Generalized Map Pyramid for Multi-level 3D Image Segmentation

Carine Grasset-Simon, Guillaume Damiand

Retrieve inter-voxel elements in a 3D Generalized map pyramid for multi-level 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 for representing an image with different 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 in order to represent 3D multi-level segmented images. We show how to construct such a pyramid, by alternating segmentations and simplifications steps. When the pyramid is constructed, the main problem consists in retrieving information on regions, for example in order to compute geometrical or topological features. In this work, we show how to retrieve two types of information concerning regions: the first one is the cells of a low levels that are merged into a unique cell of a high level, and the second one is the inter-voxel cells that compose a given region. This second type of information is particularly interesting in order to retrieve for example the surfels that composed the boundary of a region.
Keywords. Irregular image pyramid, Inter-voxel elements, Generalized map, Hierarchical segmentation.

1 Introduction

To segment an image, i.e. partition the image in distinct and homogeneous connected regions given a criterion, the region based methods are the most appropriate. A classical approach to region segmentation is the *split-and-merge* method and all its variants: *bottom-up* approach [1,2] consists in taking small regions and merging them into bigger and bigger regions; *top-down* approach [3,4] is the opposite one, starting from big regions and cutting them into smaller and smaller regions; *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 [7,8,9] allow to retrieve such information, and this is why they are used in many image processing works. But such structures have several drawbacks: they do not represent all the topological information and all the cells.

To solve these problems, structures based on combinatorial maps have been defined [10,11]. These structures have many advantages:

– they represent topological information, such that 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 on the objects;
– they allow to compute topological characteristics of image regions.

Moreover, it is often necessary to represent a same image with different segmentation level. For that, classical structures are extended in hierarchical structures 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 use the squared error.

3-G-Map pyramid used in this work is based on similar principle to the one presented in [12]. With this structure, we represent different partitions of a same image, and links between the levels in order to be able to run through the pyramid. Moreover, each cells and adjacency and incidence relations are represented for each different level. These information allow multi-level operations, such that for example a local modification of a region in 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 were merged in a unique region of a partition in an upper level in the pyramid;
– inter-voxels elements in the initial image that composed a given cell of a given region in the pyramid (for example the voxels of a region or the surfels of a face).

In this work, we show that these information can be retrieved in the 3-G-Map pyramid by using the notion of generalized cells (particular cases of generalized orbits defined in [13]). This is the main result of this work, 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 (or in general region and inter-region elements) which compose a region or the boundary of a region in an upper level. Conclusion and further issues are discussed in section 5.

2 Recall: pyramids 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 involutions[1] defined on these darts, each involution α_i representing an adjacency relation between i-dimensional cells (c.f. figure 1-a and definition in [14]).

The different cells of an image such that 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[2]. Intuitively, an orbit $< f_1, \dots f_k > (d)$ is the set of darts that we can reach by a breath 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 [14] for definition of i-cells).

The *degree* of an i-cell is the number of distinct $(i+1)$-cells incident to this cell. For example, the degree of a vertex is the number of edges incident to it. The *local degree* of an i-cell c is similar to the degree but computed locally to c 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 [15]) removes simultaneously different cells of same dimension in an n-G-map. These cells can be removed if they respect two preconditions: they have to be disjoint, and their local degree 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 involution α_i (see figure 1-b) in order to jump over the removed cell.

A pyramid of n-G-maps (or n-G-map pyramid) is a hierarchical data structure composed of several n-G-map, where each map is a reduction deduced from the previous map (cf. definition in [12]). In the particular case of a region growing segmentation method, each n-G-map is deduced from this one of the previous level by applying the general operation of cells removal. The choice of cells to remove depends on the application and is the result of an external process. Figure 1-a illustrates an example of a 3-G-map pyramid composed of 4 levels. An interesting property of this pyramid is 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 topological and geometrical information by adding attributes on cells such that coordinates of vertices, or colors of regions, ... Then, it is necessary to be able to retrieve information generally kept in the initial level.

The notion of generalized cell (particular case of generalized orbit defined in [13]) allows to retrieve the set of the i-cells of a level which have had an incidence over the formation of an i-cell c given at an upper level. This set of i-cells is composed of (see figure 1-c):

---

[1] An involution f on a finite set S is a one to one mapping from S onto S such that $f = f^{-1}$.

[2] Let $\{\Pi_0, \dots, \Pi_n\}$ be a set of permutations on D. The orbit of an element d related to this set of permutations is $< \Pi_0, \dots, \Pi_n > (d) = \{\Phi(d), \Phi \in < \Pi_0, \dots, \Pi_n >\}$, where $< \Pi_0, \dots, \Pi_n >$ denotes the group of permutations generated by $\Pi_0, \dots, \Pi_n$.

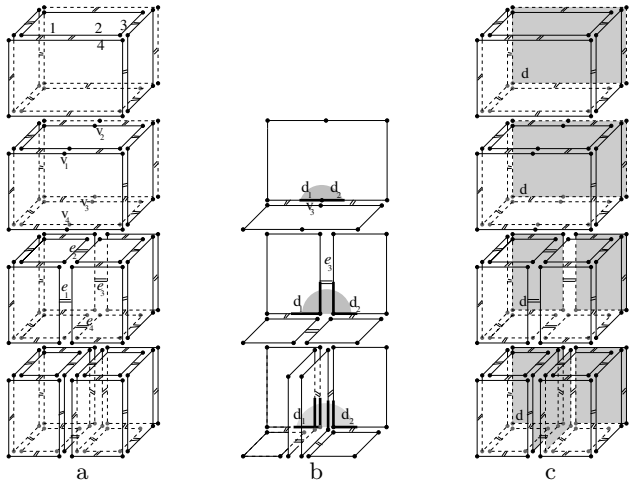

Fig. 1. (a) Example of a 3-G-map pyramid composed of four levels (level 0 at the bottom to level 3). At each level darts are represented by black segments. In this pyramid, level 0 is composed of 2 volumes, level 1 is obtained by removing the face between the two volumes, level 2 is obtained by removing 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. (b) Edges incident to d_1 and d_2 are adjacent and so linked by α_1 in level 2. This link is deduced from thus of level 1 which allow to go from d_1 to d_2 by jumping removed edge e_3. This principle is recursive. (c) 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).

- i-cells which have 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 an union of i-cells at level l'.

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 all dimension n of the space, their definition is homogeneous and so the operations defined above them are generic (in particular cell removals). It allows to represent all the cells of an image and not only the regions and their boundaries. Moreover, adjacency, incidence and inclusion relations are represented too. So it is possible to compute efficiently topological features in order to realize, for example, a segmentation with a topological criterion.

Second, a pyramidal structure allows to keep in memory different segmentation levels of a same image and so to work at the best level according to each operation. Moreover, in such a structure the levels are linked between them, and this can be useful in order to work simultaneously at different levels or to retrieve for example the set of regions of a fine segmentation which have been merged into a region of a coarse segmentation.

In order to define efficient processings and 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 [16] and combinatorial pyramids [17]. In order to add a new segmentation level in the pyramid, we propose to use three different steps:

- first, the merge of similar adjacent regions of the previous level. This is realized by removing the faces which are between them;
- second, a first simplification of the boundary of each region by merging the adjacent faces incident to two same regions. This is achieved by removing the edges which separate such faces;
- third, the second simplification of the boundary of each region by merging the adjacent edges incident to two same faces. This is achieved by removing the vertices separating these edges.

In the pyramid, these three steps are applied successively. So in order to represent a new segmentation level, three pyramid levels are constructed and in the following, we denote level $0, 1, 2 \pmod 3$ the levels of the pyramid obtained by applying respectively these three steps.

3.2 Construction of the pyramid

The construction of the 3-G-map representing a new segmentation level of the image is achieved by using the operation of cell removal. The merge of two similar adjacent regions is realized by removing the face which is in-between them. The two steps of boundary simplification, the merge of faces and edges respectively separating two same regions and two same faces are achieved by removing edges and vertices.

Each new level is added to the pyramid by applying a removal kernel 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, in order to construct a level $0 \pmod 3$, we mark faces which separate two regions homogeneous according to a criterion;
- 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 in order to add a new level is the same. The unique difference which exists between the levels concerns 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 we measure the homogeneity of a region R with the squared error. The squared error EQ of a region R corresponds to the sum of squared distance of each grey level to the mean grey level ν of R. This criterion can be formulated with 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 $\nu(R) = \frac{M_1(R)}{M_0(R)}$. By considering this second formula, the squared error of a region resulting from the merge of regions can be determined efficiently since the moments of a region can be computed incrementally:

$$\forall i \in \{0, 1, 2\}, M_i(R_1 \cup R_2) = M_i(R_1) + M_i(R_2).$$

So, in order to know if two regions have to be merged it is enough to compute the square error of their union and to compare it with a threshold T fixed by the user or computed. If the squared error of the union of two regions is inferior to threshold T, then this union is homogeneous and the two regions can be merged, otherwise, the union is non-homogeneous and the two regions cannot be merged.

In order to obtain segmentation levels more and more coarse, the threshold have to increase with the levels ($T^l > T^{l-1}$). If the user does not change it, it is possible to compute a new threshold (for example $T^l = T^{l-1} * l$ or $T^l = (T^{l-1})^2$ (if $T^{l-1} > 1$) with l the level we want to add).

In order to merge all the 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 face non marked 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).

Face removal can leads to volume disconnection and so the lost of inclusion relations In order to solve this problem we have chosen to add to the G-map an inclusion tree representing this inclusion relations (see [18,10] for more details on disconnection and possible solutions). Note that the inclusion tree not allowed to represent the interlacing information, but this is a well known problem for combinatorial structures.

Marking edges to remove In the first step of boundary simplification, two faces separating two same regions have to be merged. This is realized by removing the edge separating them. This type of edge is characterized by its local degree equal to two.

Edges removal can leads to face disconnection and object disappearance (for example in figure 2-c the removal of edge e_3 leads to a face disconnection, and the

removal of edge e_5 remove the representation of the cube). A way to solve this problem is to not allow to mark an edge if its removal leads to a disconnection or a disappearance.

Edges are marked by using algorithm 1. Each edge is considered successively and marked if:

- its local degree is two;
- its removing not leads to disappearance.
- its removing not leads 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 (coast $\mathcal{O}(f)$ with f the number of dart of the face). Note that this last test can be optimized by using a union-find tree.

Algorithm 1: Edge marking.

```
Input: G: a 3-G-map.
Output: G in which edges to remove are marked.
e ← an edge of G ;
while e ≠ null do
    // processing of edge e
    if the local degree of e is 2 then
        if the removing of e leads neither disconnection nor disappearance then
            mark e to remove ;
            foreach vertex v incident to e do
                if it exists only one non 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 ← the first of list_edge_to_treat ;
    else e ← an edge of G not yet treated ;
```

When an edge is marked to remove, the algorithm reconsiders eventually incident edges. Let we take the example of figure 2 where we want to simplify the boundary of the region represented by this 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 removed without leads to face disconnection. This is the reason why e_3 needs now to be reconsidered by the algorithm. Note that this case occurs only when the marked edge is adjacent to a unique non marked edge. To solve this case in algorithm 1, we just test both extremities of current edge after it was marked to remove, and push the incident edges in a list of edges to reconsider when they are only the unique non marked edge. Note that an edge can be treated at most twice since an edge yet treated can be reconsidered only when it becomes the unique edge incident to a vertex.

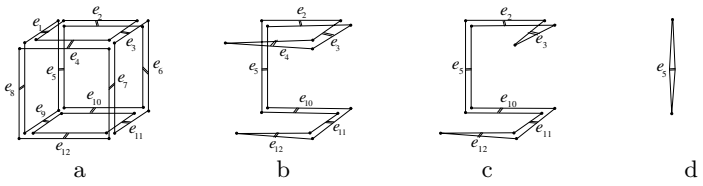

Fig. 2. Example of boundary simplification for a cube. (a) The initial 3-G-map representing a cube with a boundary composed of six faces. (b) The 3-G-map obtained if we remove local degree two edges e_1, e_6, e_7, e_8 and e_9 from the G-map of (a). (c) The 3-G-map obtained if we remove edge e_4 from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

Marking vertices to remove In the second step of boundary simplification, two edges separating two same faces have to be merged. This is realized by removing the vertex which separates them, that is to say removing a degree two vertex.

In this step it is important to test the degree and not the local degree like 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 leads to a disconnection. This step is achieved directly without problem. Each vertex is considered successively and marked to remove if its degree is two.

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. There are mainly two possibilities:

- to represent each voxels of the image,
- to 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 this one, and so it can be contain noise. Moreover, in image analysis it is common to use a pre-segmentation before to realize treatments.

To compute the first segmentation starting from the image, we have used here a semi-supervised classification based under an histogram analysis, but any method can be used. This first level is build 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 yet constructed.

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

4 Retrieving regions and inter-region elements

When we keep in memory different segmentation levels of a same image, we often want to work simultaneously at different levels, or we want to run through a same object at different levels. For a given cell c at a level, two types of informations are necessary to retrieve: the cells of a lower level which have been merged into c, and the inter-voxels element which represent c in the image. With these information, we are for example able to:

- modify the segmentation of a part of the image (at a given level) without reconstruct all the level. 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 cell defined in an n-G-map pyramid allow to retrieve these information.

4.1 Retrieve the cells

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

Volumes In order to retrieve the set of volumes at a level which have been merged into a volume at a given level, the idea is to use generalized volumes. Where 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 voolume is represented by an external boundary and eventually several internal boundaries, one for each cavity).

Algorithm 2 gives the set of volumes of a given level which have been merged into a volume V at an upper level. In a first time, it computes the generalized volume representing V at the lower level. The obtained set corresponds to the external boundary of V. In a second time, the generalized volumes are computed for all the internal boundaries.

Faces For a given face F, we want to retrieve the faces of a lower level which have been merged into F. By using generalized faces, we obtain by definition, the faces which have took a part in the formation of F: the faces which have been merged by removing edges and the faces incident to these edges which

Algorithm 2: Retrieve the volumes(d, b, b').

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: Res: the set of volumes of level l' which have been merged in order to form volume V at level b ;

```
Res ← the volumes incident to Cg_(3,l',l)(d) ;
foreach volume V' ∈ Res do
    foreach internal boundary B of V' do
        d'' ← a dart of B ;
        Res ← Res ∪ OG_(<0,1,2>,l,l')(d'') ;
return Res ;
```

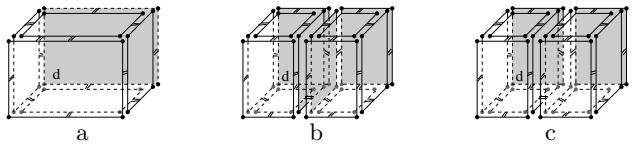

Fig. 3. Comparison between generalized face of a face and faces merged into this face. (a) Level 3 of the pyramid of figure 1. The grey face is face F. (b) Level 0 of the pyramid where grey faces corespond to $Cg_{(2,0,3)}(d)$. (c) Level 0 of the pyramid where grey faces corresponds to these ones merged into face F.

have been removed. As we can see in figure 3, this does not give imediately the wanted result since we obtain too much faces. To solve this problem, we need to progressively go down in the pyramid and add different cells depending on the current level. The principle of algorithm 3 which computes the set of faces of a given level l' which have been merged into the face F incident to dart d at level l 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 the levels 1 or 2 (mod 3) are the edges and vertices removal but not faces 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 must not 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 not have been modified. Thus we obtain the set of faces corresponding to F, we only use the existing links between the darts of F and the darts of the previous level in order to directly obtain the face at the previous level.

The algorithm stops when the current level is l' and in this case the face incident to dart d is directly given by the face orbit.

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: Res: the set of faces of level b' which have been merged in order to form face F at level l.

```
if l = l' then return < 0, 1, 3 > (d);
if l ≡ 1 or 2 (mod 3) then
    F' = set of faces incident to Cg_(2,l−1,l)(d) ;
else F' ← {the face at level (l − 1) which corresponds to face F} ;
foreach face f' ∈ F' do
    d' ← a dart of f' ;
    Res ← Res ∪ Retrieve faces(d', l − 1, l') ;
return Res ;
```

Edges In order to retrieve the set of edges of a given level which have been merged into an edge of an upper level, 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 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 of an upper level l, comes down to take the unique vertex V' at level l' which corresponds to vertex V at level l. 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 Retrieve the inter-voxel elements

In order 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 algorithms of the previous section between the 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 composing these regions by using Algo. 3,
- then to use a classical flood-fill algorithm in order to reconstruct the voxels.

Retrieving voxels is more complex than for inter-voxels 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 uses a criterion based on the squared error. 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 are lost during the simplification.

Then, 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 use the generalized orbit notion [13] plus the links between successive levels of the pyramid. When we are able to retrieve any cells between any both levels, it is then easy to retrieve inter-voxels elements since it is just a particular case where the base level is the first level of the pyramid.

Now, we want to study if it is possible to optimize our construction in order to keep only one pyramid level for each segmentation level. This construction is theoretically possible since the operation which remove simultaneously cells of different dimension is defined in [15]. 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 to locally modify a region in a given level without to re-compute all the levels.

6 Acknowledgements

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

References

1. Brice, C., Fennema, C.: Scene analysis using regions. **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) (February 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) (June 1986) 268–281
7. Montanvert, A., Meer, P., Rosenfeld, A.: Hierarchical image analysis using irregular tessellations. PAMI **13**(4) (April 1991) 307–316.
8. Jolion, J., Montanvert, A.: The adaptive pyramid : a framework for 2d image analysis. Computer Vision, Graphics and Image Processing **55**(3) (may 1992) 339–348.
9. Kropatsch, W., Macho, H.: Finding the structure of connected components using dual irregular pyramids. In: Cinquime Colloque DGCI, Université d'Auvergne (September 1995) 147–158
10. Damiand, G., Resch, P.: Split and merge algorithms defined on topological maps for 3d image segmentation. Graphical Models **65**(1-3) (May 2003) 149–167
11. Braquelaire, J., Brun, L.: Image segmentation with topological maps and interpixel representation. Journal of Visual Communication and Image Representation **9** (1998) 62–79
12. Grasset-Simon, C., Damiand, G., Lienhardt, P.: Pyramids of n-dimensional generalized maps. In: proceedings of 5th IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition. Number 3434 in LNCS, Poitiers, France (april 2005) 142–152
13. 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 (april 2005) 56–67
14. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. In: International Journal of Computational Geometry and Applications. (1994) 275–324.
15. 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 (november 2003) 408–419.
16. Kropatsch, W.: Building irregular pyramids by dual-graph contraction. IEE Proceedings Vision,Image and Signal Processing **142**(6) (December 1995) 366–374
17. Brun, L., Kropatsch, W.: Receptive fields within the combinatorial pyramid framework. In: Discrete Geometry for Computer Imagery. Number 2301 in LNCS, Bordeaux, France (april 2002) 92–101.
18. Damiand, G.: Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d. Phd thesis, Université Montpellier II, France (december 2001)