nD generalized map pyramids: definition, representations and basic operations

Carine Grasset-Simon, Guillaume Damiand, Pascal Lienhardt

nD generalized map pyramids: definition, representations and basic operations

Carine Grasset-Simon [a,*] Guillaume Damiand [a]
Pascal Lienhardt [a]

[a] *SIC FRE 2731 CNRS, bât SP2MI, BP 30179, 86962 Chasseneuil Cedex, France*

Abstract

Graph pyramids are often used for representing irregular image pyramids. For the 2D case, combinatorial pyramids have been recently defined in order to explicitly represent more topological information than graph pyramids. The main contribution of this work is the definition of pyramids of n-dimensional generalized maps. This extends the previous works to any dimension, and generalizes them in order to represent any type of pyramid built by using any removal and/or contraction operations. We give basic algorithms that allow to build an n-dimensional generalized pyramid that describes a multi-level segmented image. A pyramid of n-dimensional generalized maps can be implemented in several ways. We propose three possible representations and give conversion algorithms.

Key words: irregular pyramid, hierarchical data structure, pyramid of generalized maps, multi-level segmented image

1 Introduction

Hierarchical structures, as graph pyramids, are often used for representing image partitions at different resolution levels. Such pyramids have been widely used to encode partitions within segmentation and connected component analysis frameworks [1,2,3]. We define pyramids of n-dimensional generalized maps (or n-G-maps) for two main reasons:

* Corresponding author.
Email addresses: simon@sic.univ-poitiers.fr (Carine Grasset-Simon), damiand@sic.univ-poitiers.fr (Guillaume Damiand), lienhardt@sic.univ-poitiers.fr (Pascal Lienhardt).

Article published in Pattern Recognition, 39(4), pp. 527-538 - April 2006

- first, to represent the topological information of images at different resolution levels;
- second, to have a unique definition valid for any dimension.

So 2D, 3D or 4D images (3D plus time for instance) can be processed using a single formalism, without loss of topological information.

Topological structures are often used for representing image partitions, i.e. regions and their adjacency relations. Information described by these structures can be used for computing topological characteristics about the regions (e.g. in order to retrieve all the neighbors of a given region, during a segmentation process by region growing).

Images at different resolution levels can be represented by hierarchical structures. The first level corresponds to the highest resolution (i.e. generally to the initial image), and other levels are obtained by successive simplifications (for instance, regions at level i which are homogeneous according to a given criterion are merged at level $i + 1$). Irregular pyramids are defined as stacks of reduced graphs (e.g. adjacency graphs, dual graphs), each one being built from the previous one by a sampling or a decimation process. Such structures make it possible to:

- directly access any level;
- at the same time use information contained within different levels;
- modify a region in a particular level without having to rebuild the whole level;
- reduce the computational cost of several algorithms by applying them on reduced images;
- get a good framework for algorithms based on a divide and conquer strategy.

For image processing, the most popular hierarchical structure is the adjacency graph pyramid [1,2]. Each level is an adjacency graph, which describes regions of the image at a certain granularity level, and their adjacency relations (more precisely, a node of the graph corresponds to a region, and an edge links two nodes corresponding to adjacent regions). The problem here is the fact that all adjacency relations (simple or multi-adjacency, inclusion: cf. Fig. 1-b) are represented in the same way. Moreover, for nD images, adjacency graphs only represent n-cells (i.e. image regions) and $(n-1)$-cells (i.e. adjacency between regions): all other cells are not represented, and in the general case, they can not be computed from n and $(n-1)$-cells, leading thus to a loss of important topological information.

Dual graph pyramids [3,4] extend 2D adjacency graph pyramids. They have been defined in order to take into account multi-adjacency and inclusion relations (cf. Fig. 1-c). A dual graph is defined as a multi-graph along with its dual (these two graphs have to be connected). However, dual graphs as well

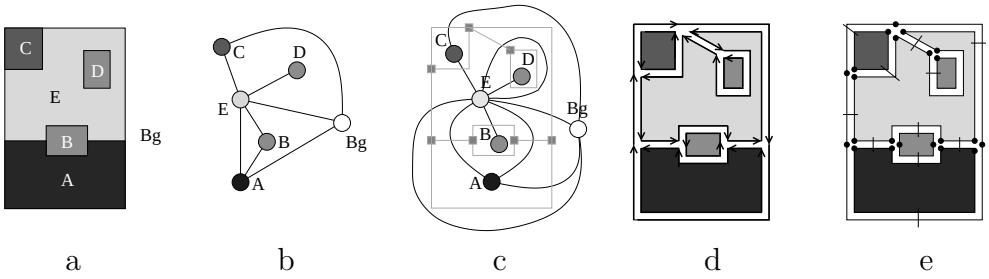


Figure 1. Representation of a segmented image. (a) An image. (b) The corresponding region adjacency graph. (c) The corresponding dual graphs (primal graph in black and its dual in grey). (d) The corresponding 2-combinatorial map. (e) The corresponding 2-G-map.

as adjacency graphs cannot represent the whole topological information. For instance, in Fig. 2, we can see that dual graphs cannot represent the order of faces around a vertex, and such information can be useful for instance to distinguish between two objects represented in a 2D image.

Brun and Kropatsch have defined combinatorial pyramids, in order to catch the whole topological information for 2D images segmented at different levels [5,6,7]. Each level of a pyramid is a combinatorial map: it can be defined as a planar graph which implicitly encodes the orientation of edges around vertices. More generally, a combinatorial map can represent the topology of any partition of any orientable surface, without loss of topological information. Note that combinatorial pyramids are 2D hierarchical structures, and that they are built in a particular way: they use only 1-removal and 1-contraction operations, and these two type of operations cannot be applied at the same time.

So, existing hierarchical structures (adjacency graph pyramids, dual graph pyramids, combinatorial pyramids) can either lead to a loss of important topological information, or they are defined only for 2D images.

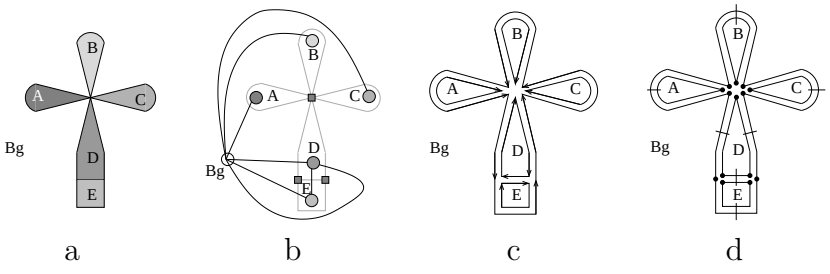


Figure 2. A 2D image of a clover. (a) The image. (b) The corresponding dual graphs. We cannot know which is the leaf between the two others. Dual graphs do not represent the topological order of edges around vertices in any case. (c) The corresponding 2-combinatorial map (d) The corresponding 2-G-map.

We define n-G-map pyramids in order to solve these problems. Generalized maps represent the topology of any partition of any n-dimensional quasi-

manifold with or without boundaries [8]. So, they can represent any partition of any nD image without loss of topological information (all cells are represented, together with all adjacency and incidence relations between cells and topological order: cf. Fig. 1-e and Fig. 2-d). Moreover, generalized maps are homogeneously defined for any dimension: this is another advantage of this structure for which the definition of generic operations and algorithms is easier. The definition of n-G-map pyramids is based upon a general operation for removing and contracting cells of any dimension [9]. More precisely, each level of a pyramid is a "reduction" of the previous one, computed by applying this operation. From this "principle" we deduce the definition of n-G-map pyramids, several useful notions and properties [10].

We are interested here in representing multi-level segmented images. So, we also propose basic algorithms for handling n-G-map pyramids which can be used for building pyramids corresponding to such images, given a segmentation criterion. Note that we do not propose a new segmentation method; our goals (and the results presented in the paper) are:

- the generic definition of n-G-map pyramids (indeed generalized map pyramids can be used in others domains, for example [11] to model complex architectural environments, or [9] to handle different representations in discrete geometry);
- the conception of basic algorithms which can be used in a segmentation process for constructing a pyramid corresponding to an nD multi-level segmented image;
- the definition of generic representations of such pyramids, and conversion algorithms between these representations.

Section 2 is a reminder of the notion of generalized map and the general operation of cell contraction and removal. Generalized map pyramids are defined in section 3. In this section, the definition is general and not devoted to image processing. We discuss in section 4 the construction of a pyramid for nD image, and we propose general algorithms for this construction. We study in section 5 three different representations of generalized map pyramids. We present in section 6 conversion algorithms between these representations. Conclusion and further issues are discussed in section 7.

2 Recalls: n-G-map, cell removal and contraction

A way to describe an image is to consider each cell of the corresponding subdivision. For instance in 3D, we represent voxels (3-cells) and also all cells of voxel boundaries [12,13,14]: surfels (2-cells between two 3-cells), linels (1-cells between two 2-cells) and pointels (0-cells between two 1-cells). n-dimensional

generalized maps (or n-G-maps) make it possible to describe this type of subdivision, and more generally any subdivision of any quasi-manifold [15]. An n-G-map is a set of abstract elements (called darts), together with applications defined on these darts.

Definition 1 (n-G-map) *Let $n \geq 0$. An n-dimensional generalized map $G = (D, \alpha_0, \ldots, \alpha_n)$ is defined by:*

(1) D a finite set of darts;
(2) $\forall k$, $0 \leq k \leq n$, α_k an involution [1] *on D;*
(3) $\forall k, j$, $0 \leq k < k+2 \leq j \leq n$, $\alpha_k \alpha_j$ is an involution.

Let G be an n-G-map, and S be the corresponding subdivision. Intuitively, a dart of G corresponds to an $(n+1)$-tuple of cells $(c_0, \ldots, c_n)$, where c_i is an i-dimensional cell that belongs to the boundary of c_{i+1} (cf. [16] and Fig. 3). α_i associates darts corresponding with $(c_0, \ldots, c_n)$ and $(c'_0, \ldots, c'_n)$, where $c_j = c'_j$ for $j \neq i$, and $c_i \neq c'_i$ (α_i swaps the two i-cells that are incident to the same $(i-1)$ and $(i+1)$-cells).

Cells are implicitly described as sets of darts through the notion of orbit (see [8] for more details).

Definition 2 (orbit and i-cell) *Let $\{\Pi_0, \ldots, \Pi_n\}$ be a set of permutations on D. The orbit of an element $d \in D$ related to this set of permutations is $< \Pi_0, \ldots, \Pi_n > (d) = \{\Phi(d), \Phi \in < \Pi_0, \ldots, \Pi_n >\}$, where $< \Pi_0, \ldots, \Pi_n >$ denotes the group of permutations generated by $\Pi_0, \ldots, \Pi_n$.*
Let $d \in D$, $N = \{0, 1, \ldots, n\}$ and let $i \in N$. The i-cell incident to d is the orbit

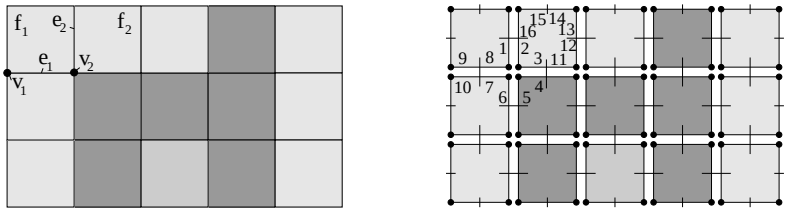
$$<>_{N-\{i\}} (d) =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (d).$$

In order to define n-G-map pyramids, Damiand and Lienhardt have defined the operation of "simultaneous removals and contractions of cells of any dimension" [9] which allows to contract and remove a set of cells of any dimension in a simultaneous way (see Fig. 4 for examples and [9] for more details). With this operation we can merge different regions (using removals of 1-cells in 2D or 2-cells in 3D), or simplify region boundaries (using removals of 0-cells in 2D or 0-cells and 1-cells in 3D).

The formal and general definition of this operation is:

Definition 3 (Simultaneous removal and contraction of any cells) *Let $G = (D, \alpha_0, \ldots, \alpha_n)$ be an n-G-map, $R_0, \ldots, R_{n-1}$ be sets of 0-cells,..., $(n-1)$-cells to be removed and $C_1, \ldots, C_n$ be sets of 1-cells,..., n-cells to be con-*

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



(a) (b)

Figure 3. (a) A 2D image. (b) The corresponding 2-G-map. Darts are represented by numbered black segments. Two darts related by α_0 share a little vertical segment (ex. darts 14 and 15). Two darts related by α_1 share a same point (ex. darts 2 and 3). Two distinct darts related by α_2 are parallel and close to each other (ex. darts 3 and 4); otherwise, the dart is its own image by α_2 (ex. dart 14). Dart 9 corresponds to (v_1, e_1, f_1), dart $8 = 9\alpha_0$ corresponds to (v_2, e_1, f_1), $1 = 8\alpha_1$ corresponds to (v_2, e_2, f_1), and $2 = 1\alpha_2$ corresponds to (v_2, e_2, f_2). The vertex (0-cell) incident to dart 2 is $< \alpha_1, \alpha_2 > (2) = \{1, 2, 3, 4, 5, 6, 7, 8\}$, the edge (1-cell) incident to dart 9 is $< \alpha_0, \alpha_2 > (9) = \{7, 8, 9, 10\}$, and the face (2-cell) incident to dart 2 is $< \alpha_0, \alpha_1 > (2) = \{2, 3, 11, 12, 13, 14, 15, 16\}$.

tracted[2]*. Let* $R = \cup_{i=0}^{n-1} R_i$ *and* $C = \cup_{i=1}^{n} C_i$*. Two preconditions have to be satisfied:*

(C1) cells are disjoint (i.e. $\forall c, c' \in C \cup R$, $c \cap c' = \emptyset$*),*
(C2) "the degree of each cell is locally[3] *two", i.e.:*
- $\forall i$, $0 \leq i \leq n-2$, $\forall d \in R_i$, $d\alpha_{i+1}\alpha_{i+2} = d\alpha_{i+2}\alpha_{i+1}$*;*
- $\forall i$, $2 \leq i \leq n$, $\forall d \in C_i$, $d\alpha_{i-1}\alpha_{i-2} = d\alpha_{i-2}\alpha_{i-1}$*.*

$\forall i \in N$*, let* $SD_i = (R_i \cup C_i)\alpha_i - (R_i \cup C_i)$ *(it is the set of surviving darts "neighbor" of removed and contracted cells). The resulting n-G-map is* $G' = (D', \alpha'_0, \ldots, \alpha'_n)$ *defined by*[4]*:*

- $D' = D - (C \cup R)$*;*
- $\forall i \in N$, $\forall d \in D' - SD_i$, $d\alpha'_i = d\alpha_i$*;*
- $\forall i \in N$, $\forall d \in SD_i$, $d\alpha'_i = d' = d(\alpha_i\alpha_{k_1}) \ldots (\alpha_i\alpha_{k_p})\alpha_i$*, where* p *is the smallest integer such that* $d' \in SD_i$*, and* $\forall j, 1 \leq j < p$*, if* $d_j = d(\alpha_i\alpha_{k_1}) \ldots (\alpha_i\alpha_{k_{j-1}})\alpha_i \in R_i$ *then* $k_j = i+1$ *else* $(d_j \in C_i)$ $k_j = i-1$*.*

[2] $R_n = \emptyset$ and $C_0 = \emptyset$ since it is not possible to remove n-cells nor to contract 0-cells
[3] The degree of an i-cell is the number of distinct incident $(i+1)$-cells. The local degree of an i-cell is its degree when we consider the cell locally. For example, the degree of a vertex incident to a loop is one but its local degree is two since locally two edges are incident to this vertex.
[4] When only removals are used (i.e. $C = \emptyset$), the definition is simplified: for $d \in SD_i$, $d\alpha'_i = d' = d(\alpha_i\alpha_{i+1})^p\alpha_i$, where p is the smallest integer such that $d' \in SD_i$.

3 Definition of n-G-map pyramids

An n-G-map pyramid is a hierarchical structure, each level of which is an n-G-map. The first level describes the initial data; the other levels describe successive reductions of the previous ones by removing and/or contracting some cells. In this work we focus on applications in image processing, but since n-G-map pyramids can be used in other domains (architectural environments [11], discrete geometry [9] ...), we give here a general definition of n-G-map pyramids.

Definition 4 (n-G-map pyramid) *Let n, $m \geq 0$. An $(m+1)$-level pyramid $\mathcal{P}$ of n-dimensional generalized maps is the set $\mathcal{P} = \{G^k\}_{0 \leq k \leq m}$ where:*

(1) $\forall k$, $0 \leq k \leq m$, $G^k = (D^k, \alpha_0^k, \ldots, \alpha_n^k)$ is an n-G-map;
(2) For each k, $0 \leq k < m$, for each $i \in N$, let R_i^k (resp. C_i^k) be sets of i-cells and $R^k = \bigcup_{i=0}^n R_i^k$ (resp. $C^k = \bigcup_{i=0}^n C_i^k$) with $R_n^k = C_0^k = \emptyset$. The two preconditions $(C1)$ and $(C2)$ of the definition 3 have to be satisfied;
(3) $\forall k$, $0 < k \leq m$, G^k is obtained from G^{k-1} by removing the cells of R^{k-1} and contracting the cells of C^{k-1}.

In the case of an n-G-map pyramid associated with a multi-level segmented image, level 0 is associated with the initial image (or with a first segmentation). New levels are obtained by merging homogeneous regions (n-cells) then by simplifying region boundaries. Merging and simplification are achieved by removing different cells under some conditions (preconditions of the removal and contraction operation, see definition 3). The choice of removed or contracted cells is realized by an external process (an oracle which depends on the application). Examples of 2D and 3D pyramids are provided in Fig. 5 and Fig. 6.

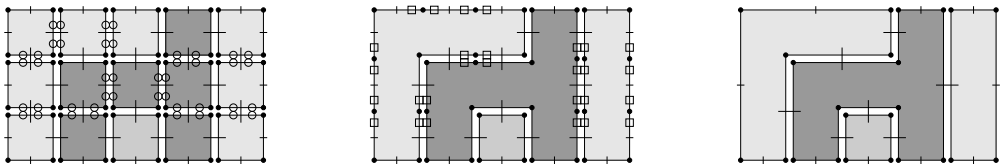


Figure 4. An example of simultaneous removals of cells. (a) A 2-G-map corresponding to an image. A ∘ marks a dart of a 1-cell to be removed in order to merge certain 2-cells. (b) The resulting 2-G-map. A □ marks a dart of a 0-cell to be removed in order to simplify the boundary of regions. (c) The resulting 2-G-map where the region boundaries are simplified.

Two major properties of n-G-map pyramids are:

Proposition 5

(1) Each dart which belongs to a removed or a contracted cell of level k does

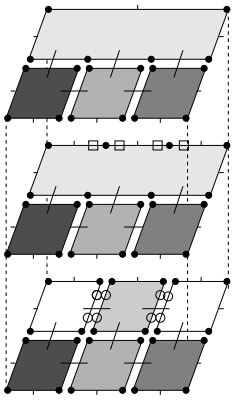


Figure 5. A 2-G-map pyramid composed of three levels. A □ (resp. a ∘) marks a dart of a 0-cells (resp. 1-cells) to be removed.

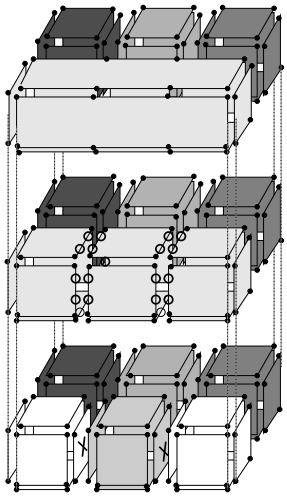


Figure 6. A 3-G-map pyramid composed of three levels. The second level is obtained by removing the 2 faces between the 3 volumes in the foreground (marked by ×). The third level is obtained by removing 6 edges around the volume in the foreground.

not belong to another removed or contracted cell at any level. More formally: $\forall i, j \in N$, $\forall k, l \in [0..m-1]$ *we have:*

$$\begin{cases} R_i^k \cap C_j^l = \emptyset, \\ R_i^k \cap R_j^l = \emptyset, \ \textit{with } i \neq j \ \textit{ or } k \neq l, \\ C_i^k \cap C_j^l = \emptyset, \ \textit{with } i \neq j \ \textit{ or } k \neq l. \end{cases}$$

(2) Let k, $0 \leq k < m$. *A one to one mapping* φ^k *exists between the surviving darts of* G^k *(i.e. the darts which are not removed nor contracted), and the darts of* G^{k+1} *(*$\varphi^k : D^k - (R^k \cup C^k) \longrightarrow D^{k+1}$*).* φ^k *is called the* successor *relation and* $(\varphi^{k-1})^{-1}$ *the* predecessor *relation.*

Remark 6 *In order to simplify the notations, a dart of* G^k *and its image in* G^{k+1} *are denoted by the same name. So,* $D^{k+1} = D^k - (R^k \cup C^k)$.

These properties can be easily deduced from the definition of simultaneous removals and contractions operation [9], and from the definition of n-G-map pyramids. Moreover these properties are useful for the definition of different representations of n-G-map pyramids.

4 Construction of a pyramid for nD images

For image processing, pyramids are used in particular in order to keep in memory different segmentations of a same image. In order to build an n-G-

map pyramid associated with a multi-level segmented image (called in the following *segmentation pyramid*), level 0 is associated with the initial image or segmentation. New levels are built through two steps: first, the cells of the "previous segmentation" corresponding to homogeneous regions (n-cells) are "merged"; second, the resulting n-G-map is simplified. And each segmentation level is obtained by the same process.

In practice, it is not possible to describe the initial image by an n-G-map since it would take too much memory space [5] . We can represent the initial n-G-map implicitly by using a simple matrix. Another idea consists in starting directly with a first segmentation, the corresponding n-G-map being level 0 of the pyramid.

Algorithm 1 achieves the construction of such a pyramid. To build a new level it uses the add_level function which takes three input parameters. The first one is the pyramid in which we add the level. The second one is the dimension of the cells to remove in order to build the new level. And the third one is an oracle that allows to know if a cell has to be removed or not.

Algorithm 1: build_segmentation_pyramid: builds the pyramid corresponding to different segmentation levels of an nD-image.

Input: I: the initial nD-image,

Output: P: the n-G-map pyramid corresponding to different segmentation levels of I.

```
build_level_0 ;
while not terminate do
    add_level(P, n − 1, region_feature) ;
    for i ← n − 2 down to 0 do
        add_level(P, i, local_degree_two_i-cell) ;
```

In the first step, the merging of regions (n-cells) is achieved by the removal of $(n-1)$-cells between regions to merge. It is realized according to an homogeneity criterion.

In the second step, the simplification of the region's boundaries, without lead topological modifications [6] , is realized by merging the different cells which composed a same boundary. For example, in 3D, the different faces that composed a boundary are merged into a unique face. The merge of two i-cells (for $1 \leq i \leq n-1$) is realized by removing the incident $(i-1)$-cell (which local degree is two). This simplification step concerns all the cells of the boundary and

[5] For example, in 3D, describing a grey level image by a G-map is about 750 times bigger than encoding the image itself. Indeed, 2 bytes are necessary to directly encode a voxel, while 1500 bytes are approximatively necessary in the G-map (32 bytes for a dart, and 48 darts for each voxel).

[6] For example, the Euler Characteristic for 2D-surfaces

not only the cells of higher dimension. It is necessary to compute the different removals successively, starting by the highest degree cells. Indeed, the removal of an i-cell decreases the degree of the incident $(i-1)$-cells which could need to be removed. Other simplifications can be made in order to obtain a minimal representation (see [17]).

It is also possible to modify the choice of removed cells in order to control the topology, for instance in order to avoid disconnections. Note that disconnections occur only when we remove a degree one i-cell (i.e. incident twice to the same $(i+1)$-cell). For example, a 2D disconnection comes from the removal of a degree one edge incident twice to the same face, and a 3D disconnection comes from the removal of a degree one face (resp. edge) incident twice to the same volume (resp. face). In order to avoid disconnections, it is enough to modify the oracles such that they return false when the i-cell degree is one.

Algorithm 2 constructs a pyramid level by applying three steps: marking of cells depending on the given oracle, duplication of the last level and removing the marked cells.

Algorithm 2: add_level: builds a new pyramid level.

Input: P: an n-G-map pyramid (which levels are $G^0, \dots G^m$),
i: the dimension of the cells to remove,
oracle: the function that knows if an i-cell need to be removed or not,

Output: P in which a level is added at the top of the pyramid.

mark_i_cells_to_remove(G^m, i, *oracle*, m) ;
duplicate_last_level(P) ;
cells_removal(G^{m+1}) ;

To build a segmentation pyramid, we use two different oracles. The first one indicates if an $(n-1)$-cell must be removed, that is to say if it separates two regions that have to be merged. Different criteria can be used according to the needs of the application. We do not provide more details about such criteria, since our goal is not the definition of a new segmentation algorithm.

The second oracle is used during the simplification step. Algorithm 3 indicates whether the local degree of a given cell is two or not, by checking for each dart the mathematical expression of a local degree two cell (see section 2).

The marking of removed cells is realized by algorithm 4. This algorithm checks each i-cell of the map by using the given oracle, and marks the cell to remove or not. In order to check each cell only once, a temporary mark is used. When an i-cell is met for the first time, all the darts of this cell are marked. Then, testing if an i-cell incident to a dart was already checked is simply achieved by checking this mark. So each dart is checked only once and this algorithm

Algorithm 3: local_degree_two_i-cell: indicates whether the local degree of the i-cell is two or not.

Input: G: an n-G-map,
i: the dimension of the concerned cells,
d: a dart,
l: the pyramid level of G,

Output: true iff the local degree of the i-cell containing d is two.

foreach *dart* d_1 *of* $<>_{N-\{i\}}(d)$ **do**
  if $d_1.\alpha_{i+1}.\alpha_{i+2} \neq d_1.\alpha_{i+2}.\alpha_{i+1}$ **then**
    return false ;
return true ;

has a cost [7] $\mathcal{O}(p)$ where p is the number of darts of G.

Algorithm 4: mark_i_cells_to_remove: marks i-cells that are going to be removed.

Input: G: an n-G-map,
i: the dimension of the cells to remove,
oracle: the function indicating if an i-cell needs to be removed or not,
l: level of G in this map,

Output: G with some i-cells marked.

foreach *dart* d *of* G **do**
  if *not is_in_an_i-cell_yet_examined(d)* **then**
    if *oracle(G, d, i, l)* **then**
      foreach *dart* d' *of the* i*-cell incident to* d **do**
        mark_by_$R_i(d', i)$;
    mark examined the i-cell incident to d ;

The duplication of the last level is realized by algorithm 5. This algorithm duplicates each dart of the last level map, links the duplicated dart to the initial dart by the two relations *successor* and *predecessor*, and then links the duplicated dart with its neighbor for each involution, if it has been already created. This is achieved by using the same technique as for algorithm 4, with a temporary mark. So this algorithm has a cost $\mathcal{O}(pn)$, where p is the number of darts of the last pyramid level and n the dimension of the G-map.

The removals of the marked cells is realized by algorithm 6. The principle of this algorithm is first to link the darts that are not going to disappear for each involution, and second to delete the marked cells. The first step concerns more particularly the neighbor darts of removed cells. In order to know which is the

[7] Note that the complexities of these algorithms depend on the used representation (see section 5). They are expressed here only for the explicit representation.

Algorithm 5: duplicate_last_level: copies the last pyramid level.

Input: P: an n-G-map pyramid (which levels are $G^0, \dots G^m$),

Output: P in which we have duplicated its last level.

```
foreach dart d of G^m do
    d' ← create_copy(d, G^{m+1}) ;
    link_pred_succ(d,d') ;
    for i ← 0 to n do
        if yet_create_copy(d.α_i^m) then
            link_by_α_i(d',copy_of(d.α_i^m))
```

new neighbor of such a dart for an involution, it follows the "path" of marked darts from the dart to its new neighbor by applying the same rules as for the removal operation. We know that these "paths" have no intersection (see [10] and [18]), so darts can be processed in any order without consequences for the final result.

Algorithm 6 examines each dart and when it follows the "path", it considers at most k darts, k being the number of marked darts in G. So this algorithm has a cost $\mathcal{O}(pk)$, where p is the number of darts of G.

Algorithm 6: cells_removal: removes the marked cells.

Input: G: an n-G-map,

Output: G in which marked cells have been removed.

```
foreach dart d of G do
    if is_marked_by_R(d) then
        i ← dim_of_rem_cell(d) ;
        if not is_marked_by_R(d.α_i) then
            d_1 ← d.α_i ;
            neighbor ← d ;
            while is_marked_by_R(neighbor) do
                neighbor ← neighbor.α_{i+1}.α_i ;
            if neighbor ≠ d then
                unlink_by_α_i(d_1) ;
                unlink_by_α_i(neighbor) ;
                link_by_α_i(d,neighbor) ;
foreach dart d of G do
    if is_marked_by_R(d) then then delete_dart(d) ;
```

Since algorithms 4, 5 and 6 have respectively a cost $\mathcal{O}(p)$, $\mathcal{O}(pn)$ and $\mathcal{O}(pk)$, algorithm 2 has a cost $\mathcal{O}(p(n+k))$, where p is the number of darts of the last level, n is the dimension of the space and k the number of darts which have disappeared during the construction of the level.

Note that algorithms presented here are basic ones and several optimizations can be made. We present here algorithms 7 and 8 which realize the optimized construction of a new pyramid level. The idea consists in only duplicating the darts that are not going to be removed. This is achieved by merging the steps of duplication and cell removals (algorithms 5 and 6). In this unique step, a dart is duplicated only if it is not going to disappear. Then it is linked to its neighbor for each involution, if it has been already duplicated, by using exactly the same rules as for the cells removal algorithm.

Algorithm 7: optimized_add_level: Optimized construction of a new level.

Input: P: an n-G-map pyramid (which levels are $G^0, \dots G^m$),
i: the dimension of the cells to remove,
oracle: the function that knows if an i-cell need to be removed or not,

Output: P in which a level is added at the top.

mark_i_cells_to_remove(G^m, i, *oracle*, m) ;
create_new_level(P) ;

Algorithm 8 links each dart of the last pyramid level with its neighbor for each involution. This neighbor can be determined by following the "path" of disappeared darts by applying the same rules as for the removal operation. Two distinct "paths" have no intersection (see [18] for the proof). So, the number of darts processed by following all the "paths" for all the involutions for a given dart is at most k. Note that, in algorithm 8, each "path" linking two darts (d and $d.\alpha_i$) is followed twice. The first time is when the first dart among d and $d.\alpha_i$ is met. At this time, the link is not realized since the copy of the second dart is not yet created. When the second dart is met, the "path" is followed a second time and the link is realized. It is possible to follow each "path" only once by keeping in the second dart which is the copy of the first dart.

So algorithms 7 and 8 have a cost $\mathcal{O}(p(n+k))$, p being the number of darts of the last pyramid level, n the dimension of the space and k the number of darts which have disappeared during the construction. Note that the complexity of algorithm 8 corresponds to the addition of the complexities of algorithms 5 and 6. This is due to the fact that this algorithm merges the duplication and cells removal steps realized by these two algorithms. So this optimization does not change the complexity order, but provides a faster algorithm since the constant of the complexity is divided by two (we explore the set of darts only once), and we do not duplicate removed darts.

Algorithm 8: create_new_level: Optimized creation of a new level.

Input: P: an n-G-map pyramid (which levels are $G^0, \dots G^m$),
i: the dimension of the cells to remove,

Output: P in which a level is added at the top.

```
foreach dart d of G^m do
    if not is_marked_by_R(d) then
        d' ← create_copy(d,G^{m+1}) ;
        link_pred_succ(d, d') ;
        for i ← 0 to n do
            neighbor ← d.α_i ;
            while is_marked_by_R_i(neighbor) do
                neighbor ← neighbor.α_{i+1}.α_i ;
            if yet_create_copy(neighbor) then
                link_by_α_i(d', copy_of(neighbor)) ;
```

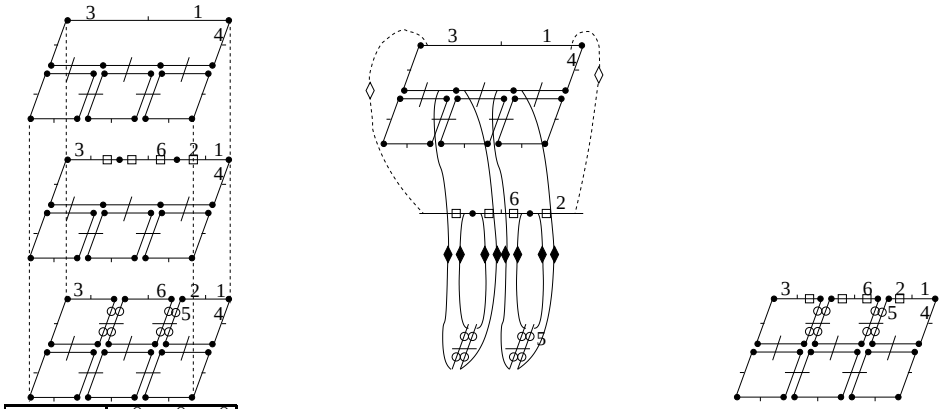
5 Different representations of n-G-map pyramids

An n-G-map pyramid can be described more or less explicitly according to the expected space/time complexity. We present in this section three possible representations that are equivalent: *explicit*, *hierarchical* and *implicit* (see Fig. 7). The first one follows immediately the definition, the two others are proposed in order to reduce the cost in memory space. These two last representations generalize similar existing structures proposed in [19,11]. Note that depending on particular needs, it could be possible to propose other representations.

5.1 Description of the representations

Explicit n-G-map pyramid: All levels and one to one mappings between the levels are explicitly represented (see Fig. 7-a). The representation contains thus $m+1$ n-G-maps, and each dart is linked with its predecessor (except the darts of level 0) and with its successor (except the darts of the last level and the darts which belong to a removed or contracted cell). Moreover, for each level, two marks are associated with each dart which belongs to a removed or contracted cell: the type of the operation (contraction or removal) and the dimension of the cell.

Hierarchical n-G-map pyramid: This representation contains a single set of darts, i.e. the darts of level 0 map (cf. remark 6). All involutions are explicitly represented for all levels. More precisely, for each dart a table gives the images of this dart by all involutions (see Fig. 7-b). Two marks are associated with



level 0	α_0^0	α_1^0	α_2^0
1	2	4	1
2	1	5	2

level 1	α_0^1	α_1^1	α_2^1
1	2	4	1
2	1	6	2

level 2	α_0^2	α_1^2	α_2^2
1	3	4	1

a

	level 0			*level* 1			*level* 2		
	α_0^0	α_1^0	α_2^0	α_0^1	α_1^1	α_2^1	α_0^2	α_1^2	α_2^2
1	2	4	1	2	4	1	3	4	1
2	1	5	2	1	6	2	–	–	–

b

	α_0	α_1	α_2
1	2	4	1
2	1	5	2

c

Figure 7. Three representations of a same pyramid. For each representation, the corresponding array shows images by involutions α for the two darts 1 and 2. In explicit, each dart is duplicated. In hierarchical, darts are not duplicated, but involutions exist at each level. Implicit describes only the darts and the involutions of the first level. (a) *Explicit*. A □ (resp. a ○) marks a dart of a 0-cell (resp. a 1-cell) to be removed. (b) *Hierarchical*. Removed cells are marked in the same way as in the explicit representation. Each dart is drawn in the last pyramid level where it exists. When two darts linked by α_i are drawn in the same level, their link α_i is drawn in the usual way. Otherwise, the links α_0^1 and α_0^0 between two darts of two different levels are represented by lines with ◊ and the link α_1^0 by line with ⧫. (c) *Implicit*. A □ (resp. a ○) marks a dart of a 0-cell of level 0 (resp. a 1-cell of level 1) to be removed.

each dart which belongs to a removed or contracted cell in the same way than for the explicit representation. A possible optimization is the following one: for each dart we only represent distinct images for all involutions and the levels where images change. A structure based upon a similar principle has been proposed in order to model complex architectural environments [11].

Implicit n-G-map pyramid: This representation contains the map of level 0 with additional information which make possible to compute the other levels. For that three marks are associated with each disappeared dart: the type of the operation (removal or contraction), the dimension of the removed or contracted incident cell, and the level at which the cell disappears. A similar representation is proposed by Brun and Kropatsch [19] for 2D combinatorial pyramids.

These three representations are equivalent [20] (in particular, the constraints of the pyramid definition can be easily retrieved for each representation). We can prove it by showing that, for each representation, a dart has the same image for a given involution. Moreover we give in the following section three algorithms allowing to go from a representation to another (explicit $\rightarrow$ hierarchical $\rightarrow$ implicit $\rightarrow$ explicit).

These three representations have different space complexities. Let p be the number of darts of level 0, n be the dimension of the space and m be the number of pyramid levels. Since we have at most pm darts in the *explicit* representation, and since a dart contains the information concerning its neighbors for all n involutions, the space complexity of this representation is $\mathcal{O}(mnp)$. In the *hierarchical* representation, we have p darts and for each dart we have at most mn neighbors. Then, the space complexity of this representation is $\mathcal{O}(mnp)$. But note that if we consider the proposed optimization (for each dart we only represent the distinct images for each involution), most of the time, the average number of different neighbors for each involution and each dart will be smaller than m and depends of the considered pyramid. In the *implicit* representation, we have p darts and for each of them n involutions. So this representation has a cost $\mathcal{O}(np)$.
This implies that the memory space of the explicit representation is bigger than the hierarchical one, which is itself bigger than the implicit one. This is due to the fact that the explicit representation is characterized by an important redundancy of information, since all darts and all involutions are present at each level. The information contained in the hierarchical representation is less redundant since the darts are not duplicated. Last, there is no redundant information in the implicit representation since only one level is explicitly represented.

Most operations (construction, features computation ...) are realized by exploring some orbits at a level of the pyramid (for example, the color of a region is retrieved by exploring the face orbits which have been merged into this region in order to use the pixel colors). An orbit is explored, dart by dart, with a breath first search algorithm which uses all the involutions of the orbit. Retrieve the neighbor of a dart for an involution is in constant time in the explicit and hierarchical representations since the information is explicitly described. But it is not the case in the implicit representation where it is necessary to compute it by following the "path" of disappeared darts. So the complexity of this computing depends on the number of darts of the "path". In consequence, most operations have a complexity proportional to the number of darts of the examined orbits for the explicit and hierarchical representations, while this complexity is multiplied by the average length of the "paths" for the implicit

representation.

We can summarize the main characteristics of the three representations by giving here their main advantage and drawback and their main complexities in Tab. 1:

- *explicit*: it has, at each level, an n-G-map which can be considered independently. It allows to use swapping techniques in order to only keep in memory the necessary information for the current process. But it takes a lot of memory space;
- *hierarchical*: it takes less memory space than the explicit representation while allowing to directly access any level by using involutions of this level. But the operations are not necessarily local since darts are shared by different levels;
- *implicit*: it minimizes the required memory space, and some modifications are easier. For instance, removing or contracting a given cell is simple, since we do not have to propagate modifications. But a level cannot be directly accessed, it is necessary to compute it when it is required.

Complexity \ Representation	Explicit	Hierarchical	Implicit
Memory space	$\mathcal{O}(mnp)$	$\mathcal{O}(mnp)$	$\mathcal{O}(np)$
Exploration of orbits	$\mathcal{O}(k)$	$\mathcal{O}(k)$	$\mathcal{O}(kl)$

Table 1
Recapitulation of the complexities for each representation. p is the number of darts of level 0, n is the dimension of the space, m is the number of pyramid levels, k is the number of darts of the orbits and l is the average length of "paths" (l depends on the pyramid).

6 Conversion algorithms

We propose here three algorithms allowing to construct any representation given another one among the three ones presented before. These algorithms are local ones: the output representation (darts, links, and marks) is built by processing each input dart in two steps: first, one or several copies of the dart are created and second, all necessary information is added. These algorithms use different functions tackled in subsection 6.4.

6.1 Explicit to hierarchical representation

Algorithm 9 builds the hierarchical representation given the explicit one. The idea consists in "merging" each dart with all its successors. The principle of

algorithm 9 is the following: for each dart, create a copy of level 0, then copy the involutions at each level and finally report the different marks on the copy. The creation of the representation is achieved dart by dart and in three steps:

(1) The creation of the corresponding dart in the hierarchical representation;
(2) The linking of this corresponding dart to its neighbor for each involution (α_i with $i \in N$) and each level if it already exists;
(3) If the dart disappears at a level, we mark the dart with the same two marks as those contained in the initial dart in order to indicate how it disappears: removal (R) or contraction (C), and the dimension of the cell ($i \in N$).

Algorithm 9: Construction of the hierarchical representation given the explicit one.

Input: Explicit representation (whose different levels are $G^0, \ldots, G^m$)

Output: Hierarchical representation (whose unique set of darts is D')

```
foreach dart d ∈ G^0 do
    d' ← create_copy(d, D') ;
    lev_d ← last_level_for_dart(d) ;
    for lev ← 0 to lev_d do
        for i ← 0 to n do
            if yet_create_copy(dα_i^lev) then
                link_by_α_i(d',the_copy_of(dα_i^lev), lev) ;
    d^lev_d ← top_most_dart(d) ;
    if is_marked_by_R(d^lev_d) then
        mark_by_R_i(d',dim_of_rem_cell(d^lev_d)) ;
    else if is_marked_by_C(d^lev_d) then
        mark_by_C_i(d',dim_of_con_cell(d^lev_d)) ;
```

This algorithm links each dart of the first pyramid level with its neighbor for each involution at each level. In order to compute the last successor of a dart (the top most dart), we explore at most all the levels. So this algorithm has a cost $\mathcal{O}(pmn)$, p being the number of darts of the first pyramid level, n the dimension of the G-map and m the number of pyramid levels.

6.2 Hierarchical to implicit representation

Algorithm 10 builds the implicit representation given the hierarchical one. The principle of this algorithm is, for each dart:

(1) to create a copy;

(2) to copy the level 0 involutions, i.e. to link the copy of the dart to its neighbor for each involution (α_i with $i \in N$) if it already exists;
(3) then to put on the copy the information contained in the initial dart: the mark if the dart belongs to a removed or a contracted cell, the dimension of the cell, and the level at which the cell disappears (this last mark is not a copy, it is computed).

Algorithm 10: Construction of the implicit representation given the hierarchical one.

Input: Hierarchical representation (whose unique set of darts is D')

Output: Implicit representation (whose unique level is G)

```
foreach dart d ∈ D' do
    d' ← create_copy(d, G) ;
    for i ← 0 to n do
        if yet_create_copy(dα_i^0) then
            link_by_α_i(d', the_copy_of(dα_i^0)) ;
    lev_d ← last_level_for_dart(d) ;
    if is_marked_by_R(d) then
        mark_by_R_i(d',dim_of_rem_cell(d^lev_d),lev_d) ;
    else if is_marked_by_C(d) then
        mark_by_C_i(d',dim_of_con_cell(d^lev_d),lev_d) ;
```

This algorithm links each dart of the implicit representation with its neighbor for each involution. The cost to compute the last level at which a dart exists is $\mathcal{O}(m)$. So this algorithm has a cost $\mathcal{O}(pmn)$.

6.3 Implicit to explicit representation

Algorithm 11 allows to build the explicit representation given the implicit one. The idea is to distribute the information through the different levels. The principle of algorithm 11 is, for each dart:

(1) to construct all darts of the different levels;
(2) to link them by the successor and predecessor relations;
(3) to link the copies with their neighbors for each involution;
(4) and then to put the information contained in the initial dart on the dart in the last level in which it exists.

This algorithm links each dart with its neighbor for each involution at each level. In order to compute the neighbor of a dart for a given involution, we follow the "path" of disappeared darts between a dart and its neighbor by applying the same rules as in the definition of removal and contraction operation

Algorithm 11: Construction of the explicit representation given the implicit one.

Input: Implicit representation (whose unique level is G)

Output: Explicit representation (whose different levels are G^0, ..., G^m)

```
foreach dart d ∈ G do
    lev_d ← last_level_for_dart(d) ;
    for lev ← 0 to lev_d do
        d'^lev ← create_copy(d, G^lev);
        if lev ≠ 0 then
            link_pred_succ(d'^(lev-1), d'^lev) ;
        for i ← 0 to n do
            d_i^lev ← d.α_i^lev ;
            if yet_create_copy(d_i^lev) then
                link_by_α_i(d'^lev,the_copy_of(d_i^lev)) ;
    if is_marked_by_R(d) then
        mark_by_R_i(d'^(lev_d), dim_of_rem_cell(d)) ;
    else if is_marked_by_C(d) then
        mark_by_C_i(d'^(lev_d), dim_of_cont_cell(d)) ;
```

(cf. notion of connecting walk defined iteratively [18]). This path is composed by at most k darts, k being the number of disappeared darts in the pyramid. So this algorithm has a cost $\mathcal{O}(pkmn)$. This is the worst case complexity. In fact, this algorithm has a cost $\mathcal{O}(pxmn)$ (with $1 \leq x \leq k \leq p$), with x being the average length of the "paths". But the average length depends on the construction of the pyramid and so it depends on the application.

So this third algorithm is a little more costly ($\mathcal{O}(pkmn)$) than the two first ones ($\mathcal{O}(pmn)$). Indeed, since the involutions are explicitly represented in the explicit and hierarchical representations, the neighbor of a given dart for a given involution is directly obtained, while the levels are not explicitly represented in the implicit representation and so the neighbor of a dart for a given involution must be computed.

6.4 The different functions used in the algorithms

There are some functions used by the conversion algorithms which are common to the three representations:

- create_copy creates a copy of a dart. It has a cost $\mathcal{O}(1)$;
- yet_create_copy indicates if the copy of a dart has been yet created. It has a cost $\mathcal{O}(1)$;

- the_copy_of gets the copy of a dart. It has a cost $\mathcal{O}(1)$;
- link_by_α_i links two darts by an involution. It has a cost $\mathcal{O}(1)$;
- is_marked_by_R (resp. is_marked_by_C) indicates if a dart is removed (resp. or contracted). It has a cost $\mathcal{O}(1)$;
- mark_by_R_i (resp. mark_by_C_i) marks a disappearing dart by two (in the hierarchical and explicit representations) or three marks (in the implicit representation). It has a cost $\mathcal{O}(1)$;

Other functions are more specific to a particular representation:

- top_most_dart indicates the upper copy of a dart in the explicit representation. This function uses successor links until it reaches a dart with no successor. It has a cost $\mathcal{O}(m)$;
- last_level_for_dart returns the last level of the pyramid at which a dart exists. It has a cost $\mathcal{O}(1)$ in the implicit representation, since the information is contained in the dart. It has a cost $\mathcal{O}(m)$ in the explicit and hierarchical representations, since the information is not contained in the dart: it is necessary to use the successor-predecessor relations until a dart without successor is reached in the explicit representation, and it is necessary to examine each level in order to find the level at which the dart has no neighbor for an involution in the hierarchical representation;
- link_pred_succ links two darts of two consecutive levels by the successor and predecessor relations in the explicit representation. It has a cost $\mathcal{O}(1)$;
- $d.\alpha_i^{lev}$ gives the neighbor of dart d by α_i at level lev. It has a cost $\mathcal{O}(1)$ in the explicit and hierarchical representations. The neighbors of darts are not explicitly represented at each level in the implicit representation. This function follows the "path" of darts, disappeared between level 0 and lev, that separate dart d and its neighbor by using the removal and contraction rules (this is similar to the notion of connecting walk [10]). For a given dart, the complexity of this function depends on the length of the "path". This length is bounded by k, the number of disappeared darts in the pyramid. So it has a cost $\mathcal{O}(k)$.

7 Conclusion and Perspectives

Pyramids of n-dimensional generalized maps are here defined as stacks of reduced n-G-maps where each n-G-map is built from the previous level by contracting or removing cells. n-G-maps unambiguously represent the topology of subdivided n-dimensional objects (for instance nD images). So, n-G-map pyramids can be used in order to process 2D, 3D and 4D images with the same formalism.

n-G-map pyramids have several advantages compared with adjacency graph

pyramids. Mainly, n-G-maps pyramids describe the topological information about n-dimensional multi-level subdivided objects. This is very important since the reduction between levels (achieved by the applications of removal and contraction operations) leads to particular cases (for instance multi-adjacency) which are usually not well handled by graphs. Moreover n-G-map pyramids can be very useful for applications in which it is necessary to check or to control the (evolution of the) topology of an object, for example for tracking objects in video sequences.

The main drawback of n-G-map pyramids is the fact that they can be very expensive in memory space since the space complexity of an nD image is intrinsically high. Different ways can be explored in order to solve this problem:

- optimized representations can be used for specific levels, depending on the applications (for instance for image processing, lower levels correspond to high resolutions; it is clear that representing the initial image by an n-G-map is not efficient);
- swapping techniques can be conceived in order to only keep in memory the information necessary for the current process;
- optimized structures can be deduced for the specific needs of an application: indeed, since topological information is represented by n-G-map pyramids, we can control the loss of topological information acceptable for a given application.

We have shown how to use n-G-map pyramids in order to describe an nD multi-level segmented image. We have described all generic algorithms that allow to build such a pyramid given a segmentation criterion. We have also proposed an optimization which, even if it does not improve the complexity, provides a faster algorithm since the constant of the complexity is divided by two and we do not duplicate removed darts.

n-G-map pyramids can be represented in different ways. We have proposed here three generic representations: *explicit*, *hierarchical* and *implicit* as well as different conversion algorithms. We have presented advantages and drawbacks of these representations. This is particularly important in order to choose an efficient representation according to the needs of an application (complexity in memory space and/or in time).

Now we are conceiving operations for handling this structure. A common problem is the propagation of modifications from a level to other ones. For instance, if we add a new cell within a given level, it is necessary to propagate this modification to the bottom of the pyramid. But if we remove or contract a cell, we have to propagate this modification to the top.

We intend to study the use of n-G-map pyramids for 3D and 4D image processing, leading to the study of more specific operations. Our goal is to develop

a computer software based on n-G-map pyramids which groups many functionalities: multi-level image segmentation, modification of a given region at a particular level by an expert, extraction of topological and geometrical characteristics...

References

[1] A. Montanvert, P. Meer, A. Rosenfeld, Hierarchical image analysis using irregular tessellations, PAMI 13 (4) (1991) 307–316.

[2] J. Jolion, A. Montanvert, The adaptive pyramid : a framework for 2d image analysis, Computer Vision, Graphics and Image Processing 55 (3) (1992) 339–348.

[3] W. Kropatsch, H. Macho, Finding the structure of connected components using dual irregular pyramids, in: Cinquième Colloque DGCI, Université d'Auvergne, France, 1995, pp. 147–158.

[4] W. Kropatsch, Building irregular pyramids by dual-graph contraction, Vision, Image and Signal Processing 142 (6) (1995) 366–374.

[5] L. Brun, W. Kropatsch, Introduction to combinatorial pyramids, in: G. Bertrand, A. Imiya, R. Klette (Eds.), Digital and Image Geometry, Vol. 2243 of LNCS, Springer Verlag, 2001, pp. 108–127.

[6] L. Brun, W. Kropatsch, Combinatorial pyramids, in: Suvisoft (Ed.), IEEE International Conference on Image Processing (ICIP), Vol. II, IEEE, Barcelona, Spain, 2003, pp. 33–37.

[7] L. Brun, W. Kropatsch, Contraction kernels and combinatorial maps, Pattern Recognition Letters 24 (8) (2003) 1051–1057.

[8] P. Lienhardt, N-dimensional generalized combinatorial maps and cellular quasi-manifolds, in: International Journal of Computational Geometry and Applications, Hamburg, Germany, 1994, pp. 275–324.

[9] G. Damiand, M. Dexet-Guiard, P. Lienhardt, E. Andres, Removal and contraction operations to define combinatorial pyramids: application to the design of a spatial modeler, Image and Vision Computing 23 (2) (2005) 259–269.

[10] C. Grasset-Simon, G. Damiand, P. Lienhardt, Receptive fields for generalized map pyramids: the notion of generalized orbit, in: Discrete Geometry for Computer Imagery, no. 3429 in LNCS, Poitiers, France, 2005, pp. 56–67.

[11] D. Fradin, Modélisation et simulation d'éclairage à base topologique: application aux environnements architecturaux complexes, Phd thesis, Université de Poitiers, France (December 2004).

[12] J. Françon, Topologie de Khalimski et Kovalevski et algorithmique graphique, Tech. Rep. 91-10, Centre de Recherche en Informatique, Strasbourg, France (1991).

[13] E. Khalimsky, R. Kopperman, P. Meyer, Boundaries in digital planes, Journal of applied Math. and Stocastic Analysis 3 (1) (1990) 27–55.

[14] V. Kovalevsky, Finite topology as applied to image analysis, Computer Vision, Graphics, and Image Processing 46 (2) (1989) 141–161.

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

[16] E. Brisson, Representing geometric structures in d dimensions: topology and order, Discrete Comput. Geom. 9 (1) (1993) 387–426.

[17] Y. Bertrand, G. Damiand, C. Fiorio, Topological map: minimal encoding of 3d segmented images, in: Workshop on Graph-Based Representations in Pattern Recognition, IAPR-TC15, Ischia, Italy, 2001, pp. 64–73.

[18] C. Grasset-Simon, G. Damiand, P. Lienhardt, Pyramides de cartes généralisées, chemins de connection et orbites généralisées, Tech. Rep. 2, SIC, Université de Poitiers, http://www.sic.sp2mi.univ-poitiers.fr/grasset (2005).

[19] L. Brun, W. Kropatsch, Implicit encoding of combinatorial pyramids, in: O. Drbohlav (Ed.), Proceedings of the Computer Vision Winter Workshop, Valtice, Czech Republic, 2003, pp. 49–54.

[20] C. Grasset-Simon, G. Damiand, P. Lienhardt, nD generalized map pyramids: three equivalent representations, Tech. Rep. 3, SIC, Université de Poitiers, http://www.sic.sp2mi.univ-poitiers.fr/grasset (2005).