 needs now to be reconsidered. Note that this case occurs only when the marked edge is adjacent to a unique not marked edge. To solve this case, we just test both extremities of the current edge after it was marked to remove. If a single not marked edge is incident to an extremity, this edge is added to a list of edges to reconsider. Note that an edge can be treated at most twice since an already treated edge can be reconsidered only when it becomes the unique edge incident to a vertex.

Marking Vertices to Remove. In the second step of the boundary simplification, the vertices separating two edges between the same two faces (i.e. degree two vertices) are removed. In this step, it is important to test the degree and not the local degree as for edges since the removal of a degree two vertex does not lead to topological modification while the removal of a local degree two vertex can lead to the disappearance of the object (in the case where the vertex is only incident to a loop). Note that the vertex removal cannot lead to a disconnection, and so this step is achieved directly without problem. Each vertex is successively considered and marked to remove if its degree is two.

Algorithm 1. Edge marking.

Data: G : a 3-G-map,
Result: G in which edges to remove are marked.
 $e \leftarrow$ an edge of G ;
while $e \neq \text{null}$ **do**
 // processing of edge e
 if *the local degree of e is 2* **then**
 if *the removing of e leads neither to disconnection nor disappearance*
 then
 mark e to remove ;
 foreach *vertex v incident to e* **do**
 if *it exists a single not marked edge e' incident to v* **then**
 add e' in list_edge_to_treat ;
 // choice of the next edge to treat
 if *list_edge_to_treat is not empty* **then**
 $e \leftarrow$ the first of list_edge_to_treat ;
 else $e \leftarrow$ an edge of G not yet treated;

Fig. 2. Example of boundary simplification for a cube. (a) The initial 3-G-map representing a cube with a boundary composed of six faces. (b) The 3-G-map obtained by removing local degree two edges e_1, e_6, e_7, e_8 and e_9 from (a). (c) The 3-G-map obtained by removing edge e_4 from (b). (d) The 3-G-map composed of two vertices, one edge and one face, which is obtained after simplification of the boundary.

3.3 The First Level

The whole pyramid is built starting from a first level, and thus the question concerning the definition of this first level is important. Indeed, this level can either represent each voxel of the image or represent a fine segmentation of this image. Note that, in the first case, the best adapted structure in order to represent a regular subdivision seems to be a matrix. For our application, we have chosen the second possibility since the initial image is not the reality but a discretization of it, and so it can contain noise. Moreover, in image analysis it is usual to make a pre-segmentation before other steps.

To compute the first segmentation starting from the image, we have used here a semi-supervised classification based on a histogram analysis, but any method can be used. This first level is built in two scans of the image:

- a first scan, in order to construct the histogram, and then the classes;
- a second scan, in order to construct incrementally the 3-G-map of the first pyramid level. In this step, the voxels are added one by one to the G-map and merged with these neighbor voxels if they are similar and already constructed.

After this first pyramid level, the construction of the next levels follows the principle explained before: add both simplification levels and possibly other segmentation levels until the desired result (or an image composed of a unique region).

4 Retrieving Regions and Inter-region Elements

When we keep in memory various segmentation levels of a same image, we often want to work simultaneously at various levels, or we want to run through the same object at various levels. For a given cell c at a level, it is necessary to be able to retrieve information: the cells of a lower level which have been merged into c , and the inter-voxel elements which represent c in the image. With this information, we are for example able to:

- modify the segmentation of a part of the image (at a given level) without reconstruct all the levels. For that we modify a given region and propagate the modifications in all the pyramid but only for concerned regions;
- compute geometric criteria, for example characteristics of face curvature by using the surfels which compose it.

The notions of cells and generalized cells defined in an n -G-map pyramid allow to retrieve this information.

4.1 Retrieving the Cells

Given a cell c at a particular level l , we want to retrieve all the cells at a lower level that are merged into c at level l .

Volumes. For a given volume V , we want to retrieve the volumes at a lower level which have been merged into V . The idea is to use generalized volumes. When there is no disconnection, the result is directly given by the generalized volume computed between the two considered levels. Otherwise, we need to make the union of generalized volumes for each boundary of each volume obtained by the initial generalized volume (each volume is represented by an external boundary and possibly several internal boundaries, one for each cavity).

Algo. 2 gives the set of volumes at a given level which have been merged into a volume V at a higher level. First, it computes the generalized volume representing V at the lower level. The obtained set corresponds to the external boundary of V . Second, the generalized volumes are computed for all the internal boundaries.

Algorithm 2. Retrieve volumes(d, l, l').

Input: d : a dart of a volume V ;
 l : the level containing V ;
 l' : the level where we want to retrieve the volumes merged into V ;
Output: the set of volumes of level l' which have been merged into V ;
 $Res \leftarrow$ the volumes incident to $CG_{(3,l',l)}(d)$;
foreach volume $V' \in Res$ **do**
 foreach internal boundary B of V' **do**
 $d' \leftarrow$ a dart of B ;
 $Res \leftarrow Res \cup CG_{(3,l',l)}(d')$;
return Res ;

Faces. For a given face F , we want to retrieve the faces at a lower level which have been merged into F . By using generalized faces, we obtain, by definition, the faces which were concerned in the formation of F . As we can see in Fig. 3, this does not give immediately the expected result since we obtain too many faces. To solve this problem, we need to progressively go down in the pyramid

Fig. 3. Comparison between a generalized face and faces merged into a face. (a) Level 3 of the pyramid of Fig. 1. Face F is in grey. (b) and (c) Level 0 of the pyramid where grey faces respectively correspond to $CG_{(2,0,3)}(d)$ and the ones merged into face F .

and add different cells depending on the current level. Algo. 3 computes the set of faces of a given level l' which have been merged into face F incident to dart d at level l . Its principle is the following:

- if $l \equiv 1$ or $2 \pmod{3}$, the set of faces which have been merged at the previous level in order to form F is the set of faces given by $CG_{(2,b-1,b)}(d)$. Indeed, the unique operations used to construct levels 1 or $2 \pmod{3}$ are the edge and vertex removal but not face removal. So by definition, the generalized face gives us directly the set of faces which have been merged into face F ;
- otherwise, $l \equiv 0 \pmod{3}$. In this case, we cannot use the generalized face since this level is obtained from the previous one by removing faces. Since between both levels, only faces have disappeared, the surviving faces have not been modified. Thus, to obtain the face corresponding to F at the previous level, we only use the existing links between the darts of F and the darts of the previous level.

The algorithm stops when the current level is l' and in this case there is only one face incident to dart d .

Algorithm 3. Retrieve faces(d, l, l').

Input: d : a dart of a face F ;
 l : the level containing F ;
 l' : the level where we retrieve the faces merged into F ;
Output: the set of faces of level l' which have been merged into F ;
if $l = l'$ **then** return the face incident to d ;
if $l \equiv 1$ or $2 \pmod{3}$ **then**
 $F' \leftarrow$ the faces incident to $CG_{(2,l-1,l)}(d)$;
else $F' \leftarrow$ the face at level $(l - 1)$ which corresponds to face F ;
foreach face $f \in F'$ **do**
 $d' \leftarrow$ a dart of f ;
 $Res \leftarrow Res \cup \text{Retrieve faces}(d', l - 1, l')$;
return Res ;

Edges. For a given edge E , we want to retrieve the edges at a lower level which have been merged into E . We use exactly the same principle than for faces. We use the generalized edge when the current level is 0 or 2 (mod 3) since the unique operation used is the face removal or vertex removal, and we use the links existing in the pyramid for level 1 (mod 3) in order to directly retrieve the corresponding edge in the previous level.

Vertices. Since vertices cannot be merged in G-map pyramids, to retrieve the set of vertices of a given level l' which have been merged into a vertex V of a higher level l , amounts to take the unique vertex V' at level l' which corresponds to V . In order to do that, we only use the bijective links allowing to go down directly to level l' , and then use the classical orbit notion in this level.

4.2 Retrieving the Inter-voxel Elements

To retrieve the voxels of the image which have been merged into a given region, or to retrieve the inter-voxel elements of an image which represent the boundary of a given region, we use the previous algorithms.

To retrieve the **surfels**, **linels** and **pointels** which respectively represent the faces, edges and vertices of the boundary of a region, it is enough to directly apply corresponding algorithms of the previous section between level l of the region and level 0. Indeed surfels, linels and pointels are directly represented in our level 0 G-map.

To retrieve the **voxels** which have been merged into a given region R , we need to:

- retrieve the regions of level 0 which have been merged into R by using Algo. 2;
- retrieve the surfels of these regions by running through their boundaries;
- use a classical flood-fill algorithm in order to reconstruct the voxels.

Retrieving voxels is more complex than for inter-voxel elements since voxels are not represented explicitly in the first pyramid level.

5 Conclusion and Perspectives

In this paper, we have presented the construction of a 3-G-map pyramid in the framework of multi-level segmentation of a 3D grey level image. Each new segmentation level is deduced from the previous level in the pyramid by applying a particular removal kernel which can use any homogeneous criterion. In order to facilitate the information retrieval, this level is simplified, and thus each new segmentation is represented by three successive levels in the pyramid. The first level is the new segmentation. The second level is obtained by removing all the degree two edges, and the third level is obtained by removing all the degree two vertices. Additional constraints are added in order to guaranty that no adjacency or incidence relation is lost during the simplifications.

We have shown how information can be retrieved in such a pyramid. We have given algorithms that allow to retrieve, given a region of a particular level, any cells that composed this region in a lower level. The methods used in these algorithms are based on the generalized orbit notion, and on the links between successive levels of the pyramid. When we are able to retrieve any cells between any levels, it is then easy to retrieve inter-voxels elements since it is just a particular case where the considered level is the first level of the pyramid.

Now, we want to study if it is possible to optimize the construction of the pyramid in order to keep only one level for each new segmentation. This construction is theoretically possible since the operation which removes simultaneously cells of various dimensions is defined in [14]. But we need to study how the generalized orbits can be used in such a case. Moreover, we are working to conceive operations for handling this pyramid. A first interesting operation consists in locally modifying a region at a given level and propagating the modifications without re-computing all the levels. After that, many other operations can be considered in order to propose a whole framework image processing.

Acknowledgements

The authors wish to thank Pascal Lienhardt for its encouragements and help.

References

1. Brice, C., Fennema, C.: Scene analysis using regions. *Artificial Intelligence* **1**(3-4) (1970) 205–226.
2. Fiorio, C., Gustedt, J.: Two linear time union-find strategies for image processing. *Theoretical Computer Science* **A: 154**(2) (1996) 165–181.
3. Lee, C.: Recursive region splitting at hierarchial scope views. *Computer Vision, Graphics, and Image Processing* **33**(2) (1986) 237–258.

4. Ohlander, R., Price, K., Reddy, R.: Picture segmentation by a recursive region splitting method. *Computer Graphics and Image Processing* **8** (1978) 313–333.
5. Pietikainen, M., Rosenfeld, A., Walter, I.: Split-and-link algorithms for image segmentation. *Pattern Recognition* **15**(4) (1982) 287–298.
6. Cheevasuvit, F., Maitre, H., Vidal-Madjar, D.: A robust method for picture segmentation based on a split-and-merge procedure. *Computer Vision, Graphics, and Image Processing* **34**(3) (1986) 268–281.
7. Braquelaire, J., Brun, L.: Image segmentation with topological maps and inter-pixel representation. *Journal of Visual Communication and Image Representation* **9**(1) (1998) 62–79.
8. Damiand, G., Bertrand, Y., Fiorio, C.: Topological model for two-dimensional image representation: definition and optimal extraction algorithm. *Computer Vision and Image Understanding* **93**(2) (2004) 111–154.
9. Montanvert, A., Meer, P., Rosenfeld, A.: Hierarchical image analysis using irregular tessellations. *PAMI* **13**(4) (1991) 307–316.
10. Kropatsch, W., Macho, H.: Finding the structure of connected components using dual irregular pyramids. In: *Discrete Geometry for Computer Imagery*, Clermont-Ferrand, France (1995) 147–158.
11. Grasset-Simon, C., Damiand, G., Lienhardt, P.: Pyramids of n-dimensional generalized maps. In: *proceedings of 5th Workshop on Graph-based Representations in Pattern Recognition*. Number 3434 in LNCS, Poitiers, France (2005) 142–152.
12. Grasset-Simon, C., Damiand, G., Lienhardt, P.: Receptive fields for generalized map pyramids: the notion of generalized orbit. In: *Discrete Geometry for Computer Imagery*. Number 3429 in LNCS, Poitiers, France (2005) 56–67.
13. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. In: *International Journal of Computational Geometry and Applications*. (1994) 275–324.
14. Damiand, G., Lienhardt, P.: Removal and contraction for n-dimensional generalized maps. In: *Discrete Geometry for Computer Imagery*. Number 2886 in *Lecture Notes in Computer Science*, Naples, Italy (2003) 408–419.
15. Kropatsch, W.: Building irregular pyramids by dual-graph contraction. *IEE Proceedings Vision, Image and Signal Processing* **142**(6) (1995) 366–374.
16. Brun, L., Kropatsch, W.: Receptive fields within the combinatorial pyramid framework. In: *Discrete Geometry for Computer Imagery*. Number 2301 in LNCS, Bordeaux, France (2002) 92–101.
17. Simon, C., Damiand, G., Lienhardt, P.: nd generalized map pyramids: definition, representations and basic operations. *Pattern Recognition* **39**(4) (2006) 527–538.
18. Damiand, G.: Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d. Phd thesis, Université Montpellier II, France (2001)
19. Damiand, G., Resch, P.: Split and merge algorithms defined on topological maps for 3d image segmentation. *Graphical Models* **65**(1-3) (2003) 149–167.