Tiled top-down combinatorial pyramids for large images representation

Romain Goffe[1], Luc Brun[2], and Guillaume Damiand[3]

[1] SIC-XLIM, Université de Poitiers, CNRS, UMR6172, F-86962, Futuroscope Chasseneuil, France
goffe@sic.univ-poitiers.fr
[2] GREYC, ENSICAEN, CNRS, UMR6072, 6 Boulevard du Maréchal Juin, F-14050, Caen, France
luc.brun@greyc.ensicaen.fr
[3] LIRIS, Université de Lyon, CNRS, UMR5205, F-69622, Villeurbanne, France
guillaume.damiand@liris.cnrs.fr

Abstract The uprising number of applications that involve very large images with resolutions greater than 30 000×30 000 raises major memory management issues. Firstly, the amount of data usually prevents such images from being processed globally and therefore, designing a global image partition raises several issues. Secondly, a multi-resolution approach is necessary since an analysis only based on the highest resolution may miss global features revealed at lower resolutions. This paper introduces the tiled top-down pyramidal framework which addresses these two main constraints. Our model provides a full representation of multi-resolution images with both geometrical and topological relationships. The advantage of a top-down construction scheme is twofold: the focus of attention only refines regions of interest which results in a reduction of the amount of required memory and in a refinement process that may take into account hierarchical features from previous segmentations. Moreover, the top-down model is combined with a decomposition in tiles to provide an accurate memory bounding while allowing global analysis of large images.

Key words: Irregular pyramid; Topological model; Tiled data structure; Combinatorial map;

1 Introduction

Applicative fields involving high-resolution images raise two main issues for automatic or semi-automatic image analysis. First, images are produced with a very high resolution that usually prevents them from being processed by common models due to memory limitations. Second, the amount of details at full resolution is likely to mask global features which only appear at lower resolutions. For example, scanners for whole slide microscopic imaging produce multi-resolution images with resolutions up to 32 000 × 32 000: low resolutions let appear global features such as tissues delimitations while high resolutions allow to discern the different phases of mitosis within cells. As a result, analyzing such images implies a hierarchical representation with memory constraint.

The segmentation process is the first phase of image analysis: it defines a partition of the image according to a given criterion. Usually, data structures encoding those partitions focus either on geometrical or topological properties of partitions. Geometrical data structures such as arrays of labels provide an efficient encoding of both colorimetric and geometrical features of the partitions but do not provide an efficient access to topological features such as region adjacencies. On the other hand, topological data structures such as RAG represent those topological relationships but do not provide an efficient access to the geometrical properties of a partition. Moreover, such data structures do not allow to discriminate fine topological relationships: for instance, they may not indicate if a region is adjacent or included into another one. Topological maps [BDM03, DBF04] are designed to efficiently access both geometrical and topological properties while allowing modifications of a partition through split and merge operations. Yet, they cannot apply to multi-resolution images since they do not encode a hierarchy of partitions.

Within the hierarchical framework, quadtrees and regular pyramids [BCR90] were the first structures used for segmentation. Both are based on psycho-visual properties but their approach is different: quadtrees use the *top-down* notion of focus of attention and performs a segmentation using a recursive splitting algorithm whereas regular pyramids adopt a *bottom-up* approach where each pixel of a level corresponds to a larger set of pixels at the level below. However, both models induce major drawbacks [BCR90]: they fail to neither encode connected regions of any size and shape at a given level nor provide an efficient access to the neighborhood of a region. Moreover, regular pyramids do not ensure that connected regions defined at a given level remain connected at the level below. Considering these limitations, [Mee89, MMR91] introduced the irregular pyramid framework which overcomes the drawbacks of its regular ancestors using a bottom-up construction scheme. Many segmentation algorithms have then been designed on this framework such as [JM92, Kro95]. Finally, in order to access both geometrical and topological information, [BK03, GSDL06] proposed a model of irregular pyramids composed of combinatorial maps.

However, a bottom-up analysis scheme raises at least two issues when applied to high resolution images: memory usage and relevance of extracted information. Indeed, encoding the whole initial partition of a large image is likely to require a large amount of memory especially if additional levels must be computed. Moreover, extracted information is usually more relevant if the construction scheme allows to use a region to influence the way its children (defined at a higher resolution) are processed. For example, within the histology application field, we may choose to refine differently the cells at full resolution depending on whether their parent region has been identified as stroma or as a cancerous area. The top-down construction scheme proposed by [GBD09] overcomes these two constraints. Nevertheless, the issue of memory constraint is only partially solved since the number of regions defined at a given level is only bounded by the size of the image encoding this level in the pyramid.

The objective of this paper is the definition of a tiled structure for top-down

irregular pyramids. We present in Section 2 the topological models used by our structure. In Section 3, we define the framework of tiled top-down pyramids. Finally, we present some experiments and segmentation results in Section 4 that highlight involved memory requirements and practical applications of the model.

2 Image Representation

2.1 Interpixel Boundaries

Within the segmentation framework, an image is decomposed into a set of *regions* where each region is a connected set of pixels. Representing the geometrical information of a partition consists in encoding the shape of its regions. We use a data structure relying on a matrix of *interpixel elements* ([Kov89, KKM90]) to encode the geometry of the partition elements. Thus, from a geometrical point of view, the partition is represented by an abstract cell complex (AC complex) composed of pointels (or points), linels (or cracks) and pixels, referred to as k-cell, $k \in \{0, 1, 2\}$ (Figure 1(a)). We consider this representation as geometrical since each k-cell has an explicit geometrical embedding. AC complexes are locally finite Alexandroff spaces [Ale37]: every cell has a minimal finite neighborhood. Topological notions within an AC complex such as incidence or adjacency have been defined by [Kov00]:

- the *border* of a k-cell is a set of $(l < k)$-cells defined by a bounding relation;
- two k-cells are *incident* if one belongs to the border of the second;
- two k-cells are *adjacent* if they are both incident to a same $(l < k)$-cell;
- the *degree* of a k-cell is the number of adjacent $(k + 1)$-cells.

Let us introduce definitions of bounding elements that compose regions boundaries (Figure 1(c)):

- a *bounding linel* separates two pixels belonging to different regions;
- a *bounding pointel* is incident to at least two pixels belonging to different regions;
- a *bounding path* is an alternated sequence of bounding pointels and bounding linels;

Finally, regions boundaries are geometrically described as follow (Figure 1(d)):

- a *segment* is a maximal bounding path between two regions since it cannot be prolongated without modifying one of the two incident regions;
- a *node* results from the intersection of at least three segments: a segment connects two nodes.

Since a segment is a sorted sequence of bounding pointels and linels, two orientations are possible: each segment defines two *oriented segments*.

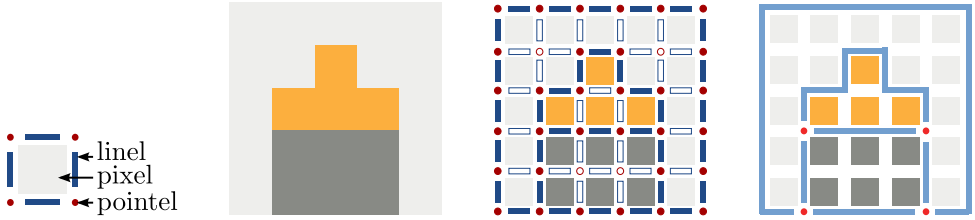


Figure 1: Interpixel boundaries. (a) Notations; (b) Original image; (c) Interpixel elements: bounding pointels and linels are filled; (d) Regions boundaries: segments and nodes.

2.2 Combinatorial Maps

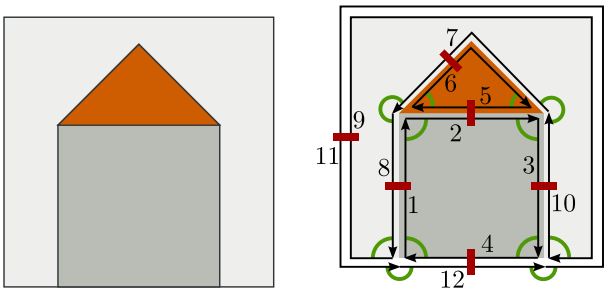
Adjacency and inclusion relationships between regions are usually referred to as the *topological description* of the image since those relationships are not tied to geometrical constraints. Contrary to the interpixel representation (Section 2.1), a two dimensional combinatorial map (2-map) only focuses on adjacency relationships between regions and, compared to a RAG structure, allows to represent multi-adjacencies. A 2-map is based on two operators β_1 and β_2 that apply onto *darts*. A dart is an abstract basic element that corresponds to an oriented segment. It is associated to a single node, segment and region.

Definition 1 (2-dimensional combinatorial map)**.** *A two-dimensional combinatorial map M (or 2-map) is a triplet $M = (\mathcal{D}, \beta_1, \beta_2)$ where:*

1. *$\mathcal{D}$ is a finite set of darts;*
2. *β_1 is a* permutation[1] *on $\mathcal{D}$;*
3. *β_2 is an* involution[2] *on $\mathcal{D}$.*

Intuitively, we can consider a map as a planar graph where β_i operators define relationships between edges. In practice, the β_1 permutation generates an orientation on the border of a face, more precisely, a clockwise cyclic order on the set of edges bounding a face. Each dart belongs to a single face and thus, to a single cycle of β_1. The β_2 involution connects two darts belonging to a same edge, encoding an adjacency relationship. For instance, in Figure 2(b), $\beta_1(1) = 2$, $\beta_2(2) = 5$ and $\beta_2(5) = 2$. For practical reasons, we also introduce the β_0 operator defined as $\beta_0 = \beta_1^{-1}$. The notion of *orbit* is commonly used to traverse edges or faces: given ϕ a set of permutations $\phi = \{f_1, \ldots, f_k\}$, the orbit $< \phi >$ of a dart d represents the set of darts reachable from dart d by applying any combination of f_i and f_i^{-1} permutations.

[1] A *permutation* is a one to one mapping from S onto S.
[2] An *involution* f is a one to one mapping from S onto S such that $f = f^{-1}$.



(a) (b)

dart	β_1	β_2
1	2	8
2	3	5
3	4	10
4	1	12
5	6	2
6	5	7

dart	β_1	β_2
7	8	6
8	9	1
9	10	11
10	7	3
11	12	9
12	11	4

(c)

Figure 2: Combinatorial maps: a topological model for multi-adjacency relationships. (a) Original image; (b) 2-Map: arrows represent darts, β_1 and β_2 operators are respectively represented by arcs and segments; (c) Explicit β_1 and β_2 mappings.

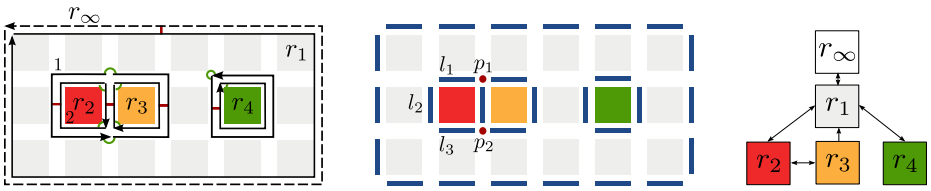
2.3 Topological Maps

A topological map [BDM03, DBF04] combines three distinct models: a 2-map that encodes adjacency relationships, a matrix of *interpixel elements* [Kov89, KKM90] that encodes the regions' boundaries and a tree of regions for inclusion relationships. These three models are illustrated in Figure 3 and described below.

Minimal combinatorial map As illustrated in Figure 3(a), a 2-map encodes topological relationships through β_1 and β_2 operators. The combinatorial map is *minimal* in number of cells: there is not any vertex with a degree lower or equal to 2 and therefore, the removal of any element would result in regions merging and thus, would change the topology. For implementation purposes, darts and regions are linked together: a dart knows the region it belongs to and a region knows a representative dart.

Matrix of interpixel elements All the cells of a 2-map are associated to their corresponding geometrical elements in the interpixel matrix. Associating geometrical information to a topological element is an operation called *embedding*. Given a dart d associated to a region r, a node n and a segment s:

- the orbit $< \beta_1 > (d)$ corresponds to a face associated to region r. This relationship is encoded by a function *region* from $\mathcal{D}$ to the set regions labels such that $region(d) = r$;
- the orbit $< \beta_1 \circ \beta_2 > (d)$ corresponds to a vertex associated to node n. Since each node correponds to a pointel, we encode this correspondence using a function *pointel* from $\mathcal{D}$ to the set of nodes labels such that $pointel(d) = n$;
- the orbit $< \beta_2 > (d)$ corresponds to an edge associated to the unoriented segment s, each dart of $< \beta_2 >$ encoding one orientation along s (for instance,



(a) (b) (c)

interpixel boundaries	geometrical embedding	2-map	orbit
node	pointel	vertex	$< \beta_1 \circ \beta_2 >$
segment	alternated sequence of pointels and linels	edge	$< \beta_2 >$
region	open 4-connected set of pixels	face	$< \beta_1 >$
oriented segment	oriented bounding path	dart	-

(d)

Figure 3: Topological map: three complementary models for partition encoding. (a) Combinatorial map for adjacency relationships. Dashed arrow denotes the dart of the infinite region; (b) Interpixel matrix for geometrical encoding: pointels and linels are represented by bold circles and segments; (c) Tree of regions for inclusion relationships; (d) Connection between the objects defined within the three models composing a topological map.

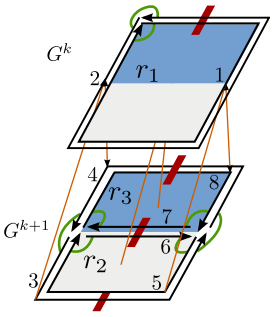
$< \beta_2 > (2) = (2, 5)$ in Figure 2(b)). Thus, we associate each dart to an oriented segment whose first linel is denoted by $linel(d)$.

For instance, in Figure 3(b), the embedding of dart 1 is the oriented segment represented by the sequence (p_1, l_1, l_2, l_3); $linel(1) = l_1$, $pointel(1) = p_1$; $degree(p_1) = degree(p_2) = 3$. Figure 3(d), summarizes associations between interpixel and topological representations.

Tree of regions The tree of regions describes inclusion relationships: a region is the father of the regions it contains. In Figure 3(c), r_1 contains r_2, r_3 and r_4, r_2 and r_3 are adjacent. The root of the tree encodes the background of the image and is called the infinite region (noted r_∞).

2.4 Top-down Pyramids

In order to fit the hierarchical framework, the topological map model has been extended to top-down pyramids [GDB09] and we recall in this section the main notions and operations of the model. Each level of such a pyramid is deduced from the previous one by splitting operations. The resulting sequence of partitions defines thus a *causal structure* [GCM06] which induces hierarchical relationships between levels:



(a)

dart	β_1	β_2	*up*	*down*
1	1	2	-	8
2	2	1	-	4
3	4	5	2	-
4	3	8	2	-
5	6	3	1	-
6	5	7	-	-
7	8	6	-	-
8	7	4	1	-

(b)

region	*up*	*down*
1	-	2
2	1	-
3	1	-

(c)

Figure 4: Representation of a top-down pyramid composed of two levels G^k and G^{k+1}. (a) Pyramid: β_1 and β_2 operators are represented by arcs and segments. Arrows between the levels show *up/down* relationships between darts and regions. Level G^k is composed of a single region r_1. The splitting operation on G^{k+1} allows to differentiate the two regions r_2 and r_3; (b) *up/down* relationships between darts; (c) *up/down* relationships between regions.

each level G^k of a top-down pyramid is encoded by a topological map where each dart and each region is connected to its parent in G^{k-1} and one of its children in G^{k+1}. These links are called *up/down* relationships.

Contrary to bottom-up methods, based on an explicit encoding of the base of the pyramid, a top-down approach starts the segmentation process with a rough partition which is refined at further levels. This construction scheme results in a major memory reduction since many regions may be encoded only at the top level of the pyramid. Moreover, a top-down construction scheme allows to take advantage of the focus of attention over interesting regions: the segmentation of a region can be adapted according to the features of its parent.

During the construction of the pyramid, the refinement of a level is the main operation. In our model, it is performed in three steps. First, the level is duplicated and the *up/down* relationships are set (Figure 5(a)). Second, a splitting criterion selects the regions to refine in the next level. Those regions are decomposed into a set of basic regions, each region enclosing a single pixel (Figure 5(b)). Third, a merging process merges those basic regions according to a merging criterion. Since any couple of adjacent regions may be merged, this refinement step may encode any subdivision of the parent region (Figure 5(c)).

The different levels of a top-down pyramid encode a sequence of nested partitions, each level encoding additional details of the previous partition. This last notion is different from the notions of resolution used within the regular pyramid framework. For instance, a single image with a fixed resolution may be used within a top-down pyramid which will then encode different levels of details of a same image. An alternative solution consists in combining a top-down and a regular bottom-up pyramid. In this case, each level of a top-down pyramid encodes a partition of the

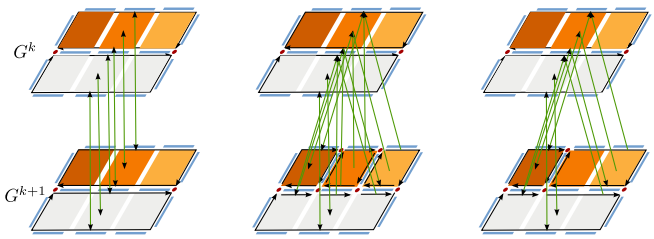


Figure 5: Refinement of the regions that compose a level of a top-down pyramid. (a) Level duplication and *up/down* relationships between darts and regions; (b) Decomposition of selected regions into basic regions enclosing a single pixel; (c) Merge regions according to criterion result.

initial image at a given resolution. The causality of the resulting hierarchy is induced by the construction scheme of a top-down pyramid. As a result, top-down pyramids can be constructed either from single or multi-resolution images.

3 Definition of a Tiled Top-down Pyramidal Model

3.1 Topological Tiles

The main drawback of the top-down pyramidal model is that the global amount of required memory is bounded below by the size of the larger partition processed within the pyramid: for each level, the whole topological map (which encodes the partition defined at this level) must be loaded. In the worst case, each region of the topological map encoding the base level partition may correspond to a single pixel hereby leading to an explicit encoding of the finest partition of the initial image. Even if a top-down scheme usually supplies a major memory reduction compared to a bottom-up scheme, there is no accurate control over memory usage: a solution to this issue is to divide the topological map that represents a level into *topological tiles*. Intuitively, a tile represents a regular and arbitrary subdivision of a level that can be recovered from the juxtaposition of all the tiles it has been split into. The interesting point in dividing the levels of the pyramid in tiles is that most of processings only require a few tiles at the same time: unused tiles can be stored apart by swapping them on disk so that a maximum of memory space is available for the tiles being processed. A simple solution to record a tile is to apply a unique label for each dart and region that identifies them in a file.

Several strategies can be considered for the subdivision in tiles. A first approach would consist in defining a tile as a set of regions. However, such a solution looses the main advantage of memory bounding (a region may contain the whole image). A second approach is to keep a constant number of rectangular tiles per level. This strategy leads to single *up/down* relationships (one tile has exactly one parent and

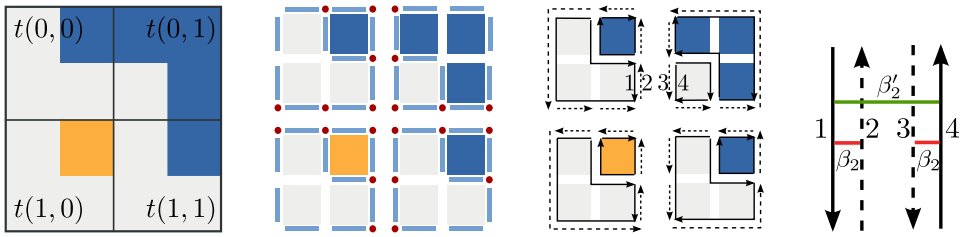


Figure 6: Topological tile. (a) An image divided into 4 tiles; (b) Geometrical representation; (c) Topological representation. β_2' operator is represented by bold segments; (d) Detail of the connection between $t(0,0)$ and $t(0,1)$. Darts 2 and 3 belong to the infinite region. $\beta_2(1) = 2$, $\beta_2(4) = 3$, $\beta_2'(1) = 4$.

one child) but the size of a tile in G^k depends on both the size of a tile in G^0 and the scale factor between the resolutions associated to G^k and G^0: in the case of a hierarchical image with important changes of resolution, the size of a tile on high-resolution images is likely to exceed the available memory. A third approach uses a constant size of tiles: it offers an accurate control of the memory requirements since the amount of memory required by a tile is bounded by the size of the topological map encoding the finest partition on the tile. This last quantity is proportional to the size of the tile which is user-defined. We have thus preferred this solution for our model.

We define a topological tile as a rectangular area encoded by a local topological map composed of a combinatorial map, a geometrical matrix encoding interpixel elements and a tree of regions. Geometrically, two adjacent tiles have the same embedding along their shared border (Figure 6(b)). Topologically, each tile is a closed topological map (Figure 6(c)). We introduce a β_2' operator on the darts that belong to the border of a tile to ensure its connection with adjacent tiles (Figure 6(d)). The procedure *ConnectTileBorders* details the whole operation and is described below.

ConnectTileBorders We call *basic dart* a dart d whose edge embedding is a single linel. Let s (resp. s') be the set of darts that are adjacent to t' (resp. t) and that belong to the infinite region of t (resp. t'). First, we split s and s' into basic darts (line 1 of Algorithm 1). This splitting operation ensures that s and s' share the same number of darts (Figure 7(a)). Second, we link t and t' thanks to the β_2' operator (line 2) by traversing simultaneously the darts of s and s'. This step is illustrated by Figure 7(b). Finally, since the previous steps may have created degree 2 vertices, we perform a simplification pass in order to maintain the minimal property of our model (line 3 and line 4). The idea is to process each vertex that belongs to the shared border (s, s') of two adjacent tiles t and t', and to remove it according to the

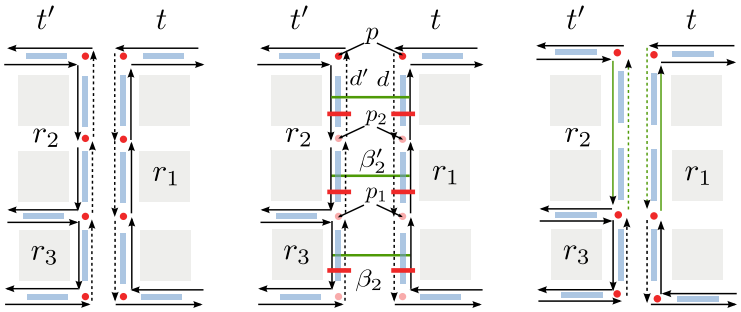


Figure 7: Connection of borders between two tiles. (a) Splitting of the border into basic darts; (b) β_2' connection: d and d' are two basic darts that belong to r_∞ and $p = pointel(d) = pointel(\beta_2(d'))$. Connecting t and t' comes to link $\beta_2(d)$ with $\beta_2(d')$ such as $\beta_2'(\beta_2(d)) = \beta_2(d')$; (c) Simplification step: $degree(p_1)$ equals 2 in t but 3 in t' whereas $degree(p_2)$ equals 2 in both t and t': the vertex removal operation is only performed on pointel p_2.

method in [DL03][3] if its embedding is a pointel whose degree is equal to 2 in both t and t' (Figure 7(c)). Indeed, the notion of minimality of the combinatorial map, presented in Section 2.3 may not be locally preserved along the tiles' borders: for example, in Figure 7(c), the tile t is not a minimal combinatorial map due to pointel p_1 whose degree is equal to 2.

3.2 Tiled Topological Maps

The connection of a set of tiles encodes a partition as a global structure called a tiled topological map. However, the geometrical subdivision entails two main consequences:

- the borders of the tiles should be considered as *fictive* if according to a given merging criterion, pixels on both sides of a tile's border belong to a same region (Figure 8(a));
- edges and regions may be split and shared by several tiles (Figure 8(b)).

Therefore, we detail below the *DetectFictiveBorders* procedure which allows to differentiate fictive borders from real ones. Moreover, we define the permutation δ_1 and the involution δ_2 in a tiled topological map in order to traverse edges and regions in the same way β_1 and β_2 allow to traverse a topological map.

DetectFictiveBorders In order to specify whether or not an edge should be considered as fictive, we use a mark on the linels of the tiles borders. All the linels

[3]The vertex removal operation on a dart d ensures that $\beta_1(\beta_0(d)) \leftarrow \beta_1(d)$ and $\beta_1(\beta_0(\beta_2(\beta_1(d)))) \leftarrow \beta_2(d)$. Then, it deletes d and $\beta_1(\beta_2(d))$.

Algorithm 1: ConnectTileBorders

Data: Two adjacent tiles t and t';
Result: t and t' connected.
Let s the set of darts adjacent to t' and that belong to the infinite region of t;
Let s' the set of darts adjacent to t and that belong to the infinite region of t';
1 Split s and s' into basic darts;
while $\exists$ *dart* $d \in s | d$ *is unmarked* **do**
  $p \leftarrow pointel(d)$;
  Let d' the dart of s' such as $pointel(\beta_2(d')) = p$;
  $p' \leftarrow pointel(\beta_2(d')$;
2 $\beta'_2(\beta_2(d)) \leftarrow \beta_2(d')$;
  if $degree(p) = degree(p') = 2$ **then**
3 vertexRemoval(d);
4 vertexRemoval(d');
  else
    Mark(d);

encoding tiles' geometrical borders are marked as real during the construction of a tiled level (Algorithm 6). Then, the merging criterion is applied on all the couples of regions separated by a tile's border (line 2 of Algorithm 2): if both regions should be merged, we mark the embedding of their shared border as fictive (line 3 and line 4). Note that the merging criterion must implicitly define a partition to ensure that no dangling edges are produced. For example, a quantization is a suitable criterion whereas a criterion based on the difference of average gray levels of the regions compared to a defined threshold is not.

A dart d is said to have a real embedding if its associated oriented segment encodes a non fictive border, the dart is called fictive otherwise. Using the oper-

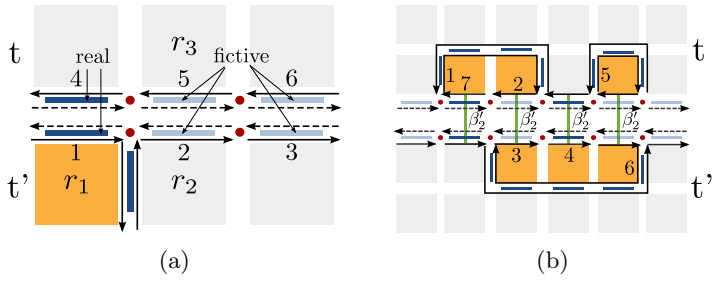


Figure 8: Consequences of the subdivision in tiles. (a) Fictive linels on the tiles borders. Darts 1 and 4 have a real embedding whereas darts 2,3,5 and 6 have a fictive one; (b) Edges and regions may be split and shared between several tiles. Dashed arrows denote the darts of the infinite region.

ation *DetectFictiveBorders*, all the linels of a segment associated to a fictive dart are marked as fictive and conversely, if all linels of a segment are fictive, the corresponding dart is fictive by definition. Therefore, the fictive property of a dart may be efficiently checked by testing the mark of one of its linels such as $linel(d)$ (Section 2.3).

Algorithm 2: DetectFictiveBorders

Data: Two adjacent tiles t and t'.
Result: Fictive borders of t detected.
Let s be the set of darts adjacent to t' and belonging to the infinite region of t;
Let s' be the set of darts adjacent to t and belonging to the infinite region of t';
while $\exists d \in s \mid d$ *is not marked* **do**
  Let d' the dart of s' such as $\beta_2(d') = \beta'_2(\beta_2(d))$;
  $r \leftarrow region(\beta_2(d))$;
  $r' \leftarrow region(\beta_2(d'))$;
1 **if** $merging\_criterion(r, r')$ *is true* **then**
2 Mark $embedding(\beta_2(d))$ as fictive;
3 Mark $embedding(\beta_2(d'))$ as fictive;
  Mark d;

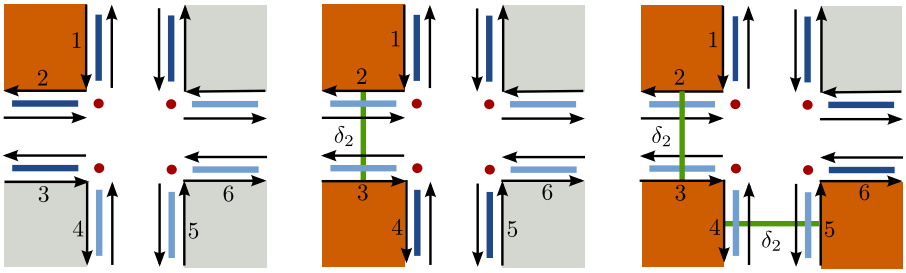
δ_2 involution The δ_2 operator indicates the opposite face independently of whether it belongs to another tile or not. Thus, given a dart d, $\delta_2(d) = \beta'_2(d)$ if $\beta'_2(d)$ is defined (i.e d belongs to a border shared by two adjacent tiles). Otherwise, $\delta_2(d) = \beta_2(d)$. As a result, we introduce the δ_2 operator as follows:

Proposition 1 (δ_2 involution)**.** *Let T be a set of connected topological tiles $T = \{t(i,j)\}_{(i,j)\in\{0,...,W\}\times\{0,...,H\}}$. Let $\mathcal{D}$ be the set of darts of T with a real embedding. δ_2 is an involution on $\mathcal{D}$ such as:*

$$\forall d \in \mathcal{D};\ \delta_2(d) = \begin{cases} \beta'_2(d) & \text{if } \beta'_2(d) \text{ exists} \\ \beta_2(d) & \text{otherwise} \end{cases}$$

Proof. If $\delta_2(d) = \beta'_2(d)$, $\beta'^2_2(d)$ exists and is equal to d by definition of β'_2. Similarly, if $\delta_2(d) = \beta_2(d)$, $\beta_2(d)$ and d have a real embedding and $\delta_2(\beta_2(d)) = \beta^2_2(d) = d$. □

δ_1 permutation The operator δ_1 allows to traverse an edge that may be shared by several tiles. For example, in Figure 8(b), starting from dart 1, iterations of δ_1 operator should lead to the successive traversal of the darts 4, 5, 6 and 7. The idea is the following: given a dart d, let us consider that $\delta_1(d) = \beta_1(d)$ as long as $\beta_1(d)$ does not have a fictive embedding. In this last case, $\delta_1(d)$ belongs to an adjacent tile so we skip fictive darts with the operation $\beta'_2 \circ \beta_1$ until $\delta_1(d)$ has a real embedding. This procedure leads to the following proposition:



(a) $k = 0$ (b) $k = 1$ (c) $k = 2$

Figure 9: The three different cases that define the δ_1 permutation. Each square belongs to a different tile. Dashed arrows denote the darts of the infinite region.

Proposition 2 (δ_1 permutation)**.** *Let T be a set of connected topological tiles $T = \{t(i,j)\}_{(i,j)\in\{0,\dots,W\}\times\{0,\dots,H\}}$. Let $\mathcal{D}$ be the set of darts of T with a real embedding. δ_1 is a permutation on $\mathcal{D}$ such as:*

$$\forall d \in \mathcal{D};\ \delta_1(d) = \beta_1((\beta'_2 \circ \beta_1)^n(d)) \text{ with } n = min\{p \in \mathbb{N} \mid linel(\beta_1((\beta'_2 \circ \beta_1)^p(d)) \text{ is real}\}$$

Proof. Since δ_1 applies from $\mathcal{D}$ onto $\mathcal{D}$, the injection property implies that δ_1 is a permutation on $\mathcal{D}$. Let us consider d and d' such as $\delta_1(d) = \delta_1(d')$. Then, $(\beta'_2 \circ \beta_1)^n(d) = (\beta'_2 \circ \beta_1)^m(d)$. If $n = m$, then $d = d'$ since $\beta'_2 \circ \beta_1$ is a permutation. Otherwise, let us suppose that $n > m$. Then, $(\beta'_2 \circ \beta_1)^{n-m}(d) = d'$ which contradicts the definition of n. Thus, δ_1 is a permutation on $\mathcal{D}$. □

Note that using a 4-connected pixel grid, n cannot exceed 3. The three cases $n = 0, n = 1$, and $n = 2$ are illustrated in Figure 9.

The previous definitions of δ_1 and δ_2 operators allow to extend the definition of combinatorial maps to tiled combinatorial maps as follows:

Definition 2 (Tiled combinatorial map)**.** *Let T be a set of connected topological tiles $T = \{t(i,j)\}_{(i,j)\in\{0,\dots,W\}\times\{0,\dots,H\}}$. Let $\mathcal{D}$ be the set of darts of T with a real embedding. A tiled combinatorial map M is a triplet $M = (\mathcal{D}, \delta_1, \delta_2)$ where:*

(1) δ_1 is a permutation on $\mathcal{D}$ which follows proposition 2;

(2) δ_2 is an involution on $\mathcal{D}$ which follows proposition 1;

3.3 Tiled Top-down Pyramids

The pyramid defines a hierarchy by introducing *up/down* relationships between topological tiles, darts and regions. Except for the tiles of top and bottom levels of the pyramid that do not have respectively any parent and any child, each tile knows its single parent (tile *up*) and one of its child (tile *down*). Several tiles of a same level may have a same parent: since all the children of a given tile are adjacent, we can

efficiently retrieve the children of a tile starting from the tile *down* and finding all its neighbors with the same tile *up*. We define a tiled top-down topological pyramid as a stack of tiled topological maps. The pyramid defines a three coordinates system of representation for a set of topological tiles denoted by $t(i, j, k)$ where (i, j, k) indicates the coordinates (i, j) of the tile t at level k. Moreover, the pyramid may swap or load tiles between memory and disk and spread modifications to ensure the coherence of the model. For example, if a splitting operation modifies the border of a tile, the pyramid may update the adjacent tiles that are either on disk or in memory.

Definition 3 (Tiled top-down topological pyramid)**.** *A tiled top-down pyramid P composed of $l+1$ tiled topological maps is defined by:*

$P = \{t(i, j, k)\}_{k\in\{0,\dots,l\},(i,j)\in\{0,\dots,W_k\}\times\{0,\dots,H_k\}}$ *where $t(i, j, k)$ is a topological tile and $\forall k$, $0 \leq k \leq l$:*

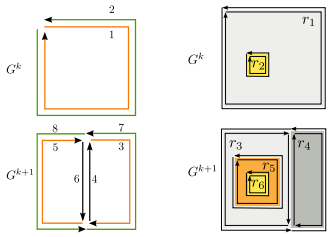
(1) $W_k H_k$ and $(i, j) \in \{0, \dots, W_k\} \times \{0, \dots, H_k\}$ encode respectively the number of tiles and the coordinates of one tile at level k.

(2) $t(i, j, k)$ is a topological tile encoding a partition of the geometrical tile (i, j) defined at level k;

(3) $t(i, j, k + 1), k < l$ is deduced from $t(i, j, k)$ by performing splitting operations.

Note that, in addition to definition 5, a tiled top-down pyramid may alternatively be denoted by $P = \{G^k\}_{k\in\{0,\dots,n\}}$ where G^{k+1} is a tiled combinatorial map (definition 2) deduced from G^k by splitting operations. The hierarchy in a top-down pyramid is implicitly induced by *up/down* relationships. However, those relationships do not grant an immediate solution to retrieve edges and regions from one level to another. Indeed, the *down* relationship only provides a representative element of the connected set that represents a given object (edge or region) in a higher level. The issue consists in retrieving the correct neighbors of this representative and is detailed by two procedures *EdgeChildrenRetrieval* and *RegionChildrenRetrieval* respectively for edges and regions between two levels G^k and G^{k+1}. Both procedures are illustrated by Figure 10.

EdgeChildrenRetrieval Algorithm 3 allows to retrieve an oriented segment defined at level G^k in G^{k+1}. Due to the decomposition of the image partition into a set of tiles, such a segment is encoded in G^k by a sequence of darts d_i such as $d_i = \delta_1(d_{i-1})$. The algorithm retrieves the set of darts they have been split into in G^{k+1}. For each d_i, we retrieve its children in G^{k+1}. For this purpose, $down(d)$ is a first representative of d in G^{k+1} (line 1 of Algorithm 3). While traversing the set of darts incident to $\delta_1(d)$, the dart *up* relationship indicates whether a dart corresponds to d_i (line 2). In the example illustrated by Figure 10.a, edge $e = (1, 2)$ may be retrieved considering the children of either dart 1 or dart 2. Let us consider dart 1. Darts 3 and 5 that correspond to 1 in G^{k+1} are respectively retrieved as $down(1)$ and as the first dart d of the orbit $< \delta_1 \circ \delta_2 >$ $(\delta_1(3) = 4)$ verifying $up(d) = 1$. As a result, $e = (1, 2)$ corresponds to the edges $(3, \delta_2(3) = 7)$ and $(5, \delta_2(5) = 8)$.



(a) (b)

Figure 10: Children retrieval operations for edges and regions in a tiled top-down pyramid. (a) Edge (1,2) corresponds to the edges (3,7) and (5,8); (b) Region r_1 corresponds to r_3, r_4 and r_5.

RegionChildrenRetrieval Retrieving the children of a region r defined in G^k can be decomposed in three main steps. First, $down(r)$ provides a representative of r in G^{k+1} (line 1 of Algorithm 4). Second, neighboring regions are traversed using the orbit $< \delta_1, \delta_2 >$ (line 2). Third, the region up relationship indicates the regions that correspond to r (line 3). Finally, we call *IncludedRegionsRetrieval* (line 4) to add their included regions as long as their up is r (detailed below). In Figure 10.b, the children of r_1 in G^{k+1} are r_3, r_4 and r_5. The procedure retrieves r_3 as $down(r_1)$, r_4 as a neighbor verifying $up(r_4) = r_1$ and r_5 as an included region of r_3. Note that r_6 is ignored although it is an included region of r_5 since $up(r_6) = r_2 \neq r_1$.

IncludedRegionsRetrieval This procedure, described in Algorithm 5, is called from Algorithm 4 on line 4. It uses the tree of regions in the topological map model to retrieve the children of a region R included in a region r such as $up(r) = R$ (line 1 of Algorithm 5). Then, the procedure selects among them the regions whose up is R (line 2).

3.4 Construction Scheme

The construction scheme of a tiled pyramid is described below by the procedure *TiledExtraction*. It relies on the *ConnectTileBorders* operation described in Section 3.1. The operation is an incremental algorithm which only requires four tiles to be loaded into memory. For simplification purposes, given a tile $t = t(i, j, k)$, $left(t)$, $top(t)$, $right(t)$, $bottom(t)$, $up(t)$ respectively denote $t(i-1, j, k)$, $t(i, j-1, k)$, $t(i+1, j, k)$, $t(i, j+1, k)$ and $t(i, j, k-1)$ as long as they are defined.

TiledExtraction The global construction scheme of a tiled top-down pyramid starts by recording all the tiles of the initial map G^0 on disk (line 1 of Algorithm 6). The extraction scheme traverses tiles line by line from the top-left to the bottom-right subdivision. In order to create a tile t (initially a copy of $up(t)$), three other tiles must be loaded into memory: $up(t)$ since t is its refinement, $left(t)$ and $top(t)$

Algorithm 3: EdgeChildrenRetrieval

Data: d: first dart of an edge e in G^k.
Result: s: sequence of darts that represents e in G^{k+1}.
foreach *dart* $d' \in e$ **do**
  $end \leftarrow$ false;
1 $cur \leftarrow down(d')$;
  Add cur on top of s;
  while *end is false* **do**
    $prev \leftarrow cur$;
    $cur \leftarrow \delta_1(cur)$;
2   **while** *end is false and* $up(cur) \neq d'$ **do**
      $cur \leftarrow \delta_1(\delta_2(cur))$;
      if $\delta_2(cur) = prev$ **then**
        $end \leftarrow true$;
    if $cur = down(d')$ **then**
      $end \leftarrow true$;
    if *end is false* **then**
      Add cur at the end of s;
Return s;

Algorithm 4: RegionChildrenRetrieval.

Data: r^k: a region of G^k .
Result: res: list of regions that compose r^k in G^{k+1}.
1 $first \leftarrow down(representative(r^k))$;
2 **foreach** *dart* $d \in < \delta_1, \delta_2 > (first)$ **do**
  $r \leftarrow region(d)$;
3 **if** r *is not marked and* $up(r) = r^k$ **then**
    Add r at the end of res;
    Mark r;
4    $res \leftarrow res \cup IncludedRegionsRetrieval(r, r^k)$;
Return res;

Algorithm 5: IncludedRegionsRetrieval

Data: Let r be a region of a level G^{k+1}.
Let R be a region of a level G^k.
Result: res: regions included in r and whose up is R.
foreach *region* $r' \in r$ **do**
  if r' *is not marked and* $up(r') = R$ **then**
    Add r' at the end of res;
    Mark r';
    $res \leftarrow res \cup IncludedRegionsRetrieval(r', R)$;
Return res;

in order to connect the left and top borders of t (line 2). The right and bottom borders of t are connected at a latter stage when respectively processing $right(t)$ and $bottom(t)$. Note that if $left(t)$ or $top(t)$ are not defined, ($i = 0$ or $j = 0$), t belongs to the border of the image and does not need to be connected with its neighbors. Next is the refinement step: t is initially a copy of $up(t)$ which is refined using a split and merge technique preserving the causality of the pyramid [GBD09] (line 3). The *ConnectTileBorders* operation ensures a coherent topology with *left* and *top* neighbors (line 4). Finally, before processing the next tile t in G^k, we save modified tiles (line 5) and free memory so that the number of tiles loaded into memory remains bounded (line 6).

Algorithm 6: TiledExtraction

Data: An image I.
Result: A tiled top-down pyramid P composed of $m + 1$ levels.
Associate I to P;
foreach *tile* $t \in G^0$ **do**
1 Save t;
for $k = 0$ to $m - 1$ **do**
  foreach *tile* $t(i, j, k) \in G^k$ **do**
    $t \leftarrow t(i, j, k + 1)$;
2   Load $left(t)$ and $top(t)$;
3   Copy and refine t from $up(t)$;
4   Connection with adjacent tiles by calling
    $ConnectTileBorders(t, left(t))$ and $ConnectTileBorders(t, top(t))$;
5   Save t, $left(t)$, $top(t)$ and $up(t)$;
6   Unload $left(t)$, $top(t)$ and $up(t)$;

Table 1: Memory and time comparison between non-tiled and tiled top-down models for different scalings of image *Lena* with a basic segmentation criterion based on average gray values of the regions with user-defined thresholds of 30/20/10 for $G^1/G^2/G^3$.

	top-down model		tiled top-down model		
img side (pixels)	extract runtime	ram (MB)	extract runtime	ram (MB)	disc (MB)
512	4s	92	4s	92	3
1 024	11s	366	8s	95	7
2 048	40s	1 412	28s	92	19
4 096	na	na	2mn	94	69
8 192	na	na	7mn	95	272

4 Experiments and Application

This section has two main objectives: demonstrate the memory feasibility of the extraction of a combinatorial pyramid from large images and present first segmentation results obtained on histological images. We also provide runtime[4] for the construction of tiled top-down pyramids from different images.

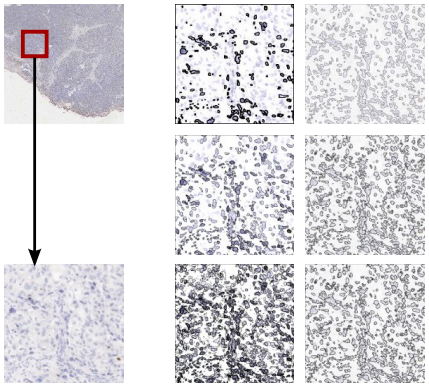
Tiled top-down model Table 1 emphasizes the advantage of using a tiled top-down model: while a plain top-down extraction cannot handle 4 levels of 4 096 × 4 096 due to memory limitations, the tiled approach can process any large image with an almost constant amount of memory. As shown by the fourth column of Table 1, the extraction process is linear with the size of the image as we traverse all pixels to get colorimetric information for the regions. The maximum size of a tile is obtained when each pixel corresponds to a different region. In this case, the size of the combinatorial maps can be estimated as: $4 \times sizeof(dart) \times \sharp pixels + sizeof(region) \times \sharp pixels$. Our implementation results in $sizeof(dart) = 60\,bytes$ and $sizeof(region) = 80\,bytes$. Experiments show similar values since the splitting method decomposes regions and produces one region per pixel before the merging step. Note that the required amount of available memory may be more important since the matrix of interpixel elements and the image[5] must be loaded too. In Table 2, we provide runtime and memory usage for an extraction from different multi-resolution images. We can notice that the subdivision in tiles allows to bound memory requirements of the segmentation step.

[4] The model is implemented in C++ and computations are carried out on an Intel E5300@2GHz with 2GB RAM.

[5] We use the tiles of the *bigtiff* library available at `http://www.aperio.com/bigtiff` to load only the subdivisions that are necessary to the topological tiles being processed.

Table 2: Runtime and memory usage for the extraction of tiled top-down pyramids from the images *Histology* and *Lena* scaled at different resolutions. The pyramids are composed of three levels, each one encoding a different resolution.

image	resolutions (pixels)	tile side	extract runtime	ram (MB)	disc (MB)
Lena	128/256/512	128	1.9s	95	2
Lena	512/1 024/2 048	128	20.6s	101	11
Lena	2 048/4 096/8 192	128	5mn20s	126	157
Lena	8 192/16 384/32 768	128	1h48mn	119	2 491
Hist.	625/2 500/10 000	400	44mn	86	274



(a) (b)

Figure 11: Extraction of a tiled top-down pyramid from a multi-resolution histological image. (a) Partial representation of the original resolution; (b) Partitions.

Segmentation results Figure 11 illustrates the tiled construction of the pyramid associated to the image *Histology* (Figure 11-a) from the last row of Table 2. The two columns in Figure 11(b) represent a same small area of the whole image at the resolutions 625×625, 2 500×2 500 and 10 000×10 000. The first and second columns respectively represent segmentation results obtained respectively with a gradient threshold [GDB09] and a quantization algorithm [KMN+02]. For each of these segmentation algorithms, one may note the progressive refinement of the initial partition which corresponds to additional information provided by higher resolutions.

5 Discussion

In this paper, we have introduced the framework of tiled top-down pyramids as an extension of the model of topological map. Our model handles the hierarchical segmentation of large multi-resolution images. Indeed, it has been designed with the

two following constraints: memory bounding and adaptive segmentation from global features present at low resolutions to finer details at high resolutions.

Our solution for memory bounding results in the decomposition of our model in a set of smaller structures, called topological tiles. Each resolution of the image is encoded by a set of connected tiles which form a tiled topological map. We have defined two new operators on those tiled levels to extend the operations defined for topological maps. These operators allow to abstract the artificial decomposition induced by the geometry of the tiles. Then, the pyramid defines a hierarchy between the topological tiles to encode the hierarchy induced by multi-resolution images. We also provide the main operations induced by this hierarchy.

In order to perform an adaptive segmentation of multi-resolution images, we have proposed a top-down construction scheme: each tile is deduced from its parent after splitting operations. The proposed algorithm is incremental and requires a maximum of four tiles in memory.

Finally, our experiments have confirmed that the memory bounding is sufficient to process large multi-resolution images such as histological images in whole slide imaging.

In our future work, we plan to improve the performance of the model with different splitting techniques and develop the segmentation aspect with new criteria specific to applications in histology.

Acknowledgments

A preliminary version of this article was presented at the 13th International Workshop on Combinatorial Image Analysis [GDB09].

This research is part of the *FoGrImMi* project, supported by the ANR foundation under grant ANR-06-MDCA-008-01/FOGRIMMI.

References

[Ale37] P. Alexandroff. Diskrete räume. *Mat. Sbornik*, 2:501–518, 1937.

[BCR90] M. Bister, Jan Cornelis, and Azriel Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, 11(9):605–617, 1990.

[BDM03] Luc Brun, Jean-Philippe Domenger, and Myriam Mokhtari. Incremental modifications of segmented image defined by discrete maps. *J. Visual Communication and Image Representation*, 14(3):251–290, 2003.

[BK03] Luc Brun and Walter G. Kropatsch. Combinatorial pyramids. In *ICIP (2)*, pages 33–36, 2003.

[DBF04] Guillaume Damiand, Yves Bertrand, and Christophe Fiorio. Topological model for two-dimensional image representation: definition and optimal extraction algorithm. *Computer Vision and Image Understanding*, 93(2):111–154, 2004.

[DL03] Guillaume Damiand and Pascal Lienhardt. Removal and contraction for n-dimensional generalized maps. In *DGCI*, pages 408–419, 2003.

[GBD09] Romain Goffe, Luc Brun, and Guillaume Damiand. A top down construction scheme for irregular pyramids. In *VISSAPP (1)*, pages 163–170, 2009.

[GCM06] Laurent Guigues, Jean Pierre Cocquerez, and Hervé Le Men. Scale-sets image analysis. *International Journal of Computer Vision*, 68(3):289–317, 2006.

[GDB09] Romain Goffe, Guillaume Damiand, and Luc Brun. Extraction of tiled top-down irregular pyramids from large images. In Petra Wiederhold and Reneta P. Barneva, editors, *13th International Workshop on Combinatorial Image Analysis (IWCIA'09)*, Research Publishing Services, pages 123–137. RPS, Singapore, November 2009.

[GSDL06] Carine Grasset-Simon, Guillaume Damiand, and Pascal Lienhardt. *N*-d generalized map pyramids: Definition, representations and basic operations. *Pattern Recognition*, 39(4):527–538, 2006.

[JM92] Jean-Michel Jolion and Annick Montanvert. The adaptative pyramid: A framework for 2d image analysis. *CVGIP*, 55(3):339–348, May 1992.

[KKM90] E. Khalimsky, R. Kopperman, and P.R. Meyer. Boundaries in digital planes. *Journal of Applied Mathematics and Stochastic Analysis*, 3(1):27–55, 1990.

[KMN+02] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. on PAMI*, 24(7):881–892, 2002.

[Kov89] Vladimir Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 46(2):141–161, 1989.

[Kov00] Vladimir Kovalevsky. Algorithms and data structures for computer topology. In *Digital and Image Geometry*, pages 38–58, 2000.

[Kro95] Walter Kropatsch. Building irregular pyramids by dual graph contraction. *IEE Proceedings - Vision, Image, and Signal Processing*, 142:366–374, December 1995.

[Mee89] Peter Meer. Stochastic image pyramids. *Computer Vision, Graphics, and Image Processing*, 45(3):269–294, 1989.

[MMR91] Annick Montanvert, Peter Meer, and Azriel Rosenfeld. Hierarchical image analysis using irregular tessellations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):307–316, 1991.