Hierarchical watersheds within the Combinatorial Pyramid framework

Luc Brun†, Myriam Mokhtari† and Fernand Meyer‡

†GreyC CNRS UMR 6072
Équipe Image - ENSICAEN
6, Boulevard du Maréchal Juin
14050 CAEN Cedex - France
‡Centre de Morphologie Mathématique (CMM)
35, rue Saint Honoré
77305 Fontainebleau Cedex - France

†{luc.brun,myriam.brun}@greyc.ensicaen.fr,‡Fernand.Meyer@cmm.ensmp.fr

Abstract. Watershed is one of the most popular tool defined by mathematical morphology. The algorithms which implement the watershed transform generally produce an over segmentation which includes the right image's boundaries. Based on this last assumption, the segmentation problem turns out to be equivalent to a proper valuation of the saliency of each contour. Using such a measure, hierarchical watershed algorithms use the edge's saliency conjointly with statistical tests to decimate the initial partition. On the other hand, Irregular Pyramids encode a stack of successively reduced partitions. Combinatorial Pyramids consitute the latest model of this family. Within this framework, each partition is encoded by a combinatorial map which encodes all topological relationships between regions such as multiple boundaries and inclusion relationships. Moreover, the combinatorial pyramid framework provides a direct access to the embedding of the image's boundaries. We present in this paper a hierarchical watershed algorithm based on combinatorial pyramids. Our method overcomes the problems connected to the presence of noise both within the basins and along the watershed contours.

1 Introduction

Segmentation and contour extraction are important tasks in image analysis. Among the multitude of methods, the watershed transformation [17, 12, 14, 8, 4] arises as a popular image segmentation algorithm. This method usually based on the gradient of the image provides a partition of the image into a set of basins corresponding to local minima of the gradient and a set of watershed pixels. These pixels may be roughly understood as the borders of the basins. Using a flooding process [17] watershed pixels are defined as the places where water coming from several basins merges. Watershed algorithms presents the main advantage of providing closed curves leading to a proper definition of regions.

A well known drawback of the watershed algorithms is the over segmentation often produced by these methods (e.g. [17]). Since the contours appear to be correct, the over segmentation problem turns out to be equivalent to a proper valuation of the saliency of each contour. This contour's saliency is generally used conjointly with an homogeneity criteria in order to derive a hierarchy of partitions.

This hierarchy of partitions may be encoded using Irregular Pyramids [13, 10, 3]. These data structures encode each partition as a graph whose nodes and edges respectively correspond to regions and region's adjacencies. Usual Irregular Pyramids [13] are made of a stack of simple graphs (i.e. graphs without multiple edges nor self-loops). Within this framework several contours between two regions are encoded by a single edge which thus simply encodes the existence of at least one contour between the two regions. However, within the hierarchical watershed framework the contours of the partition play a major role in the decimation process. The explicit encoding of each contour of the partition by one edge requires thus to encode an irregular pyramid made of non simple graphs. Such enriched graphs may be created using the Dual graph reduction scheme [10]. Within this framework, the reduction operation is performed in two steps: First, the contraction of a set of edges identifies a set of vertices. This operation may create redundant edges such as empty self-loops or double edges [10]. These redundant edges are characterized in the dual of the graph and removed by a set of edge removals. Applied to the watershed transform such a reduction scheme provides a graph where each vertex encodes a basin and each edge corresponds to one contour between two basins.

Combinatorial Pyramids inherit all the useful properties from the dual graph pyramids with several additional advantages: Firstly, within the combinatorial pyramid framework the dual graph may be implicitly encoded and thus updated. This property allows to decrease both the memory and computational time requirements. Secondly, combinatorial pyramids preserve the local orientation of edges around vertices and faces. This last property is used to retrieve efficiently the set of points encoding a contour.

The aim of this paper is to present one implementation of a hierarchical watershed algorithm within the combinatorial pyramid framework.The paper is thus organized as follows: We first present the main features of combinatorial pyramids (Section 2). Then, the specific advantages of this model within this framework are illustrated by a new hierarchical watershed construction scheme using specific features of combinatorial pyramids (Section 3).

2 Combinatorial Pyramids

A combinatorial pyramid corresponds to a stack of successively reduced combinatorial maps where the initial combinatorial map G_0 usually encodes a 4 connected planar sampling grid. A combinatorial map $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$ may be understood as an encoding of a planar graph. The construction of a combinatorial map from a plane graph is as follows: first edges are split into a set of half-edges called darts, the set of darts being denoted by $\mathcal{D}$. Two darts sharing the same edge are connected by the involution α which maps each of the two darts to the other one. The vertices of the graph are encoded by the permutation σ whose cycles correspond to the sequence of darts encountered when turning counter-clockwise around each vertex. Each vertex of the graph is thus encoded by one cycle of the permutation σ. In the same way each edge of the graph is encoded by one cycle of α. In what follows, the cycles of σ and α containing a dart d will be respectively denoted by $\alpha^*(d)$ and $\sigma^*(d)$. An introduction to combinatorial maps and Combinatorial Pyramids may be found in [2, 3].

As in the dual graph pyramid scheme [10] (Section 1) the two operations used to reduce combinatorial maps within the pyramid are the contraction and the removal. In order to preserve the number of connected components of the initial combinatorial map, we forbid the removal of bridges and the contraction of self-loops. Such contractions may be avoided by using a contraction kernel defined as a forest of the initial combinatorial map. As mentioned in the introduction of this paper the contraction operation may create redundant edges such as empty self loops and double edges. A contraction kernel is thus followed by a removal kernel removing the eventual empty-self loops and double edges. A reduction step in the pyramid involves thus the application of 2 kernels : One contraction kernel and one removal kernel. Note that, while a contraction kernel is application dependent, the removal kernel is automatically defined from one combinatorial map. Indeed, within our reduction scheme a contraction kernel specifies a set of regions to be merged while the removal kernel is restrained to the removal of redundant edges.

Given an initial combinatorial map G_0 encoding the 4 connected planar sampling grid and a sequence of contraction or removal kernels $K_1, \ldots, K_n$ each reduced combinatorial map $G_i = (\mathcal{D}_i, \sigma_i, \alpha_i)$ may be build from $G_{i-1} = (\mathcal{D}_{i-1}, \sigma_{i-1}, \alpha_{i-1})$ and the kernel K_i [3]. Note that we have $\mathcal{D}_i = \mathcal{D}_{i-1} - K_i$. The set of darts of any reduced combinatorial map is thus included in the initial set of darts $\mathcal{D}_0$. The resulting pyramid is usually stored explicitly as a sequence of successively reduced combinatorial maps $(G_1, \ldots, G_n)$. However, we have shown [2, 3] that within the combinatorial pyramid framework all the kernels and all the reduced combinatorial maps may be encoded efficiently by storing for each initial dart in G_0, the maximal level where this dart survives in the pyramid and the operation applied at each level. This implicit encoding may be performed by :

1. one function *state* from $\{1, \ldots, n\}$ to the 2 states $\{Contracted,\ Removed\}$ which specifies the type of each kernel.
2. one function *level* defined for all darts in $\mathcal{D}_0$ such that $level(d)$ is equal to the maximal level where d survives:

$$\forall d \in \mathcal{D}_0 \; level(d) = Max\{i \in \{1, \ldots, n+1\} \,|d \in \mathcal{D}_{i-1}\}$$

a dart d surviving up to the top level has thus a level equal to $n+1$.

Given the function level, each kernel K_i may be efficiently retrieved as the set of darts whose level equals to i. Moreover, any reduced combinatorial map may be retrieved from this implicit encoding in a time proportional to the total length of the boundaries encoded by this combinatorial map [2, 3].

The explicit encoding of the pyramid $(G_0, \ldots, G_n)$ may thus be replaced by $(G_0, level, sate)$. Moreover, if the initial combinatorial map G_0 encodes a planar sampling grid, the permutations σ_0 and α_0 may be implicitly encoded using any convention on the numbering of darts. The pyramid may thus be simply encoded by $(\boldsymbol{\mathcal{D}}_0, level, state)$. On the other hand, the current top level combinatorial map is frequently accessed during the construction of the pyramid. We thus decided to store additionally a combinatorial map encoding the top level of the pyramid. This combinatorial is updated at each level during the construction of the pyramid. Our encoding of the pyramid is thus defined by $(G_n, \boldsymbol{\mathcal{D}}_0, level, state)$ where G_n denotes the current top level combinatorial map. This choice allows an efficient construction scheme of the pyramid while avoiding the explicit encoding of all the intermediate combinatorial maps.

Moreover, we have shown [2] that using the two functions *level* and *state* we can associate to each edge $\alpha_i^*(d)$ an ordered sequence of 1-cells [18] (also denoted cracks or linels) which encodes the embedding of the edge, i.e. the boundary between the two regions associated to the vertices $\sigma_i^*(d)$ and $\sigma_i^*(\alpha_i(d))$. The sequence of linels of one contour is retrieved in a time proportional to its length [2].

The construction of a combinatorial pyramid is performed by successive simplifications of the top level combinatorial map. From this point of view, combinatorial pyramids may be compared to other topological data structures [6, 9]. However, combinatorial pyramids differ from these alternative encodings on two points : Firstly, the implicit encoding provided by the function *level* allows us to encode the whole sequence of reduced combinatorial maps rather than the top level one. Secondly, alternative data structures [6, 9] encode the geometry of the partition thanks to an additional geometrical model cooperating with the topological one in order to provide a full description of the partition. Using combinatorial pyramids, the geometrical embedding of the partition is provided without additional memory requirements by the function *level*.

3 Hierarchical watersheds with Combinatorial Pyramids

Within the combinatorial pyramid framework, the initial combinatorial map is usually associated to the 4 connected sampling grid. Given a $m \times n$ grey level image, we build an initial combinatorial map G_0 encoding the $m \times n$ sampling grid and we store within each vertex the grey value (or altitude) of the associated pixel. This vertex's altitude is the basic feature used to compute the watershed transform on G_0.

3.1 Building the initial watershed partition

Several methods [17, 5, 8] have been proposed to build the basins of a graph. The topological watershed method designed by Bertrand and Couprie [5] produces a grey level image W whose minima encode the basins. The construction of a contraction kernel from such an image may be performed by computing a spanning tree [10] which covers each basin. The union of all trees forms the contraction kernel. Using Meyer's [12] or Vincent [17] algorithms the basins are built iteratively using a flooding process. The main property satisfied by these algorithms are :

1. the assignment of a vertex to a label (watershed or basin) is performed only once,
2. each vertex marked as belonging to a basin is adjacent to at least one vertex already aggregated to this basin.

Starting from an empty kernel, condition 2 insures that for each vertex adjacent to a basin we can find one edge connecting it to this basin. We can thus add this edge to the contraction kernel. Moreover, the contraction kernel may contain a loop only if one vertex is aggregated twice to a same basin which is refused by condition 1. The contraction kernel can thus, in this case, be built in parallel with the watershed transform.

Using any of the above methods we can thus build a contraction kernel K_1 whose trees span each basin of G_0. The contraction of K_1 contracts each of these trees into a single vertex. Since each vertex of G_0 contains the altitude of the associated pixel, we can compute during the contraction process the minimal altitude of each tree and store the resulting value within the contracted vertex. Each vertex of G_1 associated to a basin stores thus the minimal altitude of this basin.

The kernel K_1 is followed by a removal kernel K_2 in order to remove redundant edges(Section 2). Let us denote by G_2 the combinatorial map obtained from the successive applications of K_1 and K_2. Since the kernel K_2 does not imply any merge of vertices, the vertice's values computed during the contraction step remain unchanged. Moreover, since the trees of K_1 span only the basin of G_0 the vertices of G_2 correspond either to basins or to watershed pixels.

3.2 Building a partition into basins

Hierarchical watershed algorithms are generally based on a partition of the image into a set of basins. However, watershed algorithms produce a partition of the image into a set of basins *and* a set of watershed pixels each of these pixels being encoded by one vertex in G_2. The explicit encoding of watershed vertices induces two types of problems within this framework: First of all if two basins are separated by a thin watershed line the adjacency between the two basins is not encoded by a single edge but by a sequence of two edges encoding for each watershed vertex its adjacency to the two basins. Secondly, watershed vertices may form thick connected components [17, 14] where many watershed vertices are incident to 0 or 1 basin. In such a case, the adjacency between the basins surrounding such a component and thus the existence and location of the contours between the basins is relative to a labeling of the watershed vertices to the different basins.

Two recent algorithms [11, 9] have been proposed to encode an image partition defined by pixel's boundaries. These two approaches encode a sequence of pixels defining a boundary between two basins by a single edge. However, each approach suffers of different drawbacks. The method presented by Marchadier [11] must pre-process the boundary pixels in order to avoid some configurations. This last step modifies the partition without taking into account the image's content. The method present by Köethe [9] may violate some basic topological properties by contracting basins into single points. Finally, using either of these methods boundary pixels do not belong to any basin. Some well known properties of an image partition into 4 or 8 connected regions may thus be violated. For example, the method presented by Köethe encodes 4 connected basins but may produce partitions with more the 4 basins incident to a same point.

To overcome these drawbacks we designed [4] an algorithm which aggregates the watershed vertices to the basins using a flooding process. This algorithm ensures that each watershed vertex aggregated to a basin may be connected to the minimum of this basin by an always descending path. Moreover, this algorithm satisfies the same conditions than Meyer's and Vincent's algorithms (conditions 1 and 2 Section 3.1). We can thus build a contraction kernel K_3 during the aggregation process. As previously the contraction kernel is followed by a removal kernel K_4. The final combinatorial map is denoted by G_4.

The above method is similar to the minima extension presented by Bertrand [1]. However, both methods differ on the following point: Roughly speaking, the greedy algorithm presented by Bertrand preserves the minimal altitude one as to climb to connect two adjacent basins. This method allows to attach a global pass value value to each couple of adjacent basins. Our method [4] preserves the minimal altitude one has to climb to connect two adjacent basins *while* passing by one watershed pixel. The aim of this method is to attach one pass value to each elementary element of the border between two adjacent basins (see below).

3.3 From watershed values to linel's pass values

The combinatorial map G_4 encodes a partition of the image into a set of basins. Each edge between two vertices of G_4 encodes a contour between two basins and may be associated to a sequence of linels encoding the embedding of the associated boundary (Section 2). Each linel along the contour separates two pixels belonging to each basin. Moreover, since each basin is initially surrounded by watershed pixels, at least one of these two pixels was initially marked as a watershed. Let us consider a linel l between two basins B_1 and B_2 of G_4 separating two pixels P and Q belonging respectively to B_1 and B_2. If P and Q were both initially marked as watershed pixels, there is by construction [4](Section 3.2) two descending paths from P to the minimum of B_1 and from Q to the minimum of B_2. If one of the two pixels, say P, was not initially marked as a watershed we can induce from the construction scheme of the basins [14] that P is connected to the minimum of B_1 by an always descending path. The maximum of the altitudes $h(P)$ and $h(Q)$ represents thus the minimal altitude one has to reach to connect the minima of B_1 and B_2 while passing by P and Q. This value is associated to each linel and called a linel's pass value. These valuated linels correspond intuitively to the values of the watershed pixels along the contours. However the aggregation of the watershed vertices to the basins and the transfer of the watershed pixels altitudes to the linel's pass values allows us to overcome the two drawbacks mentioned in Section 3.2.

3.4 From linel's pass values to edge's pass values

Given an edge $\alpha_4^*(d)$ of G_4 let us consider the function $Pv(t)$ which encodes the sequence of linel's pass values encountered along the contour associated to $\alpha_4^*(d)$. The symbol t may be understood as the rank of the linel along the contour while $Pv(t)$ represents the pass value of the associated linel. The value usually determined from the function Pv within the hierarchical watershed framework [14] is its minimum. Such a value may be associated to each edge of the combinatorial map G_4. However, the minimal linel's pass value along a contour is sensitive to the noise which may be present along it. Moreover, this choice does not take into account the distribution of Pv and thus the saliency of the minimum.

In order to overcome this last drawbacks we propose to measure the saliency of the different minima of the function Pv using the following decomposition: If the function Pv contains less then a given number (fixed to 5 in our experiments) of samples we consider that no reliable values on the saliency of the minima may be defined and we fix the edge's pass value to the minimum of the function Pv. Otherwise, we use the volumic filters defined by Vachier [16] to compute the saliency of the different minima as follows:

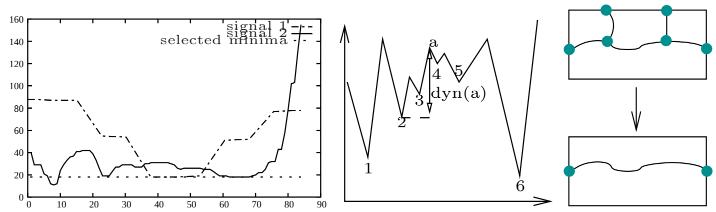
Given an edge $\alpha_i^*(d)$ of the current top level combinatorial map G_i, we consider the function Pv associated to $\alpha_i^*(d)$ as a 1D relief which is progressively flooded. When two 1D basins b_1 and b_2 merge along a maxima m the volume of b_1 and b_2 are computed by:

$$\forall j \in \{1, 2\} \quad vol(b_j) = \sum_{t \in b_j} m - Pv(t) \tag{1}$$

The two basins b_1 and b_2 are then filled up to the altitude m and the process continues on the updated signal. This process stops when the signal has only one minimum left. Note that our method is based on a family of leveling functions. Indeed, the signal used at step i of our algorithm is defined as $Pv_i(t) = \psi^i(Pv(t))$ where ψ^i is the i^{th} iteration of the leveling operator [16] ψ which merges the basins separated by the lowest maxima and fills them up to the altitude of this maxima.

Given the set $\{b_1, \ldots, b_n\}$ of 1D basins merged by our method we define the global pass value of the contour as the minimal altitude of the basins with the greatest volume:

$$pass\_value(\alpha_i^*(d)) = \underset{j \in \{1,\ldots,n\}}{Min} \{Depth(b_j) \,|\, Vol(b_j) = \underset{k \in \{1,\ldots,n\}}{max} Vol(b_k)\} \tag{2}$$



(a) edge's pass value

(b) edge's dynamic

(c) merging

Fig. 1. (a) Two different signals with a same edge's pass value. (b) Computation of the dynamic of the edge a on a 1D example. (c) Enlargement of a contour.

where $Depth(b_i)$ and $Vol(b_i)$ denote respectively the minimal altitude of b_i and its volume (equation 1).

Intuitively, this choice corresponds to a measure of the saliency of each minimum by the volume of the associated basin and a selection of the minimum of greatest volume. Note that, in practical applications, the basin of maximal volume is generally uniquely defined and the Min operator in equation 2 becomes useless.

Fig. 1(a) shows two signals with a same pass value. The signal 1 which has only one minimum is valuated by the value of this minimum. On the other hand, the small gaps at the beginning of signal 2 are not selected since the last minimum has a higher altitude but a maximal volume.

3.5 From edge's pass values to edge's dynamics

The computation of the edge's pass values allows us to reduce the influence of noise along the contour by affecting to each contour its more significant minimum. However, the contrast between two basins is relative both to the pass values of their common contours and to the minima of the two basins. In order to reduce the influence of noise inside the basins which may induce the presence of many non significant basins, we use the contour's dynamic introduced by Najman [15]. Intuitively, the dynamic of a contour is defined by a flooding process which progressively merges all basins. The dynamic of each edge is then defined as the maximal difference between the edge's pass value and the depth of the two basins which merge along the contour. An illustration of the computation of the edge's dynamics on a 1D signal is provided in Fig. 1(b). Our algorithm floods thus progressively the current combinatorial map by merging at each step the two basins separated by the edge with the lowest pass value. The dynamic of the edge is then computed and we store within the basin with the higher altitude a pointer to the remaining basin. For example, in Fig. 1, before the flooding of edge a, the basins 3 and 4 points respectively towards the basins 2 and 5. After the merge of edge a, basin 5 points to the basin 2. These pointers allows us to retrieve for each basin the deeper basin to which it has been merged in order to compute the edge dynamic. This set of pointers defines a forest within the set of basins, the root of each tree being retrieved in almost constant time using union-find operations [7].

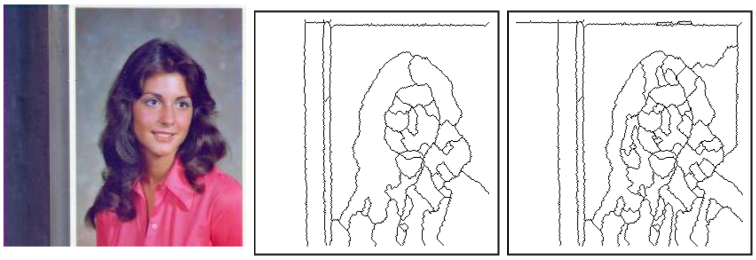
Note that the computation of the dynamics is based on the edge's pass value (Section 3.4) rather than the minimum of each contour. This difference influence both the computation of the dynamic at each step and the flooding process which is based on the edge values. The edge's dynamics computed by our algorithm are thus different from the ones computed using the contour's minima.

3.6 From edge's dynamic to hierarchical segmentation

Within the hierarchical watershed framework, the edge's dynamics are usually computed once and combined with an other homogeneity criteria to merge progressively the different basins. This approach suffers from two main drawbacks: First of all, as mentioned in Section 1 the edge's dynamics are often used to reduce the over segmentation of the image produced by watershed algorithms. Due to the over segmentation, many contours of the partition are initially composed of a small number of linels (e.g. 4 or 5). The reliability of a global value from a such reduced sample of data is difficult to state (Section 3.4). Secondly, the edge's dynamics are not updated according to the updates of the partition and may thus contain unreliable values all along the reduction process. However, after each sequence of merge operations, the removal of redundant edges (Section 2) either removes a contour or enlarges it by a concatenation with other contours (Fig. 1(c)). Therefore, the length of a contour in the pyramid is an increasing function of the level and the problems connected with the presence of very short contours tends to disappear as we go up in the hierarchy. In order to overcome the drawbacks connected with the poor reliability of the edge's dynamics at the first levels of the pyramid our method update the edge's pass values and edge's dynamics after each sequence of contraction and removal operations. More precisely, our method iterates the following steps:

1. Initialization step: Compute the edge's pass value and goto step 3,
2. Update the pass value of edges adjacent to a merged region,
3. Compute edge's dynamics,
4. Build a contraction kernel containing the edges with the lowest dynamic ; apply the contraction kernel and remove redundant edges. If more than one region left goto step 2.

Step 2 corresponds to a lazy programming. Indeed, since the computation of an edge's pass value requires only features of the associated contour, we can ensure that an edge not adjacent to a region merged at the previous step keeps its pass value. Step 3 performs the operations described in Section 3.5. Note that, after the first iteration some vertices do not encode a single basin but a set of merged basins. In this case, the minimal altitude of the vertex is defined



(a) original image (b) minimum (c) max. volume

Fig. 2. Two segmentations using different edge's pass values

as the minimal altitude of the merged basins. Step 4 builds a contraction kernel from the set of edges with a low pass value. Note that this set of edges may defines loops in the current combinatorial map. In this case one of the edge of each loop is not added to the kernel in order to respect the forest requirement of a contraction kernel(Section 2). However, this case is rare in practical cases and the contraction kernel generally include all the edges with the lowest dynamic.

Fig. 2(b) and (c) shows two levels of two pyramids built by valuating edges respectively with the minimal value of the contour and the edge pass value as defined in Section 3.4. The levels in each pyramid have been selected such as the white bar on the left of Fig. 2(a) forms only one region at the level above. Much more meaningful details are preserved in Fig. 2(c) which thus better fit to the intuitive notion of contour's saliency. This phenomena is due to the edge's pass value which do not take into account minima with a small volume within the profile of the contours. Note that the operations used to obtain Fig. 2(b) may be performed without our hierarchical data structure (using e.g. [15]) while Fig. 2(c) is obtained using both the geometrical and topological features of Combinatorial Pyramids.

4 Conclusion

We have presented in this paper a new hierarchical watershed method based on the edge's dynamic. The different partitions of the hierarchy are encoded within the combinatorial pyramid framework. The main advantages of combinatorial pyramids within this framework are the encoding of each contour by one edge and the efficient retrieval of each contour's embedding as a sequence of linels. We used these properties to define a new edge's pass value which allows us to overcome the noise which may be present within the contours. The presence of noise within the basins is corrected using edge's dynamics based on the edge's pass values previously computed. In future studies we are planing to combine the edge's dynamic with statistical tests on the content of the regions. More studies should also be undertaken on the valuation of the minimal value of a contour.

References

[1] G. Bertrand. Some properties of topological greyscale watersheds. In *procs. SPIE Vision Geometry XII*, volume 5300, pages 182–191, 2004.

[2] L. Brun. *Traitement d'images couleur et pyramides combinatoires*. Habilitation à diriger des recherches, Université de Reims, 2002.

[3] L. Brun and W. Kropatsch. Combinatorial pyramids. In Suvisoft, editor, *IEEE International conference on Image Processing (ICIP)*, volume II, pages 33–37, Barcelona, September 2003. IEEE.

[4] L. Brun, P. Vautrot, and F. Meyer. Hierarchical watersheds with inter-pixel boundaries. In *Image Analysis and Recognition: International Conference ICIAR 2004, Part I*, pages 840–847, Proto (Portugal), 2004. Springer Verlag Heidelberg (LNCS).

[5] M. Couprie and G. Bertrand. Topological grayscale watershed transformation. In *SPIE Vision Geometry VI Proceedings*, volume 3168, pages 136–146, 1997.

[6] G. Damiand. *Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d*. PhD thesis, Université des Sciences et Techniques du Languedoc, Décembre 2001.

[7] C. Fiorio and J. Gustedt. Two linear time Union-Find strategies for image processing. *Theoretical Computer Science*, 154(2):165–181, 5 Feb. 1996.

[8] R. Glantz and W. Kropatsch. Plane embedding of dually contracted graphs. In *Discrete Geometry for Computer Imager DGCI'2000*, Lecture Notes in Computer Science. Springer, Berlin Heidelberg, New York, 2000. In Press.

[9] U. Köthe. Deriving topological representations from edge images. In *Geometry, Morphology, and Computational Imaging, 11th Intl. Workshop on Theoretical Foundations of Computer Vision, LNCS, Springer Verlag*, volume 2616, pages 320–334, 2003.

[10] W. G. Kropatsch and H. Macho. Finding the structure of connected components using dual irregular pyramids. In *Cinquième Colloque DGCI*, pages 147–158. LLAIC1, Université d'Auvergne, ISBN 2-87663-040-0, September 1995.

[11] J. Marchadier, D. Arquès, and S. Michelin. Thinning grayscale well-composed images. *Pattern Recognition Letters*, 25:581–590, 2004.

[12] F. Meyer. Topographic distance and watershed lines. *Signal Processing*, (38):113–125, 1994.

[13] A. Montanvert, P. Meer, and A. Rosenfeld. Hierarchical image analysis using irregular tessellations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):307–316, APRIL 1991.

[14] L. Najman and M. Couprie. Watershed algorithms and contrast preservation. In *Discrete geometry for computer imagery*, volume 2886, pages 62–71. LNCS, Springer Verlag, 2003.

[15] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEETPAMI*, 18(2):1163–1173, December 1996.

[16] C. Vachier and F. Meyer. A morphological scale-space approach to image segmentation based on connected operators. In *Workshop on Mathematics and Image applications*, Paris, September 2000.

[17] L. Vincent and P. Soille. Watersheds in digital spaces : an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[18] J. Webster. Cell complexes, oriented matroids and digital geometry. *Theoretical Computer Science*, 305(1–3):491–502, Aug. 2003.