



Comparison of Local and Global Region Merging in the Topological Map* **

Alexandre Dupas¹ and Guillaume Damiand²

¹ Université de Poitiers, SIC, Bâtiment SP2MI, F-30179 Futuroscope, France
dupas@sic.univ-poitiers.fr

² LaBRI, Université de Bordeaux 1, UMR 5800, F-33405 Talence, France
damiand@labri.fr

Abstract. The topological map is a model that represents 2D and 3D images subdivision. It aims to allow the use of topological and geometrical features of the subdivision in image processing operations. When handling regions in an image, one of the main operation is the region merging, for example in segmentation process. This paper presents two algorithms of region merging in 3D topological maps: one local which modifies locally the map around merged regions, and another one global which runs through all the elements of the map. We study their complexities and present experimental results to compare both approaches.

Key words: Combinatorial maps, Intervoxel boundaries, Region merging, Image segmentation

1 Introduction

Region based segmentation consists in partitioning an image into connected sets of pixels or voxels called regions. One of the approaches, derived from the *split-and-merge* [9] methods and called *bottom-up*, begins with an over-partition of the image, and decreases the number of regions using successive region merging operations. These algorithms require a model that describes images, and operations onto these models like the region merging.

Many works have studied models representing partitions of an image. Topological data structures describe images as a set of elements and their adjacency relations. The most famous example is the Region Adjacency Graph (RAG) [13] which represents each region by a vertex, and where neighboring regions are connected by an edge. But the RAG suffers from several drawbacks as it does not represent multiple adjacency or makes no differences between inclusion and adjacency relations. To solve these issues, the RAG model has been extended, for instance in dual-graph structure to represent 2D images [11] or in topological maps [1, 2, 5, 8] used to represent 2D and 3D images.

* Partially supported by the ANR program ANR-06-MDCA-008-05/FOGRIMMI.

** Paper published in Proceedings of 12th IWCIA 2008, LNCS 4958, pp. 420-431, 2008. Thanks to Springer-Verlag Berlin Heidelberg. The original publication is available at <http://www.springerlink.com/content/u32162261m518711/>

The aim of our work is to provide image processing operations using 3D topological maps. We need algorithms that maintain and update the information stored, like relations between regions (adjacency, inclusion, etc.). In a previous work [7], a region merging method has been defined on 3D topological maps, but it is limited to the merging of two adjacent regions.

In this paper, we present and compare two algorithms, one *local* and one *global*, for region merging. The first one is dedicated to interactive processing of 3D topological maps as it aims to minimize the number of modifications of the map while merging together a set of connected regions. The global algorithm is designed for automated processing for example in segmentation operations. It allows the merging of any number of regions by connected components. Some experiments show that the two proposed algorithms behave as intended. The *local region merging* is slower when used intensively contrary to the *global region merging*.

In Sect. 2, we first present topological maps, which are combinatorial maps verifying specific properties used to represent 3D images. Then, Sect. 3 studies the local approach of region merging in topological maps. We explain the algorithm and give its complexity. In Sect. 4, we present the global approach. Section 5 shows the experimentation results, and compares both approaches in different cases. Lastly, we conclude and give some perspectives in Sect. 6.

2 Recalls on 3D Topological Maps

A 3D topological map is an extension of a combinatorial map used to represent a 3D image partition. Let us recall the notions on combinatorial maps, 3D images, intervoxel elements and topological maps that are used in this work.

A combinatorial map is a mathematical model describing the subdivision of a space, based on planar maps. A combinatorial map encodes all the cells of the subdivision and all the incidence and adjacency relations between the different cells, and so describe the topology of this space.

The single basic elements used in the definition of combinatorial maps are called *darts*, and adjacency relations are defined onto darts. We call β_i the relation between two darts that describes an adjacency between two i -dimensional cells (see Fig. 1 B for one example of combinatorial map and [12] for more details on maps and comparison with other combinatorial models). Intuitively, with this model, the notion of cells is represented by a set of darts linked by specific β_i relations. For example, a face incident to a dart d is represented by the set of darts accessible using any combination of β_1 and β_3 relations. Moreover, given a dart d , which belongs to an i -cell c , we can find the i -cell adjacent to c along the $(i - 1)$ -cell which contains d by using $\beta_i(d)$. For example, given a dart d that belongs to a face f and a volume v , the volume adjacent to v along f is the 3-cell containing $\beta_3(d)$.

Let us now present some usual notions about image and intervoxels elements. A voxel is a point of discrete space \mathbb{Z}^3 associated with a value which could be a color or a gray level. A three dimensional image is a finite set of voxels. In

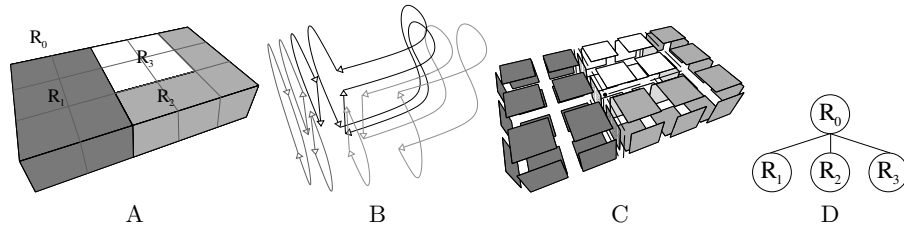


Fig. 1. The different parts of the topological map used to represent an image. (A) 3D image. (B) Minimal combinatorial map. (C) Intervoxel matrix (embedding). (D) Inclusion tree of regions.

this work, combinatorial maps are used to represent voxel sets having the same labeled value and that are 6-connected. The label of a voxel is given by a labeled function $l : \mathbb{Z}^3 \rightarrow L$ that gives for each voxel its label (a value in the finite set L). We speak about region for a maximal set of 6-connected voxel having the same label.

To avoid particular process for the image border voxels, we consider an infinite region R_0 that surrounds the image. If a region R_j is completely surrounded by a region R_i we say that R_j is *included* in R_i .

In the intervoxel framework [10], an image is considered as a subdivision of a 3-dimensional space in a set of cells: voxels are the 3-cells, surfels the 2-cells between two 3-cells, linels the 1-cells between two 2-cells and pointels the 0-cells between two 1-cells (see example in Fig. 1 C where 2-cells, 1-cells and 0-cells are drawn).

The *topological map* is a data structure used to represent the subdivision of an image into regions. It is composed of three parts:

- a minimal combinatorial map representing the topology of the image;
- an intervoxel matrix used to retrieve geometrical information associated to the combinatorial map. The intervoxel matrix is called the *embedding* of the combinatorial map;
- an inclusion tree of regions.

Figure 1 presents an example of topological map. The 3D image, composed of three regions plus the infinite region R_0 (Fig. 1 A), is represented by the topological map which is divided in three parts labeled B, C and D. The minimal combinatorial map extracted from this image is shown in Fig. 1 B. The embedding of the map is represented in Fig. 1 C, and the inclusion tree of regions in Fig. 1 D.

The combinatorial map allows the representation of all the incidence and adjacency relations between cells of an object. In the topological map framework, we use the combinatorial map as a topological representation of the partition of an image in regions. Each face of the topological map is separating two adjacent regions and two adjacent faces do not separate the same two regions. With these rules, we ensure the minimality (in number of cells) of the topological map (see [7, 4] for more details on topological maps).

The intervoxel matrix is the embedding of the combinatorial map. Each cell of the map is associated with intervoxel elements representing geometrical information of the cell. A face, in the combinatorial map, is embedded by a set of surfels separating voxels of the two incident regions. The edges, which are the border of faces, are represented by a set of linels. The vertices, which are the border of edges, are embedded by pointels. Thus the intervoxel matrix allows to retrieve the geometry of the labeled image represented by the combinatorial map.

The inclusion tree of regions represents the inclusion relations. Each region in the topological map is associated to a node in the inclusion tree. The nodes are linked together by the inclusion relation previously defined. To link the tree with the combinatorial map, each dart d of the map knows its belonging region (called $region(d)$). Each region R knows one of its dart called *representative dart* (called $rep(R)$). $rep(R)$ has to belong to the external surface of R and its other incident region R_2 , given by $region(\beta_3(rep(R)))$, has to be a smaller region than R considering the sweeping order of the image voxels (i.e. R_2 is found before R when we run through the image with a scan line algorithm).

3 Local Region Merging

The objective of the local merging approach is to modify the topological map in a local way to reflect the merging of the selected regions. As its a local operation, we do not want to run through all the darts of the topological map. We also aim to locally transform the inclusion tree of regions. Lastly, it is necessary to respect the minimality property of the topological map and update the geometrical embedding of the map.

3.1 Algorithm

The region merging algorithm takes in input a topological map M and a set S of connected regions to merge. The algorithm modifies the topological map M such as all the regions of S are merged together in the *resulting region*.

Algorithm 1 presents the local approach of the region merging operation. An overview of the local process divides it into three main steps:

- initialize the main region and mark the internal faces (i.e. faces incident to two regions that will be merged);
- update the inclusion tree to take into account the possible modifications of the inclusion relations;
- update the combinatorial map and the embedding by removing internal faces and simplifying their incident edges and vertices.

Before starting the merging process itself, we choose in set S a region, called *main region* (line 1 of Algo. 1). Instead of creating a new region, we use the main region as the resulting region of the merging algorithm. Thus, we choose the main

Algorithm 1: Local approach of the region merging operation**Data:** Topological map M ; Connected set of regions S **Result:** Merge together all the regions of S in M .

- 1 Choose the main region among the region of S ;
- 2 **foreach** dart d belonging to regions of S in the map **do**
- 3 **if** $region(\beta_3(d)) \in S$ **then**
 - └ Mark all the darts belonging to the face incident to d ;
- 4 Update the inclusion tree;
- 5 Remove all internal faces (previously marked);
- 6 Simplify the cells incident to the removed faces;

region so its representative dart respects the definition of the representative dart (see Sect. 2) for the resulting region.

The next step is the marking of the internal faces between the regions of set S . We run through the darts belonging to regions of S in the topological map (line 2). Each dart d such as $region(d) \in S$ and $region(\beta_3(d)) \in S$ is incident to an internal face. We mark all the darts incident to the face incident to such a dart. In order to check if a dart belongs to a region of set S in a constant time, we use a second mark applied on regions that belong to S .

To update the inclusion tree (line 4) we remove the selected regions of the tree and then we run through all the darts of the selected regions to find newly internal surfaces. For each one, we build a new connected component of regions in the inclusion tree.

The next step concerns the two other structures of the topological map: the combinatorial map and its embedding, the intervoxel matrix. We want to remove internal faces from the combinatorial map (line 5). This is achieved by using the face-removal (defined in [6] like any other cell-removal operations used in this work) on each internal face previously marked.

The last step of this operation is the map simplification (line 6). Indeed, the removal of the internal faces modifies the degree of the incident edges. Degree two edges are not minimal according to the constraints of the topological map: they are removed with the edge-removal operation. If a degree-two edge is removed, the degree of the two incident vertices changes, so they are simplified if needed.

When a cell is removed in the combinatorial map by using cell-removal operation, the intervoxel matrix is also modified in order to remove the corresponding embedding. For instance, when a face (2-cell) is removed in the combinatorial map, all the corresponding surfels are removed in the embedding.

3.2 Complexity

Now we are going to study the time complexity of the local region merging algorithm presented in Algo. 1. The selection of the main region is achieved in $O(|S|)$ where $|S|$ is the number of regions to merge. Checking the validity of the representative dart is a constant time operation.

The loop marking internal faces process each dart belonging to the regions of S exactly once. Testing if $region(\beta_3(d)) \in S$ is a constant time operation, so this step has time complexity $O(D_{selected})$ where $D_{selected}$ is the number of darts belonging to the regions of S .

Updating the inclusion tree is the most expensive operation regarding the number of covered darts. To find new internal surfaces, we run through all the darts belonging to selected regions. This step has complexity $O(D_{selected})$. If no inclusion appears, $D_{included}$, the number of darts belonging to newly included regions, is equal to zero. Otherwise all the included darts are covered. Thus this operation has time complexity $O(D_{selected} + D_{included})$ and in the worst case, when all regions are newly included in the resulting region, we cover the darts of the whole topological map.

The cell-removal operation in the combinatorial map has linear time complexity in number of darts incident to the corresponding cell. We only remove cells belonging to the surface of the merged regions. Thus, the complexity is linear in number of darts of the merged region: $D_{selected}$. Updating the embedding is linear in the number of intervoxel elements representing the cells. The whole process may be upper-bounded by $S_{removed}$, the number of surfels removed during the face-removal operation (because this number is always greater than the number of linels and the number of pointels). This stage has time complexity $O(D_{selected} + S_{removed})$.

Algorithm 1 has time complexity $O(D_{selected} + D_{included} + S_{removed})$. It can be upper bounded by the number of darts and the number of surfels of the whole topological map.

4 Global Region Merging

As a first approach of the region merging in the topological map, the local merge does not provide an effective way to deal with multiple sets of connected regions to merge. To overcome this drawback, we look for another approach allowing several merges at the same time.

The principle of the global merging algorithm is to separate the modifications of the topological map from the merging of regions. The aim is at first to handle the regions at an high level, and then to translate the high level merging into an effective one by removing cells from the topological map.

The first part of the algorithm concerns the symbolic merging of regions. To handle the high level merging and the representation of region sets, we use a disjoint-set forest [3] of regions. Disjoint-sets are used to represent multiple sets. The main operations are the retrieval of the belonging set of an element (**find**), and the merging of two sets (**union**). We use union-find trees to represent disjoint-sets. In [14], R. Tarjan shows that the union and find operations can be considered as constant time operations in practical cases.

During the symbolic merging, all regions in a same connected component are merged together in the same union-find tree. Then the merge operation, defined on disjoint-set forest, is used and some internal features of regions like number

Algorithm 2: Global approach of the region merging operation

Data: Topological map M ; **Oracle** function**Result:** Merge all the regions by connected components according to **Oracle** in M .

```

1 foreach dart  $d$  of  $M$  do
2   if Oracle(region( $d$ ),region( $\beta_3(d)$ )) then
3      $\perp$  Merge the union-find trees of region( $d$ ) and region( $\beta_3(d)$ );
3 Remove all internal faces;
4 Simplify the cells incident to the removed faces;
5 Rebuild the inclusion tree of regions;

```

of voxels, or mean color are propagated. The root of each tree in the disjoint-set forest will be the resulting region for the merge of the underlying connected set of regions. When all the sets have been processed, each root of the disjoint-set forest is a remaining region of the merging process.

The second part of the algorithm removes all the internal faces. Then, we simplify the topological map and rebuild the inclusion tree.

4.1 Algorithm

The global approach of the region merging algorithm takes in input a topological map M and an oracle function that tells how regions have to be merged. The algorithm modifies the topological map M such that all the connected regions that have to be merged according to the oracle are actually merged.

Algorithm 2 presents the global approach of the region merging operation, which is divided into three main steps:

- compute the disjoint-set forest of regions: this is the symbolic merging;
- remove the internal faces and simplify the incident cells;
- build the new inclusion tree.

The first step of the algorithm (line 1 of Algo. 2) concerns the symbolic merging. If the oracle merges r and its neighboring region r_2 in a same connected component, we merge the union-find trees containing the two regions. At the end of the symbolic merging, since only neighboring regions have been merged, each union-find tree represents a connected component of regions. The oracle function could be for example a labelling function which merges regions having the same label.

The next step (line 3) concerns the removal of the internal faces in the topological map. This is the same process as the one used in the internal face removal for the local approach, but without using a mark on internal faces. Indeed, a dart d belongs to an internal face if $find(region(d)) = find(region(\beta_3(d)))$ (i.e. both adjacent regions are in the same disjoint-set). We run through all the darts of the topological map, and for each dart d validating the previous assertion, we use the face-removal operation.

Then, we use the map simplification, presented in the local approach, to obtain the minimal combinatorial map and its corresponding embedding (line 4).

The last step of this approach concerns the building of the new inclusion tree of regions (line 5). This operation is processed with the same algorithm as the one used during the extraction of the topological map (see [4] for more details). Contrary to the local approach, the inclusion tree is completely destroyed and then rebuilt ; we do not modify the previous tree. This is justified because the global approach is generally used to merge many regions, and in such a case it is more expensive to update the tree than to rebuild it (see experiments in Sect. 5).

4.2 Complexity

The first step of the global region merging (presented in Algo. 2) is the symbolic merging. The test line 2 of Algo. 2, which has constant time complexity using a mark on regions, is performed on all the darts of the topological map. The merging of the disjoint-sets containing the two implied regions is considered as a constant time operation. This step has time complexity $O(|D|)$ where $|D|$ is the number of darts of the topological map.

The complexity of the internal face removal is $O(|D|)$ where $|D|$ is the total number of darts in the map. The map simplification algorithm and the updating of the embedding are the same as the ones used in the local approach. So this whole step, covering the face removals and the simplification of the map, has complexity $O(|D| + S_{removed})$ where $S_{removed}$ is the number of removed surfels in the embedding. The complexity of the inclusion tree building is in $O(|D|)$ which give us the time complexity of Algo. 2 in $O(|D| + S_{removed})$. We can notice that the complexity of the two approaches is the same in the worst case, but they have different processing times depending on the number of merged regions.

5 Experiments and Analysis

In this section, we are interested in comparing the two approaches of the region merging in topological maps. We study the processing time of the two methods. All the following experiments use as input a same topological map representing 32^3 regions in a 32^3 voxels image (which means each voxel belongs to a different region). In this topological map, we select several regions to merge, in order to study some specific configurations. The goal of the experiments is to compare the behaviour of both algorithms and thus showing the more interesting approach in different cases. For this reason, we use small artificial images to facilitate the comparison without depending on the content of the image.

5.1 Merging of a Connected Component of Regions

We have compared the two approaches when the selected regions form only one connected component. We have introduced two protocols of experimentation in

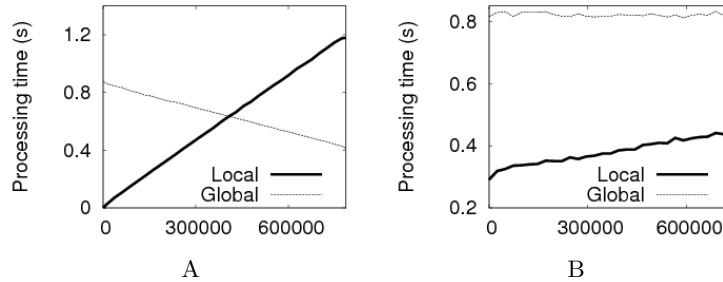


Fig. 2. Processing time comparison of the two region merging approaches (processing time given in seconds). (A) By the number of darts belonging to merged regions. (B) By the number of darts belonging to newly included region.

this case. For the first one, we have merged an increasing number of regions using both approaches. Figure 2 A shows the total processing time of both methods in function of the number of darts belonging to merged regions. This shows that the processing time of the local approach increases linearly with the number of darts. The processing time of the global algorithm tends to decrease as the number of merged regions increases. Indeed, during the merging operation, most of the darts are removed, and the simplification and the building of the inclusion tree are cheaper when the number of darts is smaller. To conclude this experiment, we can observe that the local approach is faster than the global one when the number of merged regions remains small. On the contrary, the global approach is faster when the number of regions is bigger.

Table 1 presents the processing times of the three different steps of both algorithms in this experiment. The chosen steps are the same as the ones used in the overview of both algorithms. We can observe, in the local approach results, that each step takes an increasing time as the number of merged regions increases. The global approach behaves differently. The symbolic merging grows slowly, but the face-removal, the simplification, and the building of the inclusion tree take less time as the number of merged regions increases. These values show the differences between the two approaches and explain the behavior of their processing times.

The second experiment, presented in Fig. 2 B, compares the processing time of the two methods when the number of included regions increases. We have merged a constant number of regions (5768), but these regions are merged in order to include a specified number of regions. This example shows the merging of a small number of regions regarding to the topological map size. This explains why the global approach is slower than the local one in this experiment. The main point we can observe is that the processing time of the global approach remains constant whereas in the local approach where it slowly increases. This is due to the updating of the inclusion tree of regions which depends on the number of included darts. The global approach does not depend on the configuration of these included regions so its processing time is more reliable in general cases.

Table 1. Processing time of different steps of both approaches in function of the number of darts belonging to merged regions.

Merged darts	24576	196608	393216	589824	786432
Local approach					
Initialization	0.008	0.052	0.100	0.148	0.196
Face-removal & simplification	0.028	0.156	0.208	0.316	0.388
Inclusion tree updating	0.012	0.120	0.304	0.480	0.616
Total	0.048	0.328	0.612	0.944	1.200
Global approach					
Initialization	0.060	0.064	0.064	0.072	0.072
Face-removal & simplification	0.520	0.476	0.424	0.380	0.336
Inclusion tree building	0.288	0.228	0.152	0.084	0.008
Total	0.868	0.768	0.640	0.536	0.416

5.2 Merging of Several Connected Components of Regions

We have also compared the two approaches when the selected regions form several connected components. We have studied the impact of the number of sets on the merging process, as well as the impact of their size. We have used the same kind of topological map representing 32^3 regions. In this image, we have selected several regions that belongs to different connected components. To merge such sets of connected regions with the local approach, we have used the local algorithm on each set, and we have computed the total processing time. The same result is obtained by giving to the global method the union of all the sets.

Firstly, we have merged an increasing number of sets of connected regions. Each set contains two regions. Figure 3 A presents the processing time of the two approaches in function of the number of merged sets. It shows that the local approach processing time increases with the number of merged set whereas the global method one decreases. The local approach behaves as expected since it is the addition of the processing time of a single merge in the topological map. The global approach behavior is explained, as previously, by the fact that the number of remaining darts decreases and thus the map simplification cost decreases. Figure 3 A shows that the global approach becomes more efficient than the local one as the number of sets growth.

Secondly, we are looking into the influence of the set size on both approaches. In the experiment, we have always merged 256 sets of regions, but we increase the size of each set from 2 to 64 regions. As in the previous experiment, the global merging processing time decreases as the number of merged regions increases. The processing time of the local approach is more expensive when the size of the sets increases. Figure 3 B shows that the global merging operation becomes more efficient than the local one when the size of the connected components increases.

In all the experiments, we have a same conclusion. The local approach is better when the number of selected regions is small and when the number of connected components remains low. This is mainly the case when the region

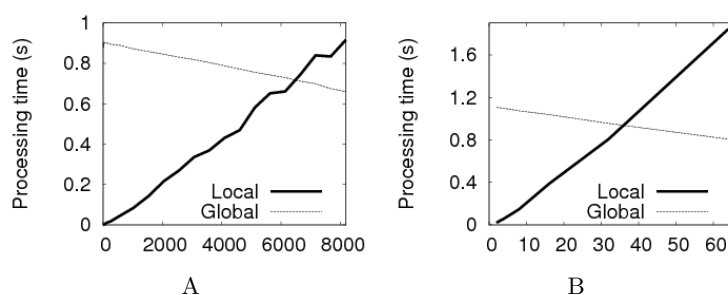


Fig. 3. Processing time comparison of the two region merging approaches (time given in seconds). (A) By the number of sets (size 2) merged. (B) By the size of the 256 sets merged.

merging is used in an interactive way or when we merge regions during a local process. In the case of a more global process, we prefer to use the global approach of the region merging as it will be generally more efficient and also have a more predictable processing time in function of the image size.

6 Conclusion

In this paper, we have presented two different approaches of the region merging operation in the 3D topological map. This work shows that the topological map could be efficiently modified in order to represent the evolution of the representation of 3D images during an image processing operation.

We have detailed the local approach of the region merging in 3D topological maps. It processes by applying local modifications to the three components of the topological map: the combinatorial map, the intervoxel matrix and the inclusion tree. The local method allows an efficient processing when the number of merged regions remains small compared to the map size. The configuration of the selected regions also influences the processing time of the region merging when there is included regions. Another drawback of this approach is that the selected regions to merge have to be in only one connected component. If it is not the case, the local region merging have to be applied to each connected component.

To overcome issues of the local approach, we have proposed a second method of region merging. It aims to merge by connected component any number of regions by processing the whole map at once. As shown in the experiments, the processing time does not change even if there are included regions or if there are several connected components. On the contrary, the processing time of the global approach tends to decrease when the number of merged regions increases. This method gives a better way to merge regions in an automated way.

Thus, we have two region merging algorithms on 3D topological maps that allow users to process regions interactively or automatically. These are the first needed tools for image processing with the 3D topological maps.

The next step of our work is to use these operations in a real bottom-up segmentation process. Our idea is to change the symbolic merging step of the global approach to merge regions according to a criterion, and then applying the last steps of the algorithm to produce the segmented image. We also want to study the opposite operation of the merge called region splitting. The aim of this operation is to divide a region into several smaller regions given a criterion, which may be a cut surface or an homogeneity measure. A splitting operation will give the ability to implement a split-and-merge segmentation in the 3D topological map.

References

- [1] Y. Bertrand, G. Damiand, and C. Fiorio. Topological map: Minimal encoding of 3d segmented images. pages 64–73, Ischia, Italy, may 2001. IAPR-TC15.
- [2] J.-P. Braquelaire and J.-P. Domenger. Representation of segmented images with discrete geometric maps. 17(10):715–735, 1999.
- [3] T. H. Cormen, C. E. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [4] G. Damiand. *Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d*. Thèse de doctorat, Université Montpellier II, décembre 2001.
- [5] G. Damiand, Y. Bertrand, and C. Fiorio. Topological model for two-dimensional image representation: definition and optimal extraction algorithm. *Computer Vision and Image Understanding*, 93(2):111–154, february 2004.
- [6] G. Damiand and P. Lienhardt. Removal and contraction for n-dimensional generalized maps. In *Discrete Geometry for Computer Imagery*, volume 2886/2003 of *Lecture Notes in Computer Science*, pages 408–419, Naples, Italy, november 2003. Springer Berlin / Heidelberg.
- [7] G. Damiand and P. Resch. Split and merge algorithms defined on topological maps for 3d image segmentation. *Graphical Models*, 65(1-3):149–167, May 2003.
- [8] C. Fiorio. A topologically consistent representation for image analysis: the frontiers topological graph. Number 1176, pages 151–162, Lyon, France, november 1996.
- [9] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split and merge procedure. pages 424–433, 1974.
- [10] E. Khalimsky, R. Kopperman, and P.R. Meyer. Boundaries in digital planes. *Journal of Applied Mathematics and Stochastic Analysis*, 3(1):27–55, 1990.
- [11] W.G. Kropatsch and H. Macho. Finding the structure of connected components using dual irregular pyramids. In *Discrete Geometry for Computer Imagery*, pages 147–158, *invited lecture*, september 1995.
- [12] P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82, 1991.
- [13] A. Rosenfeld. Adjacency in digital pictures. *Information and Control*, 26(1):24–33, 1974.
- [14] R. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.