A Generic and Parallel Algorithm for 2D Image Discrete Contour Reconstruction

Guillaume Damiand, David Coeurjolly

A Generic and Parallel Algorithm for 2D Image Discrete Contour Reconstruction[⋆]

Guillaume Damiand and David Coeurjolly

Université Lyon 1, LIRIS, UMR5205, CNRS, F-69622 Villeurbanne Cedex, France
{*guillaume.damiand,david.coeurjolly*}*@liris.cnrs.fr*

Abstract. In this paper, we present a generic topological and geometrical framework for the digital reconstruction of complex contours from labeled images. The proposed technique is based on combinatorial map simplifications guided by digital straight segments. We illustrate the genericity of the framework with a parallel contour reconstruction algorithm.
Keywords. combinatorial maps, 2D contour recontruction, discrete geometry.

1 Introduction

Image processing algorithms often need to compute, to extract or to analyze information contained in images. These information can be computed by decomposing an image into regions and by extracting topological and geometrical features from them. In this paper we present a generic algorithm to compute discrete reconstructions of a multi-region image using combinatorial maps and digital straight segments (DSS for short).

Indeed, DSSs play an important role in polygonalization of discrete object contours [1–5]. Recognition algorithms are used to reconstruct a binary curve, *i.e.* construct a polygonal representation of the digital curve (see Fig. 1). More precisely, a classical approach is to decompose the digital curve into maximal DSS and then to extract a representative edge per maximal DSS. If the problem is quite simple if the contour is topologically equivalent to a circle (simple closed curve) or to a point (simple open curve), problems arise when more complex digital objects are considered such as branching, nested contours, ...

In computer vision, many segmentation algorithm outputs are decompositions of the image into labeled regions such that pixels in a region have uniform features (image intensities, texture characteristics,...). To represent regions and to be able to perform efficient operations on them, a topological data structure is required in order to describes boundaries and adjacency information. There are many different structures to represent the region boundaries of a given image, the

[⋆] Paper published in Proceedings of 4th International Symposium on Visual Computing ISVC, LNCS 5359, pp. 792-801, 2008. Thanks to Springer Berlin Heidelberg. The original publication is available at `http://dx.doi.org/10.1007/978-3-540-89646-3_78` first one being the Region Adjacency Graph (RAG) [6]. However, a RAG does not describe all the information contained in the image (like multi-adjacencies or inclusion relations). To represent all the information, several models based on combinatorial maps were defined **[7–10]**. The main advantage of these models is to describe the subdivision of regions into cells: vertices, edges and faces; and to describe all the incidence and adjacency relations between these cells and thus to represent the topology of the image.

In this paper, we present a generic discrete contour reconstruction algorithm that uses both combinatorial maps to control and maintain the topology of the reconstructed curves, and DSS recognition algorithms. The 2D combinatorial map allows us to link DSS parameters to edges, and to merge two DSS by removing a vertex, which gives a simple and efficient algorithm. Moreover, the algorithm can be easily parallelized by adding mutexes on edges and use the model to avoid concurrent access on a same vertex or a same edge. This shows the interest of our approach.

In section 2, we first introduce DSS, DSS recognition algorithms and combinatorial maps. In section 3, we present the sequential reconstruction algorithm of complex discrete contours, and the parallel approach in section 4. Section 5 finally presents some experiments.

2 Preliminaries

2.1 Digital Straight Segments and Digital Contour Reconstruction

In the literature, many DSS characterizations exist (see [5] or [11] for a complete survey), we consider here a digital straight line (DSL) as the set of pixels $(x, y) \in \mathbb{Z}^2$ satisfying:

$$\mu \leq ax - by < \mu + |a| + |b| \tag{1}$$

with $a,\ b, \mu \in \mathbb{Z}$. Hence, b/a is the DSS slope and μ its intercept. According to [12], the resulting set of pixels is a $4-$arc, which means that each pixel of the DSL has exactly two 4-adjacent neighbors (see Fig. 1). A DSS is defined as a finite connected subset of a DSL.

Based on this definition, a recognition problem may arise: given a set of grid points, does there exists a DSS containing it ? In the literature, many algorithms have been proposed [1–5]. In the following, we do not go further into algorithmic details, we just focus on a main result: adding a new 4-adjacent pixel to a DSS, deciding if the resulting set is still a DSS and updating the DSS parameters can be done in constant time $O(1)$. Hence, we have a simple greedy algorithm to compute in linear time a discrete contour segmentation into maximal DSS (see Fig. $1-b$): we start from a pixel and choose a direction, then we add neighboring 4-adjacent pixels one by one. If the recognition test fails, we stop the current DSS and start a new one. In this framework, even if some results exist concerning the reconstruction into minimal number of DSS [13], some problems remain concerning the choice of the starting point and of the direction. Note also that if

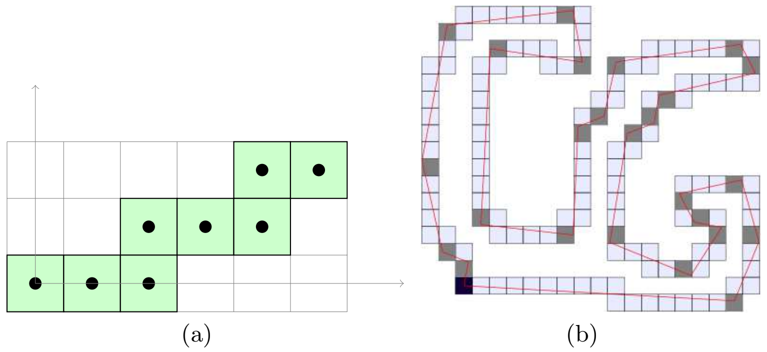


Fig. 1. Illustration of a DSS with parameters $a = 1$, $b = 2$ and $\mu = 0$ (a) and an example of the discrete curve reconstruction process (b).

the contour is complex (not a simple curve), such an algorithm cannot be used directly.

In our framework, we consider a reconstruction based on a union predicate between two adjacent DSS. More precisely, a naive algorithm to decide if the union of two DSS S and T (*i.e.* the union of the two grid point sets) is a DSS or not can be done in $O(|T|)$. Note that more complex algorithms using preimages [3] can be design to obtain a computational cost in $O(\log(\alpha))$ where α corresponds to largest side of the bounding box containing S and T.

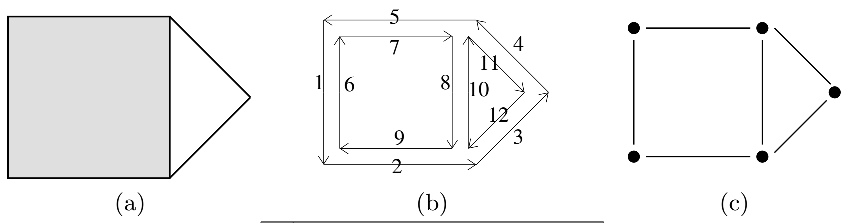
2.2 Combinatorial Maps

A combinatorial map is a mathematical model of space subdivision representation based on planar map [14, 15]. The subdivision of a 2D topological space is a partition of the space into 3 subsets whose elements are *cells* of 0, 1 and 2 dimension (respectively called vertices, edges and faces, and noted i-cell for a i-dimensional cell).

Intuitively a 2D combinatorial map (or 2-map) is constructed by decompositions and splits of a 2D object into faces, edges and vertices. The basic element of a 2-map is called a *dart* (sometimes called half-edge in 2D). Each dart is incident to a vertex, an edge and a face. Darts are linked together with two one-to-one mappings β_1 and β_2 which keep the structure of the subdivision: β_1 connects one dart belonging to an edge to the dart of the next edge of the same face; β_2 connects one dart belonging to an edge to the dart of the other face of the same edge (see example Fig. 2). When two darts d_1 and d_2 are such that $\beta_i(d_1) = d_2$ $(1 \leq i \leq 2)$, we say that d_1 is i-sewn to d_2.

Definition 1 (2D combinatorial map [16]). *A 2D combinatorial map is a triplet $M = (D, \beta_1, \beta_2)$ where:*

1. *D is a finite set of darts;*



	1	2	3	4	5	6	7	8	9	10	11	12
β_1	2	3	4	5	1	7	8	9	6	11	12	10
β_2	6	9	12	11	7	1	5	10	2	8	4	3

Fig. 2. 2D combinatorial map example. (a) A 2D object. (b) Corresponding combinatorial map. Darts are represented by numbered black arrows (to represent the orientation). Two darts 1-sewn are drawn consecutively, and two darts 2-sewn are concurrently drawn and in reverse orientation. (c) Corresponding subdivision made of 5 vertices, 6 edges and 3 faces (by counting the infinite face). We use sometimes this representation instead of drawing all the darts of the map as in (b) to make figures lighter.

2. β_1 *is a* permutation[1] *on D;*
3. β_2 *is an* involution[2] *on D;*

Combinatorial maps encode subdivisions and incidence relations between all the different cells of the space, and so represent the topology of this space. Thanks to this model and its implementation [**10**], we can directly (i.e. in $O(1)$) retrieve all the different relations by darts and one-to-one mappings. We note $v(d)$ (resp. $e(d)$, $f(d)$) for the vertex (resp. edge, face) incident to dart d.

Moreover, several operations exist which allow to modify a combinatorial map. The main operation used in this paper is the removal of a i-cell, called i-removal operation, which removes the i-cell and merges the two incident *(i+1)*-cells. The i-removal operation is possible only for degree[3] one or two cell. Indeed, otherwise it is not possible to decide how to connect cells around the removed cell. Thanks to combinatorial maps, it is possible to test in $O(1)$ if the degree of a vertex is 2 (see [17, 10] for more details on removal operations and algorithms).

In this work, we use combinatorial maps to describe regions contained in a *labeled image*. Regions of such image are the maximal sets of 4-connected pixels with same label. Note that we can consider any type of image and use the color of each pixel as a label.

Moreover, we use the cellular framework that decomposes the digital space pixels into linels, pointels and faces [5]: linels are one dimensional elements separating two pixels and pointels are zero dimensional elements between linels.

[1] A *permutation* on a set S is a one-to-one mapping from S onto S.
[2] An *involution* f on a set S is a one-to-one mapping from S onto S such that $f = f^{-1}$.
[3] The degree of a i-cell c is the number of distinct *(i+1)*-cells incident to c.

3 The Sequential Algorithm for 2D Image Contour Reconstruction

The main principle of the sequential algorithm, presented in Algorithm 1, is first to compute a combinatorial map where each edge corresponds to a linel between two pixels belonging to two different regions (by using algorithm presented in [10]). Then we scan the vertices of the map and remove each degree two vertex such that the union of both edges incident to the vertex is still a DSS.

Algorithm 1: Sequential discrete reconstruction of contours of a labeled image.

```
Input: A labeled image $I$.
Output: A discrete reconstruction of contours of $I$.
$M \leftarrow$ combinatorial map where each edge corresponds to a linel of $I$;
foreach dart $d$ of $M$ do
    if the degree of $v(d)$ is 2 then
        let $e_1$ and $e_2$ the two edges incident to $v(d)$;
        if $e_1 \cup e_2$ is a DSS then
            Remove $v(d)$;
            update DSS attributes of the new edge;
```

The main loop of Algorithm 1 considers each dart successively, and processes only degree two vertices. Indeed vertices with degree greater than 2 are at the junction of several branches and thus cannot be removed. During this loop, if the current vertex is removed, we jump over removed darts and continue the loop with the next dart.

At the end of the algorithm, we have considered each vertex of the map, and we have removed the ones such that the union of both incident edges is still a DSS. This allows us to prove that each edge in the resulting combinatorial map is a maximal DSS, since no more vertex can be removed.

The key point of Algorithm 1 is that each operation used in the main loop is a local process, and that each adjacency and incidence relations can be retrieved directly in $O(1)$ thanks to our model. This gives us a generic algorithm and allows us to easily parallelized this method.

Fig. 3 illustrates each step of Algorithm 1: first we consider an input labeled image (Fig. 3a), in Fig. 3b the initial combinatorial map where each edge corresponds to a linel between two pixels with different labels, and in Fig. 3c one result obtain at the end of the algorithm where each edge corresponds to one maximal DSS.

As discussed in section 2, the 2D discrete reconstruction depends on the starting point, and here, on the order in which vertices are processed in the main loop of Algorithm 1. If darts are processed locally, *i.e.* if the dart at step i is adjacent to the dart at step $i-1$, we obtain similar on simple closed curves as

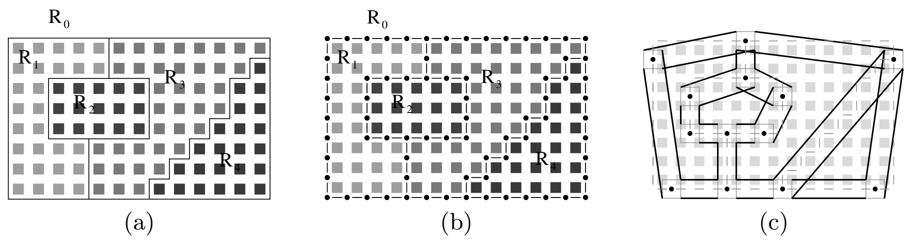


Fig. 3. Example of 2D discrete reconstruction of contours of a labeled image. (a) A labeled image with 4 regions (plus the infinite region R_0). (b) Initial combinatorial map where each edge of the map corresponds to a linel between two pixels with different labels. (c) Result obtain after the discrete reconstruction. Each edge of this map corresponds to one maximal DSS.

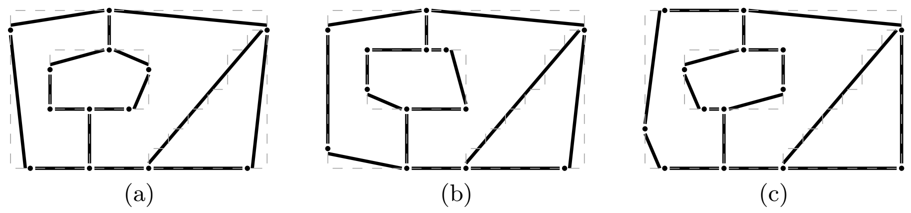


Fig. 4. 3 possible results of 2D discrete reconstruction of contours of the same labeled image (image given in Fig. 3a). Results can be different for the shape of DSS, but can also be different in number of DSS. (a) and (b) Two results with 13 vertices, 16 edges (and thus 16 DSS) and 5 faces. (c) One result with 14 vertices, 17 edges and 5 faces.

with the classical greedy techniques detailed in the preliminaries. Using the union predicate and since the combinatorial map data structure supports constant time access to darts, we can easily introduce a randomization on the for each loop with the idea to overcome or distribute error propagations. Each resulting map is composed of maximal DSS in the sense that the union of two adjacent DSS is no more a DSS. However, depending on the order of processed vertices, DSS can be different, and moreover the number of DSS can also be different (like for example the map shown in Fig. 4b which contains 16 DSS and the map shown in Fig. 4c which contains 17 DSS). A discussion on this point is addressed in the conclusion.

4 The Parallel Algorithm for 2D Image Contour Reconstruction

The main idea of the parallel algorithm, is to split the main loop of the sequential one into several loops, and execute each loop in parallel using threads. Since the processing of a vertex is local to the two incident edges, this parallelization can be achieved easily just by adding mutexes on edges to avoid concurrent accesses. Algorithm 2 details the pseudo-code executed by each thread independently.

Algorithm 2: Parallel discrete reconstruction: one thread algorithm.

```
Input: A combinatorial map $M$ where each edge corresponds to a linel of the
       image;
       $L$ a list of darts to process.
Output: Sub-part of $M$ corresponding to $L$ is modified such that each edge
        corresponds to one maximal DSS.
Let $P$ a stack of darts to consider latter;
foreach dart $d$ of $L \cup P$ do
    if the degree of $v(d)$ is 2 then
        let $e_1$ and $e_2$ the two edges incident to $v(d)$;
        take if possible the mutex $m_1$ associated to $e_1$ (resp. $m_2$ associated to
        $e_2$);
        if both mutexes $m_1$ and $m_2$ are not taken then
            Push $d$ in $P$;
            Release the taken mutex;
        else
            if $e_1 \cup e_2$ is a DSS then
                Remove $v(d)$;
                update DSS attributes of the new edge;
            Release mutexes $m_1$ and $m_2$;
```

The main loop of this algorithm is very similar to Algorithm 1 one. We can point out two main differences: firstly, the list of darts to process is now only a part of the whole darts of the map since other darts are processed by other threads. Secondly, we have added particular process to avoid conflict due to concurrent access. For that, we try to take the mutexes associated to edges e_1 and e_2. If both mutexes are taken, we have a guaranty that there is no other thread processing a vertex incident to either edge e_1 or e_2. Note that mutexes are taken in non-blocking mode to avoid inter-blocking situations.

If at least one mutex cannot be taken, the current vertex cannot be processed because incident edges can change (if the other thread removes its current vertex). Thus, we need to re-test this vertex later to consider the case if the adjacent vertex is not removed by the other thread. For that, the vertex is pushed in a stack of vertices to consider later.

5 Experiments

We have implemented both algorithms, sequential and parallel, and run some experiments to compare times required by both versions. Our experiments were

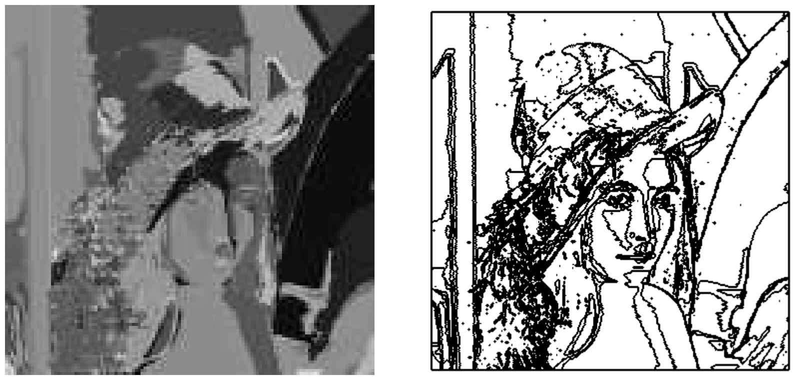


Fig. 5. `Lena` labeled image used for our experiments, and the resulting reconstruction.

made with an "Intel Core2 Duo CPU" at 2.20GHz, with 1.5 mega-bytes of memory and 4 kilo-bytes of cache. We have used six segmented images: `Airplane, Baboon, Cornouaille, Goldhill, Lena` (see Fig. 5) and `Peppers`. The size of each image is 512 × 512 pixels, except `Cornouaille` which is 256 × 256 and `Goldhill` which is 720 × 576. Number of regions in each image is respectively 6360, 6704, 5409, 8005, 6197 and 5764.

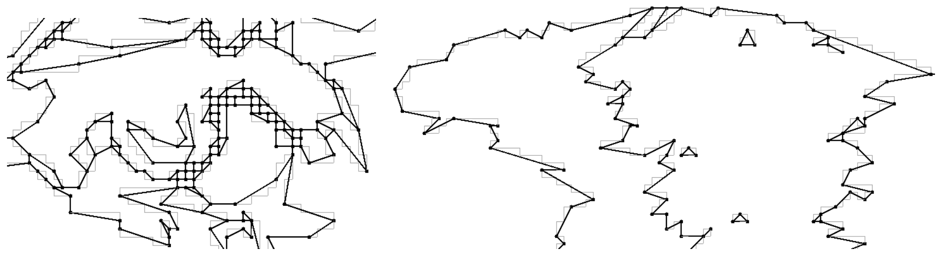


Fig. 6. Two zooms in image of Fig. 5: the first one on the left eye of Lena, and the second one on the hat. Points show the extremities of DSS, linels are drawn in grey, and DSS in black.

Fig. 6 shows two zooms of the discrete reconstruction of Lena. We can remark that there are many small DSS due to small regions in the labeled image (sometimes regions made of only one pixel). We can improve the reconstruction by removing these small regions in a pre-processing step.

Table 1 gives the times taken by our methods (values are average made on 20 tests for each image). First column gives the time required to compute the initial combinatorial map where each edge corresponds to a linel. Second and third columns give the times required to compute the discrete reconstruction by the sequential and parallel algorithm. In average, the gain of the parallel method is about 32%. This gain is already important, furthermore we hope obtain better results by using processor with more cores or by using computer with several processors.

Table 1. Times in mili-seconds taken to compute the first combinatorial map (*Ext.*), to compute the discrete reconstruction by the sequential (*Seq.*) and parallel (*Par.*) algorithm. The last column (*Speed-up*) gives the percentage of gain of the parallel method on the sequential one.

	Ext.	Seq.	Par.	Speed-up
`Airplane`	216	35	22	35%
`Baboon`	234	60	42	30%
`Cornouaille`	72	17	11	34%
`Goldhill`	316	89	63	28%
`Lena`	180	39	26	32%
`Peppers`	240	41	27	34%

Lastly, we can notice that the time required by the computation of the first combinatorial map take the most important part of the global process. To improve this step, we can for example use algorithms bases on precodes [10]. Furthermore, a perspective is to propose a parallel algorithm to extract this initial combinatorial map.

6 Conclusion

In this paper, we have presented an algorithm which allows to compute the discrete contour reconstruction of any labeled image. By using a combinatorial map which describes the image, we use cells of the subdivision to add DSS parameters, and we use incident and adjacency relations to merge two DSS when it is possible. This gives a simple and efficient algorithm. Moreover, since all processes are local, the algorithm can easily be parallelized with independent threads sharing the same map. Our experiments show that the gain of the parallel algorithm is about 32% with a “Intel Core2 Duo CPU”. Since our algorithm is fully parallel, the speed of our method will be improved by using future generation of processors with more cores.

As a consequence our framework genericity, this work allows to use other types of reconstruction (e.g. polygonalization, approximation…). Indeed, the principle is always the same: start from the initial map which represents all the linels, and simplify it by testing one (or several) geometrical criterion. Moreover, we can easily add the notion of critical points (vertices that cannot be removed) by marking these particular vertices and do not process them during our algorithm.

Another advantage of our method is the possibility to process vertices in any order. This can be easily achieved by adding a step which sort vertices of the initial combinatorial map in a specific order. This allows to consider different orders and to overcome problems that could appear during the DSS reconstruction. For example, we can use random orders to obtain results which can be statistically studied. Another example is to sort vertices depending on the angle between adjacent edges, in order to begin by considering vertices between align edges, and consider at the end of the process salient angles. Lastly, we plan to extend this method in 3D by using 3D combinatorial maps [18].

References

1. Kovalevsky, V.A.: New definition and fast recognition of digital straight segments and arcs. In: Proc. 10th Intl. Conf. on Pattern Recognition. (1990) 31–34
2. Dorst, L., Smeulders, A.W.M.: Decomposition of discrete curves into piecewise straight segments in linear time. In: Contemporary Mathematics. Volume 119. (1991)
3. Lindenbaum, M., Bruckstein, A.: On recursive, $o(n)$ partitioning of a digitized curve into digital straigth segments. IEEE Trans. on Pattern Analysis and Machine Intelligence **15** (1993) 949–953
4. Debled-Rennesson, I., Reveillès, J.P.: A linear algorithm for segmentation of digital curves. In: International Journal on Pattern Recognition and Artificial Intelligence. Volume 9. (1995) 635–662
5. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Series in Computer Graphics and Geometric Modelin. Morgan Kaufmann (2004)
6. Rosenfeld, A.: Adjacency in digital pictures. Information and Control **26** (1974) 24–33
7. Domenger, J.: Conception et implémentation du noyeau graphique d'un environnement 2D1/2 d'édition d'images discrètes. Thèse de doctorat, Université Bordeaux I (1992)
8. Fiorio, C.: A topologically consistent representation for image analysis: the frontiers topological graph. In: DGCI. Volume 1176 of LNCS., Lyon, France (1996) 151–162
9. Brun, L., Domenger, J.P.: A new split and merge algorithm with topological maps and inter-pixel boundaries. In: The fifth International Conference in Central Europe on Computer Graphics and Visualization. (1997)
10. Damiand, G., Bertrand, Y., Fiorio, C.: Topological model for two-dimensional image representation: definition and optimal extraction algorithm. Computer Vision and Image Understanding **93** (2004) 111–154
11. Klette, R., Rosenfeld, A.: Digital straightness–a review. Discrete Applied Mathematics **139** (2004) 197–230
12. Reveillès, J.P.: Géométrie discrète, calcul en nombres entiers et algorithmique. PhD thesis, Université Louis Pasteur - Strasbourg (1991)
13. Feschet, F., Tougne, L.: On the min dss problem of closed discrete curves. Discrete Applied Mathematics **151** (2005) 138–153
14. Edmonds, J.: A combinatorial representation for polyhedral surfaces. Notices of the American Mathematical Society **7** (1960)

15. Tutte, W.: A census of planar maps. Canad. J. Math. **15** (1963) 249–271
16. Lienhardt, P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. Computer-Aided Design **23** (1991)
17. Damiand, G., Lienhardt, P.: Removal and contraction for n-dimensional generalized maps. In: Procedings of 11th Discrete Geometry for Computer Imagery. Volume 2886 of LNCS., Naples, Italy (2003) 408–419
18. Damiand, G.: Topological model for 3d image representation: Definition and incremental extraction algorithm. Computer Vision and Image Understanding **109** (2008) 260–289