Combinatorial Maps for 2D and 3D Image Segmentation

Guillaume Damiand, Alexandre Dupas

▶ **To cite this version:**

Guillaume Damiand, Alexandre Dupas. Combinatorial Maps for 2D and 3D Image Segmentation. Digital Geometry Algorithms, Springer, pp.359-393, 2012, 10.1007/978-94-007-4174-4_12 . hal-01353024

HAL Id: hal-01353024
https://hal.science/hal-01353024v1

Submitted on 24 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Combinatorial Maps for 2D and 3D Image Segmentation

Guillaume Damiand and Alexandre Dupas

1 Introduction

In the image analysis framework, image segmentation is one of the main issues, and probably the most discussed one in the literature. Image segmentation methods were subject of numerous research papers. Surveys on existing approaches can be found, for example, in [32, 23, 36, 22, 24]. Image segmentation is usually the first step before any algorithm for computer vision, such as, i.e., object recognition, is applied. The segmentation operation consists of grouping the elements of an image, usually known as pixels or voxels (for 2D and 3D images, respectively), in homogeneous areas called regions. Each region is uniform regarding some properties based on intensity (gray levels), texture, or colors. The set of regions forms a partition of the image elements, and thus any pixel or voxel of the image belongs to exactly one region.

For efficiency purposes, some segmentation algorithms need an effective representation of the image partition and operations so that the partition can easily be modified. The cost related to the partition modification is closely connected to the data structure related to the particular segmentation algorithm. One of the first data structures described in the literature is the region adjacency graph, called RAG [33]. Regions of the image are represented by the vertices of a graph, and the edges are linking each pair of vertices if the two corresponding regions are adjacent. However, a RAG does not describe all the relations between regions in 2D images, and in higher dimensions it is even more so. To overcome this issue, other solutions are available in the literature. In dual graphs approach [35, 28], two graphs are used: a RAG is coupled with its dual. While dual graphs solve some of the issues of RAG,

Guillaume Damiand
Université de Lyon, CNRS, LIRIS, UMR5205, F-69622, France, e-mail: guillaume.damiand@liris.cnrs.fr

Alexandre Dupas
Inserm, Unit 698, 75018 Paris, France e-mail: alexandre.dupas@gmail.com

this solution is not complete, and moreover it is meant to only represent 2D images partitions.

Several solutions based on 2D combinatorial maps have been proposed [6, 14, 7, 19, 5, 9], and then extended in 3D [4, 8]. These models have advantages that justify their use. First, combinatorial maps fully describe the topology of the image partition in regions, that is, the representation of all cells of the partition and all the incidence and adjacency relations between these cells. Second, combinatorial maps allow efficient algorithms to retrieve information and modify the partition. Finally, combinatorial maps are defined in any dimension allowing to easily generalize the algorithms. For these reasons, combinatorial map based models have been used in many works in image segmentation [2, 3, 12, 26, 15].

The objective of this chapter is to present all the basic knowledge required to implement a generic 2D and 3D image segmentation algorithm based on combinatorial maps. We start in Sect. 2 by introducing combinatorial maps and the related definitions. Then, we give several notions about 2D and 3D images, and finally we define topological maps, which are specific combinatorial maps describing a partition of an image in regions. In Sect. 3 we give the generic segmentation algorithm based on the global merging operation. The algorithm is generic because it is defined in any dimension, and it takes as parameter two functions allowing to control its behavior depending on the image properties. To illustrate the genericity, we introduce four different segmentation criteria based on different information. As our interest in combinatorial maps is to fully describe the topology of image partitions, we use this asset and propose a segmentation criterion based on a topological invariant: the Betti numbers. In Sect. 4, we show how to compute Betti numbers in the topological map framework, and we propose two segmentation criteria that allow, with the generic segmentation algorithm, to explore a segmentation method that take into account some topological features. Finally, Sect. 5 presents some experiments of 2D and 3D image segmentation using the different criteria proposed in this chapter.

2 Topological Maps

In this section, we introduce the basic notions leading to the definition of a topological model used to represent 2D and 3D image partition. We start by introducing combinatorial maps that describe subdivided objects in any dimension. Then, we recall the basic notions related to images (pixels, voxels, adjacency, regions, inter-elements). Last, we present 2D and 3D topological maps which are specific combinatorial maps that describe 2D and 3D image subdivisions.

2.1 Combinatorial Maps

An nD combinatorial map is a model representing an nD subdivided orientable object by describing all its cells, and all the neighborhood relations between these cells. We denote by *i-cell* a cell in dimension i: 0-cells are called *vertices*, 1-cells *edges*, 2-cells *faces*, and 3-cells *volumes*. Neighborhood relations are defined on the basis of the *incidence* and *adjacency* relations. Two cells c_1 and c_2 are adjacent if they have the same dimension i, and if they share a common $(i-1)$-cell c. In this case, c is said to be incident to c_1 and to c_2. Incidence relation is symmetric: if c_1 is incident to c_2, then c_2 is incident to c_1. Moreover, the incidence relation is extended by transitivity: two cells c_1 and c_2 are incident if there is a path of cells starting from c_1 to c_2 such that each couple of consecutive cells are incident (see Fig. 1(a) for an example in 2D).

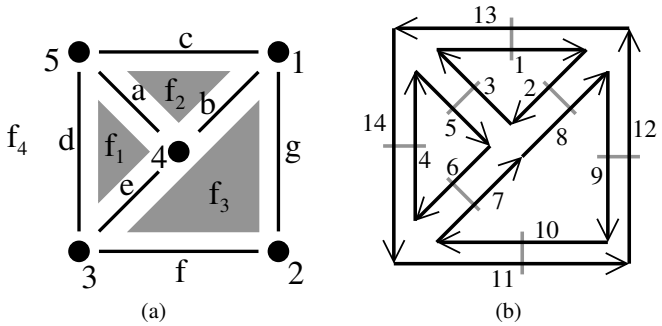


Fig. 1 (a) Example of 2D orientable subdivision. The object is composed of five 0-cells (numbered from 1 to 5), seven 1-cells (labeled from a to g) and four 2-cells (labeled f_1 to f_4). Edges a and b are adjacent since they share vertex 4, and faces f_1 and f_2 are adjacent along edge a. Thus, vertex 4 is incident to edge a, and edge a is incident to face f_1. By transitivity, vertex 4 is incident to face f_1. (b) Corresponding 2D combinatorial map. Darts are represented by numbered black segments ending with arrows. Two darts linked by β_1 are drawn consecutively (for example, $\beta_1(1) = 2$) and two darts linked by β_2 are drawn parallel to each other in reverse orientation connected by a little gray segment (for example, $\beta_2(1) = 13$).

Any orientable nD subdivided object cannot be described by an nD combinatorial maps: only quasi-manifold orientable objects without boundary can. Quasi-manifold means that an object consists only of $(n-1)$D quasi-manifold orientable objects glued together along $(n-1)$-cells. Note that in 2D, quasi-manifolds are manifolds, but this is no more true in higher dimension. "Orientable" means that it is possible to define a global orientation "left" and "right" in each point of the object. Lastly, "without boundary" means that each $(n-1)$-cell is without boundary, and that the boundary of each n-cell is fully described by $(n-1)$-cells.

An nD combinatorial map is defined as a set of basic elements, called *darts*, with one to one mappings defined onto the set of darts. Each dart describes a part of a

0-cell, a 1-cell, ..., an n-cell. The mappings β allow to link together darts and thus group the darts that describe the same cell. Definition 1 gives the definition of nD combinatorial maps (see [29, 30] for definitions and more details on combinatorial maps).

Definition 1 (nD combinatorial map). An *nD combinatorial map* (or an n-map) is a $n+1$-tuple $M = (D, \beta_1, \ldots, \beta_n)$ where:

1. D is a finite set of darts;
2. β_1 is a *permutation*[1] on D;
3. $\forall i : 2 \leq i \leq n$: β_i is an *involution*[2] on D;
4. $\forall i, j$: $1 \leq i < i+2 \leq j \leq n$: $\beta_i \circ \beta_j$ is an involution[3].

Intuitively, given a dart of an n-map, β_1 gives the next dart of the same face, and β_i gives the dart of the adjacent i-cell (see example in Fig. 1(b)). The property that the represented object is without boundary ensures that any dart is linked to another dart by β_i, which is a prerequisite of the permutation property. Moreover, the quasi-manifold property ensures that there are at most two i-cells along each $(i-1)$-cell belonging to the same $(i+1)$-cell, which explains why β_i is an involution. We denote by β_0 the permutation β_1^{-1}. Note that this is only a notation and not a new permutation.

The last line of the definition ($\beta_i \circ \beta_j$ is an involution) ensures the quasi-manifold property. This condition guarantees that when two darts of two i-cells are linked by β_{i+1}, all the darts of the two cells are also linked two by two by β_{i+1}. Intuitively, this means that two i-cells are either disjointed or completely identified, but they cannot be partially identified.

Now, thanks to darts and the β relations, we can retrieve the cells of the subdivision which are implicitly represented in combinatorial maps by sets of darts and by the *orbit* notion.

Definition 2 (orbit). Let $\Phi = \{f_1, \ldots, f_k\}$ be a finite set of permutations on D. We denote by $\langle \Phi \rangle$ the permutation group generated by Φ. This is the set of permutations obtained by any composition and inversion of permutations contained in Φ. The *orbit* of a dart d with respect to Φ is defined by $\langle \Phi \rangle(d) = \{\phi(d) | \phi \in \langle \Phi \rangle\}$.

Intuitively, the orbit $\langle \Phi \rangle(d)$ is the set of darts that can be reached from d by using any combination of permutations in Φ. Each i-cell of an nD combinatorial map is obtained by a specific orbit:

Definition 3 (i-cell). Let $M = (D, \beta_1, \ldots, \beta_n)$ an n-map, $d \in D$, and $i \in \{0, \ldots, n\}$. The i-cell incident to d, denoted by $c_i(d)$, is:

- if $i = 0$: $\langle \beta_1 \circ \beta_2, \ldots, \beta_1 \circ \beta_n \rangle(d)$;
- otherwise: $\langle \beta_1, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_n \rangle(d)$.

---

[1] A *permutation* on a set D is a one to one mapping from D onto D.

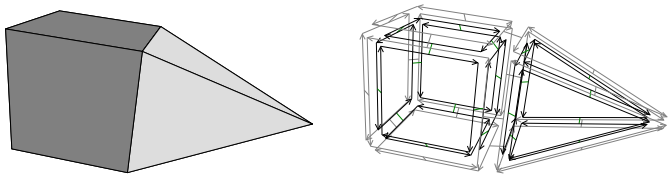
[2] An *involution* f on a set D is a one to one mapping from D onto D such that $f = f^{-1}$.

[3] $\beta_i \circ \beta_j$ is the composition of both permutations: $(\beta_i \circ \beta_j)(x) = \beta_i(\beta_j(x))$.

Due to the definition of cells as sets of darts, the incident and adjacency relations on cells can easily be tested. Two distinct cells c_1 and c_2 are *incident* if $c_1 \cap c_2 \neq \emptyset$, and two distinct i-cells c_1 and c_2 are *adjacent* if there are two darts $d_1 \in c_1$ and $d_2 \in c_2$ satisfying $d_1 = \beta_i(d_2)$ (or $d_2 = \beta_i(d_2)$ in the case of 1-cells).

We can see an example of 2D combinatorial map in Fig. 1(b). In 2D, a 2-map is a triplet $M = (D, \beta_1, \beta_2)$ and the last line of the definition ($\beta_i \circ \beta_j$ is an involution) does not apply. Face f_3 (2-cell) corresponds to $\langle \beta_1 \rangle(7) = \{7,8,9,10\}$, edge a (1-cell) corresponds to $\langle \beta_2 \rangle(3) = \{3,5\}$ and vertex 1 (0-cell) corresponds to $\langle \beta_1 \circ \beta_2 \rangle(13) = \{2,9,13\}$. Edge b and face f_3 are incident since $b = \{2,8\} \cap f_3 = \{7,8,9,10\} \neq \emptyset$, and f_2 and f_3 are adjacent since $2 \in f_2$, $8 \in f_3$ and $2 = \beta_2(8)$.

One main advantage of combinatorial maps is their definition in any dimension. We can see an example in 3D in Fig. 2. A 3D combinatorial map is a 4-tuple $M = (D, \beta_1, \beta_2, \beta_3)$ such that $\beta_1 \circ \beta_3$ is an involution. A 3-cell (volume) corresponds to $\langle \beta_1, \beta_2 \rangle(d)$; a 2-cell (face) to $\langle \beta_1, \beta_3 \rangle(d)$; a 1-cell (edge) to $\langle \beta_2, \beta_3 \rangle(d)$; and a 0-cell (vertex) to $\langle \beta_1 \circ \beta_2, \beta_1 \circ \beta_3 \rangle(d)$.



(a) (b)

Fig. 2 (a) Example of 3D orientable subdivision. The object is composed by nine 0-cells, sixteen 1-cells, ten 2-cells and three 3-cells (the cube, the pyramid, plus an unbounded volume not drawn). (b) Corresponding 3D combinatorial map having 144 darts. The 64 darts describing the unbounded volumes are drawn in gray thin lines.

2.2 Removal Operations

The basic operations used to simplify a combinatorial map are the *removal operations*. These operations were first defined on generalized maps [11, 10] and then transposed to combinatorial maps [20, 21]. An i-removal operation allows to remove an i-cell while possibly merging the two incident $(i+1)$-cells around the removed cell. There are several removal operations since we can remove an i-cell in an n-map for any $i : 0 \leq i < n$.

However, removing an i-cell is not always possible: there is a constraint that the i-cell must satisfy: the notion of *removable cell*. Intuitively the removable constraint ensures that there are at most two $(i+1)$-cells around the removed cell. Otherwise it is not possible to automatically decide how to modify the different $(i+1)$-cells while removing the i-cell.

Definition 4 (Removable cell). An i-cell c in an n-map is *removable* if $0 \leq i < n$, and if $i = n-1$ or $\forall d \in c, \beta_{i+1} \circ \beta_{i+2}(d) = \beta_{i+2} \circ \beta_{i+1}^{-1}(d)$.

As explained above, the notion of being removable is related to the number of $(i+1)$-cells around the removed cell which is the notion of degree of a cell.

Definition 5 (Cell degree). Let c be an i-cell in an n-map, with $0 \leq i < n$. The *degree* of c is the number of $(i+1)$-cells incident to c.

We can easily prove that if an i-cell c is removable, then its degree is at most 2 (i.e. equal to 1 or 2 since it is not possible to have a degree equal to 0 by definition of cells).

Now, we give a generic definition of removal operations. This is Definition 6 which is valid for any removable i-cell with $0 < i < n$. Intuitively, the definition "modifies" only β_i relations for the neighboring darts of the removed i-cell.

Definition 6 (i-removal operation). Let $M = (D, \beta_1, \ldots, \beta_n)$ be an n-map, and c a removable i-cell, with $0 < i < n$. The combinatorial map $M' = (D', \beta'_1, \ldots, \beta'_n)$ obtained by removing c from M is defined by:

- $D' = D \setminus c$;
- $\forall j : 1 \leq j \leq n$: $\forall d \in D'$:
  - if $j = i$ and $d \in \beta_i^{-1}(c) \setminus c$: $\beta'_j(d) = (\beta_j \circ \beta_{j+1})^k(\beta_j(d))$,
    with k the smaller positive integer such that $(\beta_j \circ \beta_{j+1})^k(\beta_j(d)) \notin c$;
  - otherwise: $\beta'_j(d) = \beta_j(d)$.

$\beta_i^{-1}(c)$ is the set of darts $\{\beta_i^{-1}(d) | d \in c\}$. This is the set of darts which are neighbors of c by β_i. In the removal definition, only darts of $\beta_i^{-1}(c) \setminus c$ have their β_i modified since all the darts of c are removed by the operation. For all these darts, the new β'_j are defined by $\beta'_j(d) = (\beta_j \circ \beta_{j+1})^k(\beta_j(d))$. Intuitively, we jump over the removed darts until we obtain a dart that does not belong to c.

We can see in Fig. 3 two examples of removal operations in a 3D combinatorial map. First, we want to remove a 2-cell represented by the darts drawn in bold black in Fig. 3(a). The darts, neighbors of the removed darts by β_2 and drawn in bold gray, are the only darts modified by the operation, for example $\beta'_2(1) = \beta_2 \circ \beta_3 \circ \beta_2(1) = 2$. Second, we want to remove the edge represented by the darts drawn in bold black in Fig. 3(b). Only neighbor darts of these darts by β_0 are modified by the operation, for example $\beta'_1(3) = \beta_1 \circ \beta_2 \circ \beta_1(3) = 4$. Note that this edge is removable in the combinatorial map of Fig. 3(b) (in the sense of Definition 4) but not in the combinatorial map of Fig. 3(a) because there are more than two 2-cells incident to this edge.

We have a specific case for 0-removal operation, that is the removal of a vertex. This is due to the inhomogeneous definition of combinatorial maps where β_1 is a permutation while other β are involutions. We can see in Definition 7 the main difference with the generic definition. Indeed, we need to modify all the β links of neighbor darts of the removed cell. Moreover, the definition of the modified relation is different.

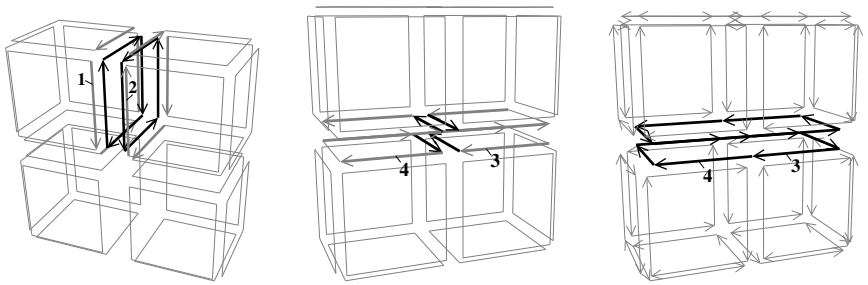


Fig. 3 Example of some removals operations in a 3D combinatorial map (only partially drawn). (a) The initial configuration from which we want to remove the 2-cell drawn in bold black. (b) The resulting map obtained after the 2-removal. The two volumes incident to the removed 2-cell are merged. Now we want to remove the 1-cell drawn in bold black. (c) The resulting map obtained after the 1-removal. The two faces incident to the removed edge are merged.

Definition 7 (0-removal operation). Let $M = (D, \beta_1, \ldots, \beta_n)$ an n-map, and c a removable 0-cell. The combinatorial map $M' = (D', \beta'_1, \ldots, \beta'_n)$ obtained by removing c from M is defined by:

- $D' = D \setminus c$;
- $\forall j : 1 \leq j \leq n$: $\forall d \in D'$:
  - if $d \in \beta_j^{-1}(c) \setminus c$: $\beta'_j(d) = \beta_j((\beta_1)^k(d))$,
    with k the smaller positive integer such that $\beta_j((\beta_1)^k(d)) \notin c$;
  - otherwise: $\beta'_j(d) = \beta_j(d)$.

Here, contrary to the general case, we do not need to only modify β_i, but all the β. This is due to the definition of cells. Indeed, given an i-cell c, with $0 < i < n$, we know that for any dart $d \in c$, $\beta_j(d) \in c$, $\forall j \neq i$. This property ensures that only β_i has to be modified, but this is no more true for vertices.

2.3 Images, Regions and Inter-elements

In this chapter we are interested in 2D and 3D images segmentation, and thus we now recall some usual notions. A pixel (resp. voxel) is an element of the discrete space $\mathbb{Z}^2$ (resp. $\mathbb{Z}^3$), associated with a value (for example a color or a gray level). A 2D image is a set of pixels and a 3D image is a set of voxels.

Two pixels $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ are 4-adjacent if $|x_1 - x_2| + |y_1 - y_2| = 1$. Two voxels $v_1 = (x_1, y_1, z_1)$ and $v_2 = (x_2, y_2, z_2)$ are 6-adjacent if $|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| = 1$. A 4-path (resp. 6-path) between two pixels (resp. voxels) b and e is a sequence of pixels (resp. voxels) $(b = e_1, \ldots, e_k = e)$ such that any couple of consecutive pixels (resp. voxels) of the path are 4-adjacent (resp. 6-adjacent).

A set of pixels S (resp. voxels) is 4-connected (resp. 6-connected) if there is a 4-path (resp. 6-path) between any couple of pixels (resp. voxels) of S with all the elements of the path belonging to S.

A region R is a maximal set of 4-connected pixels (resp. 6-connected voxels) having the same label. To avoid having a specific process for the image borders, we consider an infinite region, usually called R_0, that surrounds the image (i.e. this region is the complement of the image). If a region R_j is completely surrounded by a region R_i we say that R_j is *enclosed* in R_i.

We can see in Fig. 4 an example of a 2D labeled image and the illustrations of the main notions.

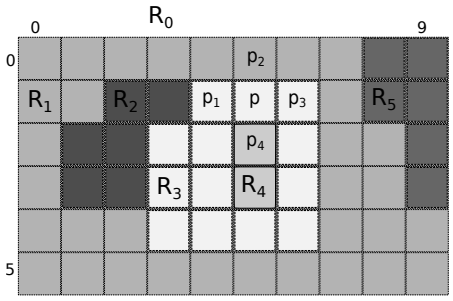


Fig. 4 Example of a 2D labeled image having 10 pixels in x axis, and 6 pixels in y axis. Pixel $p = (5,1)$ belongs to region R_3. Pixel p is 4-adjacent to pixels $p_1 = (4,1)$, $p_2 = (5,0)$, $p_3 = (6,1)$ and $p_4 = (5,2)$. The image contains five regions (labeled from R_1 to R_5), plus the infinite region R_0. R_4 is enclosed into region R_3, and regions R_2 and R_3 are enclosed into region R_1.

In the interpixel or intervoxel framework [27, 25], pixels or voxels are not the only considered elements. We also consider all the elements of a cellular decomposition of the paving of $\mathbb{Z}^n$. In 2D, pixels are unit squares, linels are unit segments separating two squares, and pointels are points at the extremity of linels. In 3D, voxels are unit cubes, surfels are unit squares separating two voxels, and linels and pointels definitions are similar to the 2D case (see example in Fig. 5).

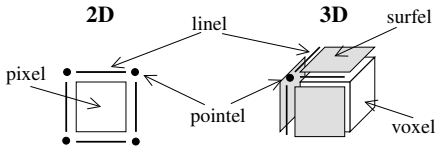


Fig. 5 Interpixel and intervoxel elements.

2.4 Topological Maps

A topological map is a combinatorial map describing an image. For this reason, topological maps are not as general as combinatorial maps since they have some particular properties implied by the specifics of the described partitions. We present here only the main principle of topological maps. Interested readers can found more details in the referenced papers.

Given a 2D labeled image, the problem is to describe the partition in regions using a combinatorial map. We want to describe the multi-adjacency relations between regions, to retrieve all the regions adjacent to a given one, and to know how many times two regions are adjacent. Thus, we have to build a combinatorial map where each edge corresponds exactly to a maximal set of linels between two regions [1, 9]. However, using a combinatorial map is not enough to represent all the information contained in the image. We need to add an interpixel matrix to represent the geometry of the regions, and a tree of regions to describe the enclosed relations between the regions. This is the notion of topological map given in Definition 8.

Definition 8 (2D topological map). Given a 2D labeled image, its 2D topological map is a data structure composed of three parts:

- A minimal 2D combinatorial map describing the partition of regions: the external boundary and each cavity of each region is described by one cycle of darts, linked by β_1. Note that the infinite region is a special case since it does not have an external boundary but only one internal boundary. Each edge of the combinatorial map corresponds to a maximal frontier between two regions;
- An interpixel matrix containing all the linels that belong to a region boundary, and all the pointels having a degree greater than two;
- An enclosed tree of regions describing all the enclosed relations between the regions.

The 2D combinatorial map is minimal in number of cells which means there is no combinatorial map that describes the same partition in regions with a smaller number of cells. This minimal property ensures that we represent each maximal frontier between two regions by exactly one edge in the map. We can see in Fig. 6 an example of a 2D labeled image and the corresponding 2D topological map.

The topological map definition can be extended in 3D by using a 3D minimal combinatorial map, an intervoxel matrix, and an enclosed tree of regions. This definition is given in Definition 9 and detailed in [8].

Definition 9 (3D topological map). Given a 3D labeled image, its 3D topological map is a data structure composed of three parts:

- A minimal 3D combinatorial map describing the partition of regions: the external boundary and each cavity of each region is described by volumes (i.e., orbits $\langle \beta_1, \beta_2 \rangle$). As in 2D, the infinite region is a special case since it does not have an external boundary; it has only one internal boundary. Each face of the combinatorial map corresponds to a maximal frontier between two regions, and each edge corresponds to a maximal junction between faces;

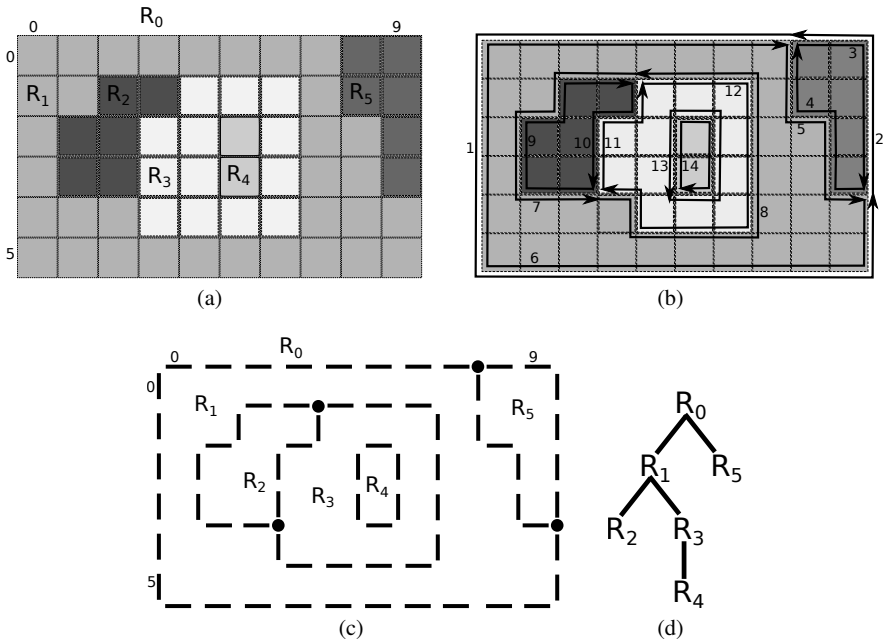


Fig. 6 Example of 2D topological map. (a) A 2D labeled image. (b) The minimal combinatorial map describing the image. Darts are numbered from 1 to 14. Regions R_2, R_4, and R_5 have only one external boundary, thus only one β_1 cycle of darts. Regions R_1 and R_3 have one cavity, and thus two β_1 cycles of darts, one for their external boundary, and one for their cavity. (c) The interpixel matrix. (d) The enclosed tree of regions.

- An intervoxel matrix containing all the surfels that belong to a region boundary, all the linels having a degree greater than two, and all the pointels having a degree greater than two;
- An enclosed tree of regions describing all the enclosed relations between regions.

Figure 7 shows an example of a 3D labeled image and the corresponding topological map.

3 Image Segmentation Algorithm

Now, we present a bottom-up segmentation algorithm based on topological maps [15]. The guiding principle consists in successively merging adjacent regions satisfying a given criterion starting with an initial partition represented by a topological map. The initial partition can be a partition where each pixel/voxel is in its own region, a partition where pixels/voxels having same color are grouped, a partition which is an over-segmentation of the initial image, or any other kind of partition.

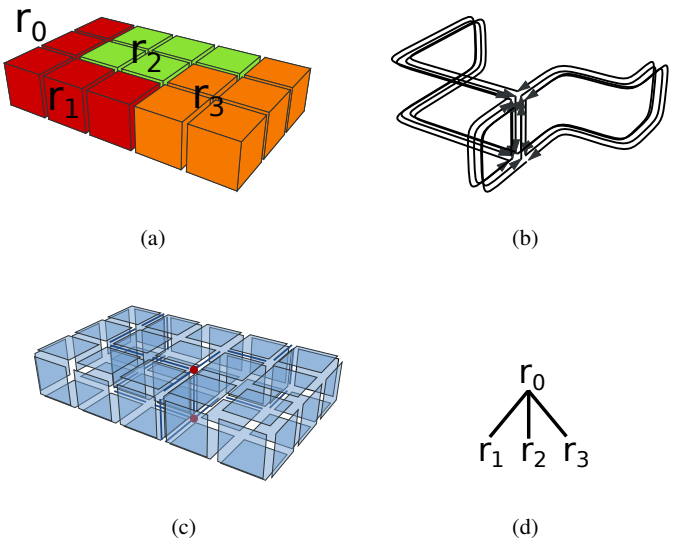


Fig. 7 An example of 3D topological map. (a) A 3D labeled image. (b) The minimal combinatorial map describing the image. (c) The intervoxel matrix. (d) The enclosed tree of regions.

A bottom-up approach is chosen here since it allows to incrementally update regions characteristics without the need of running through all the pixels/voxels of the region. Such incremental update is not possible for top-down or mixed approaches.

3.1 The Global Merging Algorithm

The principle of the global merging algorithm consists in merging each couple of adjacent regions of a given topological map satisfying a given criterion. To optimize the algorithm and minimize topological maps modifications, we decompose the process in two steps:

1. Symbolic merging: regions are "merged" by using union-find trees; no modification is made on the topological map;
2. Effective merging: the topological map is modified to group together all regions that belong to the same union-find tree.

For the symbolic merging, we use an union-find tree forest [34] to partition the set of regions in disjointed sets. Each region has a reference to its father in a union-find tree. To handle trees, we use two functions: $find(r)$ which, given a region r, finds the root of the union-find tree, and $union(r_1, r_2)$ which, given two different regions r_1 and r_2 merges their trees. Before starting the symbolic merging, we initialize the pointer of each region to the region itself. This means that each region r is in its own

union-find tree since we have $find(r) = r$. During the symbolic merging, an external process successively merges couples of regions. The external process can be driven by the user who selects some regions to merge during an interactive session, or as we will see in the next section, by a segmentation algorithm. The merging of the two union-find trees containing regions r_1 and r_2 is simply done by modifying the father pointer of the root of one tree to the root of the other tree. To improve the complexity of this step, we use the two following heuristics:

- We put the smaller tree (in terms of the number of regions) as a child of the largest one during the *union* function;
- We compress the path from region r and its root r' during the $find(r)$ function; for that we assign the father of all the regions between r and its root to r'.

Thanks to these two heuristics, it is proved in [34] that union and find operations on disjoint sets represented by trees can be considered as constant time operations. Figure 8 shows a simple example of the symbolic merging.

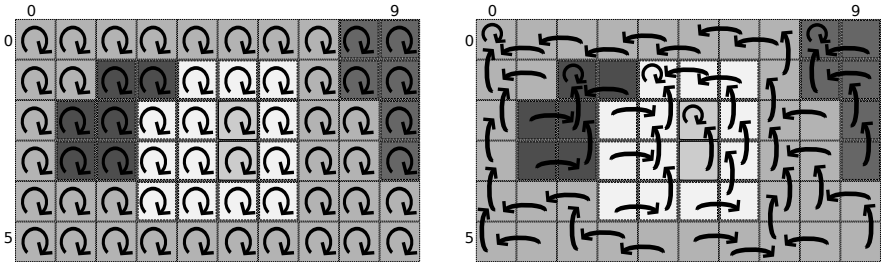


Fig. 8 Illustration of the symbolic merging. The father relation of union-find trees is represented by bold black arrows. (a) An initial forest of union-find trees where each pixel belongs to its own region; thus each pixel is the root of its own union-find tree. (b) Result of the symbolic merging of each couple of 4-adjacent pixels having same label (in this example we do not use the two heuristics to simplify the figure). Each pixel has an unique father, and thanks to these union-find tree, we can simply test if two pixels belong to the same region just by testing if $find(p_1) = find(p_2)$.

We use the union-find trees during the second step of the global merging algorithm. The objective of this step, presented in Algo. 1, is to merge all the regions that belong to a same union-find tree in the topological map. The principle of this algorithm is to test for each couple of adjacent regions r_1 and r_2 whether they belong to the same union-find tree. In this case we remove all the $(n-1)$-cells that separate the two regions.

We can efficiently test each couple of adjacent regions thanks to the relations represented in the topological map. We run through all the darts of the map. Each dart d belongs to its region r_1, and each dart $\beta_n(d)$ belongs to a region r_2 adjacent to r_1. The `foreach` loop allows testing all the couples of adjacent regions in a time linear to the number of darts. Moreover, if two regions are adjacent k times,

Algorithm 1: Effective merging step

Input: An nD topological map T;
Disjointed sets partitioning the regions of T.
Result: T is modified so that all regions belonging to the same union-find tree are merged in an unique region.

let $toSimplify$ be an empty set of darts;
foreach *dart* d *in* T **do**
  if $find(region(d)) = find(region(\beta_n(d)))$ **then**
    $toSimplify \leftarrow toSimplify \cup$ one dart per $(n-2)$-cell incident to $c_{n-1}(d)$;
    remove $c_{n-1}(d)$;
simplify($toSimplify$);
recompute the enclosed tree of regions;

they are separated by k different $(n-1)$-cells. If the two regions belong to the same union-find tree, all these $(n-1)$-cells are removed since during the `foreach` loop, we will consider one dart for each of these cells, and for each of these darts the condition $find(region(d)) = find(region(\beta_n(d)))$ will be satisfied.

If we remove an $(n-1)$-cell, we might need to simplify the topological map because it does not possess the minimal number of cells property anymore. Indeed, the degree of each $(n-2)$-cell incident to the remove cell is decreased by one, and thus these cells can possibly be simplified. To solve this issue, during the effective merging, we keep one dart per each $(n-2)$-cell incident to a removed $(n-1)$-cell, and these cells will be tested during a post-processing simplification step. This simplification step is specialized for 2D and 3D topological maps since in 2D we only need to test and possibly simplify vertices, while in 3D we need to test and possibly simplify edges, then test and possibly simplify vertices (see [9] and [8] for details and Fig. 9 for an example in 2D).

3.2 The Segmentation Algorithm

In our approach, the segmentation algorithm, given in Algo. 2, is only a specific case of the global merging algorithm. In fact, as we have seen in the previous section, it is enough to control the symbolic merging step to propose a segmentation algorithm. Then, the effective merging step will modify the initial partition and will produce the topological map representing the result of the segmentation.

Algorithm Algo. 2 is a generic segmentation algorithm taking as parameters an nD topological map which describes an initial partition, two functions giving a weight for each $(n-1)$-cell, and a criterion to determine whether an $(n-1)$-cell must be removed.

The initial topological map can be either initialized with a region for each image element, or it can be any initial partition, for example obtained by a pre-segmentation step. The two functions allow to tune the segmentation algorithm by

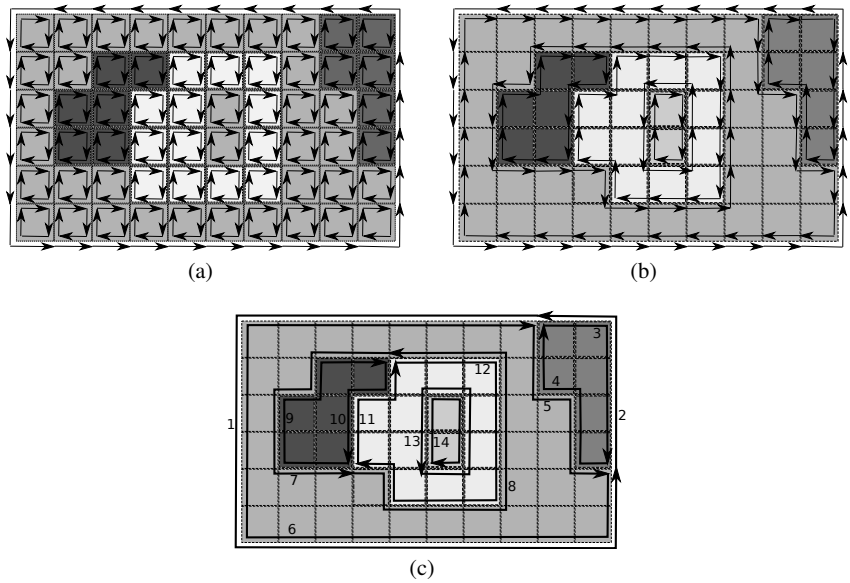


Fig. 9 Example of effective merging. (a) A topological map describing a 2D labeled image where each pixel is in its own region. (b) The combinatorial map obtained after the first step of the effective merging (removal of 1-cells) given the disjoint sets shown in Fig. 8(b). This map is not minimal since some vertices are removable. (c) The combinatorial map obtained after the simplification step. This is the 2D topological map of the partition in regions given by the disjoint sets.

Algorithm 2: Generic Segmentation Algorithm

Input: An nD topological map T associated with an image I;
A function $weight(c)$ giving a weight of each $(n-1)$-cell;
A boolean function $criterion(d)$ returning true if cell $c_{(n-1)}(d)$ must be removed.
Result: T represents the optimal segmentation of I for the *weight* and *criterion* functions.

$L \leftarrow$ sorted list of one dart per $(n-1)$-cell of T according to *weight*;
foreach *dart* $d \in L$ **do**
  $r_1 \leftarrow region(b)$; $r_2 \leftarrow region(\beta_n(b))$;
  if $r_1 \neq r_2$ ***and*** $criterion(d)$ **then**
    $r \leftarrow$ union(r_1, r_2);
    update region r;
effective merging of regions of T;

using any kind of information associated with cells and regions of the topological map (see Sect. 3.3 for some examples of such functions).

Before processing the regions, we start by sorting the list of $(n-1)$-cells according to the *weight* function. This allows us to start with the consideration of a couple of regions that are close with respect to the given *weight*. In Sect. 5, we show that this approach gives a better result that processing edges randomly.

The main loop of the algorithm consists of testing all couples of adjacent regions, and in merging them if the given criterion returns a true value. The merging is only made in the symbolic way using the union find trees. The last line of the algorithm is the effective merging step presented in Algo. 1.

The algorithm is generic since it is defined in any dimension, and can be tuned easily by modifying the weight and criterion functions. Its time complexity is linear in number of darts of the topological map, times the complexity of the criterion function. As we will see in the next section, most of our criteria have constant time complexity, and thus the segmentation algorithm becomes linear in number of darts of the map.

3.3 Different Criteria of Segmentation

The main interest of our approach is the genericity and the possibility to mix different criteria associated with a different type of cells (for example a colorimetric criteria associated with the regions, and a gradient associated with the edges). To illustrate these interests, we present here four segmentation criteria, based on:

- The range of gray levels in the regions;
- The gradient of the $(n-1)$-cells separating regions;
- External/internal contrasts of the regions and $(n-1)$-cells;
- The size of the regions and the gradient of $(n-1)$-cells.

All these criteria are defined for nD topological maps, and use different kind of information (cells, adjacency and incidence relations, geometrical information, etc.). Moreover, the information used can often be initialized without additional complexity cost during the topological map construction, and can often be used and updated in constant time which results in efficient segmentation algorithms.

3.3.1 Range of Gray Levels

The first version is a basic criteria based on gray levels of pixels. We associate with each region an interval $[g_{min}, g_{max}]$ of min gray level and max gray level of all the pixels contained in the region. The value of each interval is initialized during the construction of the topological map (without modifying the complexity of the construction).

The weight function is given in Algo. 3. We define the weight of the removal of a cell $c_{n-1}(d)$ by the difference between the length of the new interval of the two merged regions around $c_{n-1}(d)$ and the maximum length of the two original intervals of r_1 and r_2. Thanks to this definition, the weight is zero if one interval is included into another, or if the two interval are equal. The weight increases if the two region intervals move away.

Algorithm 3: Weight function based on gray levels ranges

Input: A dart d of an nD topological map.
Result: The weight of the removal of $c_{n-1}(d)$.

$r_1 \leftarrow region(d)$; $r_2 \leftarrow region(\beta_n(d))$;
$dist_0 \leftarrow \max(r_1.g_{max} - r_1.g_{min}, r_2.g_{max} - r_2.g_{min})$;
$dist_1 \leftarrow \max(r_1.g_{max}, r_2.g_{max})$ - $\min(r_1.g_{min}, r_2.g_{min})$;
return $dist_1 - dist_0$;

The removal criterion given in Algo. 4 returns true if the length of the new interval after the merging of the two regions around $c_{n-1}(d)$ is smaller than a threshold τ given by the user.

Algorithm 4: Removal criterion based on gray level ranges

Input: A dart d of an nD topological map;
A threshold τ.
Result: `true` iff $c_{n-1}(d)$ must be removed.

$r_1 \leftarrow region(d)$; $r_2 \leftarrow region(\beta_n(d))$;
$dist_1 \leftarrow \max(r_1.g_{max}, r_2.g_{max})$ - $\min(r_1.g_{min}, r_2.g_{min})$;
return $dist_1 < \tau$;

Since we associate an interval with each region, we have a direct access to each information associated with the darts ($region(b)$, $\beta_n(d)$, $r.g_{min}$ and $r.g_{max}$) and thus the two algorithms have a constant time complexity.

Moreover, when two regions r_1 and r_2 are merged during the symbolic merging, we can easily, and in constant time, update the interval of the region r which is the union of r_1 and r_2: we only have to set $r.g_{min} \leftarrow \min(r_1.g_{min}, r_2.g_{min})$, and $r.g_{max} \leftarrow \max(r_1.g_{max}, r_2.g_{max})$.

3.3.2 Gradient on $(n-1)$-cells

Another criterion often used in image segmentation is an image gradient. We associate with each $(n-1)$-cell c a value corresponding to the gradient of c. More precisely, for each inter-element i belonging to c the gradient is the sum of the absolute difference of the gray levels of the two image elements around i. As for the

previous criterion, the gradient of each $(n-1)$-cell is initialized during the construction of the topological map without modifying the complexity of the construction. However, gradients are associated with $(n-1)$-cells and not with the regions. This illustrate an interesting feature of topological maps as we can associate information with any cell and possibly mix information associated with different type of cells.

The first approach is to use the gradient in the segmentation algorithm is to consider the gradient of each cell $c_{n-1}(d)$ as its weight. Then, the merging criterion returns true if the gradient of $c_{n-1}(d)$ is smaller than a threshold τ given by the user. The problem of this approach is that it does not account for the fact that if we remove an $(n-1)$-cell, it will result in the merging of the two regions around this cell, and thus it will remove all the $(n-1)$-cells between these two regions. If we only consider the current $(n-1)$-cell, the merging criterion can return true because its associated gradient is small, even if merging the two incident regions will remove other $(n-1)$-cells with stronger gradients.

To solve this problem, we define the weight function given in Algo. 5 which takes into account all the $(n-1)$-cells which will be removed if we merge the two regions incident to the considered cell $c_{n-1}(d)$.

Algorithm 5: Weight function based on gradients

Input: A dart d of an nD topological map.
Result: The weight of the removal of $c_{n-1}(d)$.

```
res ← 0;
r2 ← region(β_n(d));
foreach dart d' ∈ c_n(d) do
    if d' not marked treated then
        if region(β_n(d')) = r2 then
            res ← res + c_{n-1}(d').gradient;
        mark treated all the darts of c_{n-1}(d');
return res;
```

In this algorithm, we run through all the darts $d' \in c_n(d)$ to sum all the gradients of all the $(n-1)$-cells separating the same two regions.

The removal criterion given in Algo. 6 returns true if the sum of all the gradients of all the $(n-1)$-cells separating the two regions is smaller than a threshold τ given by the user. This test can be achieved by reusing the *weight* function and comparing its value with the threshold.

Algorithm 6: Removal criterion based on gradients

Input: A dart d of an nD topological map;
A threshold τ.
Result: `true` iff $c_{n-1}(d)$ must be removed.

```
return weight(d)< τ;
```

Due to these definitions, all the $(n-1)$-cells that separate the same couple of regions have the same weight and thus give the same answer for the removal criterion. Moreover, when two regions are merged during the symbolic merging, there is no modification to apply to gradient values. Lastly, during the simplification step of the effective merging, if two $(n-1)$-cells are merged, the gradient of the new $(n-1)$-cell is the sum of the two gradients of the original cells. Note that the complexity of the weight and removal criterion algorithms is linear in number of darts of $c_n(d)$.

3.3.3 External and Internal Contrasts

This method is based on the segmentation algorithm proposed by Felzenszwalb and Huttenlocher in [17, 18] which uses a criterion based on intensity differences between neighboring pixels in 2D images. In the original work, authors used region adjacency graphs to represent the image partition. In [15] we transposed this method for topological maps and extended it to 3D, but as with the previous criteria, we can extend this criterion in any dimension thanks to the topological maps.

To define this criterion, we associate with each region an internal contrast, called *int*, which is the minimal gray level difference between two adjacent image elements of the region. We also associate an external contrast (called *ext*) with each $(n-1)$-cell c, which is the minimal value for each inter-element i belonging to c of the absolute difference of the gray level of the two image elements around i. These contrasts are initialized during the construction of the topological map.

Thanks to these two values, we can define the weight function given in Algo. 7 which is equal to the external contrast of the considered $(n-1)$-cell.

Algorithm 7: Weight function based on contrasts

Input: A dart d of an nD topological map.
Result: The weight of the removal of $c_{n-1}(d)$.

return $c_{n-1}(d).ext$;

According to the original work of Felzenszwalb and Huttenlocher [17, 18], the removal criterion given in Algo. 8 returns true if the external contrast of the considered $(n-1)$-cell is smaller than the minimal internal contrast of the two incident regions. These two internal contrasts are weighted by a threshold function f allowing the user to control the degree to which the external variation can actually be larger than the internal variations. We can use any defined positive function for f. In practice, we use the same function than the one proposed by the authors of the original method: $f(r) = k/|r|$ with $|r|$ the size of region r (i.e., its number of image elements), and k a constant defined by the user and allowing to tune the algorithm.

The two algorithms have a constant time complexity since we have a direct access to all the information. Moreover, when two regions are merged in region r during the symbolic merging, it is proved in [17] that the internal contrast of r is equal to

Algorithm 8: Removal criterion based on contrasts

Input: A dart d of an nD topological map;
A threshold function f.
Result: `true` iff $c_{n-1}(d)$ must be removed.

$r_1 \leftarrow region(d)$; $r_2 \leftarrow region(\beta_n(d))$;
return $c_{n-1}(d).ext \leq min(r_1.int + f(r_1),\ r_2.int + f(r_2))$;

the external contrast of the removed cell, while there is no modification on external contrasts. Thus, we can update the contrast in constant time during the symbolic merging.

3.3.4 Size of Regions

We present now the last criterion which illustrates one more time the interest of our generic approach. Indeed, thanks to topological maps, we can use different cells and associate different type of information with these cells, but we can also mix colorimetric criteria and geometrical ones. To illustrate this possibility, we present a criterion which mixes the size or regions and the range of gray levels given in Sect. 3.3.1. Thanks to these two pieces of information, the proposed criterion allows removing all the small regions (i.e., regions with size smaller than a given threshold) by merging them to the closer regions in their neighborhood (closer in the sense of range of gray levels).

For the weigh function, we use the function already given in Algo. 3 based on the distance between the ranges of the two regions, and the range of the region after the union. This allows to start processing the $c_{n-1}(d)$ cells that separate closer regions.

For the removal criteria, we use Algo. 9 which do not use range of gray levels, but which is based on region sizes. This criterion returns true if one of the two regions incident to the considered $(n-1)$-cell is smaller than a given threshold. Thanks to this criterion, we ensure that at the end of the process, all the regions have their size greater than the threshold. As we will see in our experiments, this process can be used as post-processing step to remove small regions that are often due to noise in the original image.

Algorithm 9: Removal criterion based on sizes

Input: A dart d of an nD topological map;
A threshold τ.
Result: `true` iff $c_{n-1}(d)$ must be removed.

$r_1 \leftarrow region(d)$; $r_2 \leftarrow region(\beta_n(d))$;
return $size(r_1) < \tau$ *or* $size(r_2) < \tau$;

The range of gray levels and the size of regions can be computed during the topological map extraction, and can both be updated in constant time after the merging of two regions during the symbolic merging.

4 Betti Numbers and Topological Criteria

We presented a generic segmentation algorithm that is parametrized by two functions *weight*, and *criterion*. The algorithm controls the symbolic merging step of the global merge operation to produce the optimal segmentation of the image. We also introduced several criteria based on intensity values of the image, or on simple geometrical property like the region size. While such kind of criteria are commonly used to define segmentation algorithm, topological features are equally important to retrieve useful information from the image segmentation. The presence of a cavity in a 2D region representing a plain object might indicate of a defect in the production factory. Tunnels in a 3D region might be an indication of a segmentation error if, for instance, the segmented object is known for not having any. Thus, we want to define criteria that account for topological features, and that can be used in conjunction with geometric or colorimetric criteria.

In computational topology, topological invariants characterize some of the topological feature we want to take into account. One of the most well known invariants is the Euler characteristic χ of a 2D surface which is linked to the genus of the surface. The genus allows to discriminate between sphere-like surfaces and torus-like surfaces. Definition 10, found in [31], defines the value of the Euler characteristic as the alternating sum of number of cells for cellular complexes. For instance, in 2D, the Euler characteristic is equal to the sum of the number of faces and the number of vertices minus the number of edges.

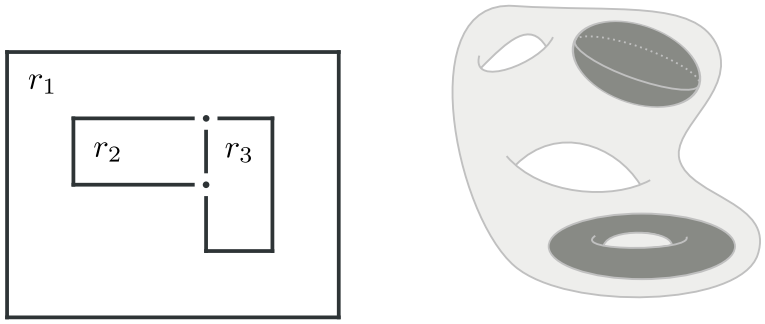
Definition 10. The Euler characteristic $\chi(r)$ of a region r in an nD space is defined as: $\chi(r) = \sum_{i=0}^{n}(-1)^i \#c_i$, with $\#c_i$ is the number of i-cells belonging to r.

While the Euler characteristic allows to uniquely qualify a topological property for 2D surfaces, it is not the same for 3D volumes like regions in a 3D topological map. Other invariants are, for instance, Betti numbers that express some topological features like the number of cavities or the number of tunnels of a region, and work both in 2D and 3D.

4.1 Betti Numbers

In computational topology, Betti numbers are the rank of the homology group generators, an advanced topological invariant. From a practical point of view, Betti numbers of an object represent the number of *holes* in each dimension. The first

Betti number, noted b_0, counts connected components of the object. In 2D, the second Betti number, b_1 counts the number of cavities in the object. In 3D, the second Betti number, b_1 counts tunnels and the third Betti number, b_2 is equal to the number of cavities of the object. For closed oriented nD objects, as regions in a 2D or 3D image, Betti numbers b_k with $k > n$ are equal to zero. For instance, non-zero Betti numbers of the 2D region r_1 in Fig. 10(a) are $b_0 = 1$ and $b_1 = 1$ and the non-zero Betti numbers of the 3D object presented in Fig. 10(b) are $b_0 = 1$, $b_1 = 3$ and $b_2 = 2$.



(a) (b)

Fig. 10 Example of Betti numbers. (a) In 2D for region r_1: $b_0 = 1$, $b_1 = 1$; (b) In 3D: $b_0 = 1$, $b_1 = 3$, $b_2 = 2$.

Definition 11 of [31], establishes a relation between Betti numbers and the Euler characteristic of an object. The relation gives an alternative approach to compute the Euler characteristic.

Definition 11. The Euler characteristic $\chi(r)$ of a region r in nD is defined as the alternating sum of Betti numbers: $\chi(r) = \sum_{i \leftarrow 0}^{n} (-1)^i b_i(r)$, where the $b_i(r)$ are the Betti numbers of region r.

Computing Betti numbers allows to use the number of connected components, the number of tunnels (in 3D), or the number of cavities to serve as criteria during a segmentation step. In the following sections, we present how Betti numbers can be computed in topological maps and how they can be used as segmentation criteria.

4.2 Computation Algorithms using Topological Maps

In this section, we present the computation of Betti numbers using information provided by the topological map representing an image partition. We avoid the complex

computation of homology group generators as we only require the rank of these groups. The goal is to compute Betti numbers in 2D and 3D image partitions using the practical definition of Betti numbers. Thus, depending on the dimension of the topological map, we count the number of connected components, the number of tunnels, and the number of cavities to obtain the Betti numbers.

In the following, the computation algorithms of Betti numbers of region r of topological map M are explained. First, we detail the algorithms in the 2D case. Second, we present the modifications needed to handle 3D image partitions.

4.2.1 Computation of Betti Numbers in 2D Image Partition

The number of connected components of region r in a 2D image partition is equal to the first Betti number $b_0(r)$. By definition of topological maps, a region is a 4-connected set of pixels. Thus, each region has only one connected component. The first Betti number is constant and equal to one: $b_0(r) = 1$ for all region r.

The number of cavities of a region r in a 2D image partition is equal to the second Betti number $b_1(r)$. Note that the set of enclosed regions of region r fills the cavities of region r, and each 8-connected component of the enclosed regions fills exactly one cavity of region r. Counting the number of connected components of enclosed regions allows to retrieve the number of cavities, and thus the second Betti number. In the topological map framework, a tree of region represents the enclosed relation.

The tree of regions is organized so that each connected component $R_{enclosed}$ of regions enclosed in a region r is represented in the tree by a representative region which is in direct relation with r. For instance, r has a direct enclosed relation with $r_i \in R_{enclosed}$, and the other regions within $R_{enclosed} \setminus r_i$ can be retrieved using the connected component relation. Thus, in a 2D topological map, the second Betti number $b_1(r)$ of region r is obtained by counting the number of regions having a direct enclosed relation with r.

4.2.2 Computation of Betti Numbers in 3D Image Partition

There are analogies between the definition of the two first Betti numbers in 2D and the definition of the first and third Betti numbers in 3D. By definition of a 3D topological map, regions are 6-connected sets of voxels. Thus, as in 2D, there is only one connected component for each region implying that the first Betti number is constant and equal to one: $b_0(r) = 1$ for any region r.

The third Betti number $b_2(r)$ counts the number of cavities of a region r following the same principle as the second Betti number in a 2D topological map. The 3D region tree represents the enclosed relation, and regions enclosed within region r fill up the cavities of r. The tree of regions is organized as the tree of region in a 2D topological map. Thus, in a 3D topological map, counting the number of regions in direct enclosed relation with region r gives $b_2(r)$, the third Betti number of region r.

4.2.3 Computation of the Second Betti Number in 3D Image Partition

In 3D, the second Betti number $b_1(r)$ counts the number of tunnels of region r. As there is no simple way to determine the number of tunnels of a region in 3D topological maps, we use the relation between the Betti numbers and the Euler characteristic given by Definition 11.

The computation of the Euler characteristic using the alternated sum of volumes, faces, edges, and vertices is only valid if the volume is represented by a cellular complex composed of i-cells homeomorphic to i-balls. As the topological map is not a full cellular complex since it only represents cells that belong to the border of the regions of the partition, the represented cells of a region do not satisfy the prerequisite for direct computation of χ using Definition 10.

In [13], the authors present the computation of the sum of Euler characteristics of each border of region r represented by a 3D topological map. We call this sum Euler characteristic of the border of region r denoted $\chi'(r)$. The computation of the alternating sum of cells belonging to the border of a region is possible since the topological map represents all cells belonging to the border of each region.

In [16], the authors define the notion of implicit cells that allows to convert a region represented in a 3D topological map as a cellular complex. Using the fact that the regions in topological maps are composed of only one connected component of voxels, the authors prove Proposition 1 which gives a relation between χ and χ' for region r represented by a 3D topological map.

Proposition 1. *The Euler characteristic $\chi(r)$ is linked to the Euler characteristic of the border $\chi'(r)$ by the relation $\chi(r) = \chi'(r)/2$ with r – a region of a 3D topological map.*

In Proposition 2, we use Definition 11 and Proposition 1 to obtain the second Betti number of region r of a 3D topological map as a function of the Euler characteristic of the border of r, the number of connected components of r, and the number of cavities of r. The complete proof of Proposition 2 can be found in [16].

Proposition 2. *In a 3D topological map, the second Betti number $b_1(r)$ of region r is given by $b_1(r) = b_0(r) + b_2(r) - \chi'(r)/2$.*

We have defined the computation formulas of Betti numbers for regions represented by 2D and 3D topological maps. But these formulas do not allow to compute the Betti numbers for a set of regions as the ones defined during the segmentation process in the symbolic step of the global merging algorithm. To propose such a feature, we introduce in Sect. 4.3 the incremental computation algorithms of Betti numbers.

4.3 Incremental Computation Algorithms

In Sect. 4.2, we have defined simple formulas that allow to compute the Betti numbers of any region in 2D and 3D topological maps. To use Betti numbers as topo-

logical criteria during segmentation operation, we need incremental computation algorithms that efficiently update Betti numbers during merge operations.

Symbolic regions are a composition of regions merged in a same disjoint-set of regions during the symbolic step of the global merging algorithm. Incremental computation algorithms have to handle symbolic regions as if they already had been merged together. We use special border coverage algorithms to this purpose removing the need to take a special care of such configurations.

In 2D and 3D image partitions represented by topological maps, the number of connected components per region is constant and equals to one. The value of the first Betti number $b_0(r)$ never changes. The second Betti number in 2D and the third Betti number in 3D count the cavities of the region. In Sect. 4.3.1, we propose an incremental approach that updates the number of cavities during the merging of two regions from the symbolic merge step. In Sect. 4.2.3, the number of tunnels in a 3D image partition represented by a topological map cannot be obtained directly. To overcome this issue, we give in Sect. 4.3.2 algorithms used to incrementally compute the Euler characteristic of the border, the number of connected components, and the number of cavities so that we can use the formula provided by Proposition 2 linking these values to the value of the second Betti number in 3D topological maps.

4.3.1 Incremental Computation of the Number of Cavities

The incremental computation of the number of cavities in a topological map during the merging of two regions r_1 and r_2 consists in computing the number of connected components of the enclosed regions after the merge which is the number of internal borders of a region. Let region r_1 and region r_2 be adjacent. In the following, we suppose that region r_1 is not enclosed in region r_2. This leads to three possible configurations (we can assume this because if it is not the case we only have to swap the notations of r_1 and r_2):

1. Merging r_1 and r_2 does not change the number of cavities;
2. Merging r_1 and r_2 removes a connected component of enclosed region, if r_2 fills completely a cavity of r_1;
3. Merging r_1 and r_2 adds new connected components of enclosed regions (either by surrounding new components of enclosed regions or splitting an existing connected component into several chunks).

The number of components of the border of a region is linked to the number of cavities: as there is only one connected component of regions, the number of cavities is equal to the number of borders minus one, the external border. To compute the changes in the number of borders, we use Proposition 3 proved for the 3D case in [16] and also valid in 2D. In the proposition, k is the number of new internal borders.

Proposition 3. *Let r_1 and r_2 be two adjacent regions such that $r_1 < r_2$. We have $\#borders(r_1 \cup r_2) = \#borders(r_1) + \#borders(r_2) + k - 2$, where k is the number of borders containing part of the external border of r_2 and that are not between r_1 and r_2.*

We have now a formula linking the number of cavities of a region with the number of borders of the union of two regions. Let us details the algorithm that we use to effectively compute the number of new borders. First, we define the notion of inner border of two regions r_1 and r_2 as the border composed by the cells that lies between r_1 and r_2. We implement traversal algorithms allowing to run through darts of the border ignoring inner borders. That algorithm is a slightly modified version of the classical border traversing algorithm with the exception that if the next dart to be traversed belongs to an inner border, we ignore the corresponding cell (an edge in 2D), and proceed with the next suitable cell (the next non-inner edge around the vertex in 2D). The principle is to cover the border as if region r_1 and r_2 are merged.

An example presenting inner cells in a 2D topological map is presented Fig. 11. The border traversing algorithm, starting in the top left corner of r_1, and going to the right comes to vertex 1. The classical coverage algorithm would go through the edge e_1. Since we want to cover the border as if $r_1 \cup r_2$ is a unique region, the coverage algorithm proceeds with the next edge that is not marked as inner. Thus the next edge is the one leading to the top right corner.

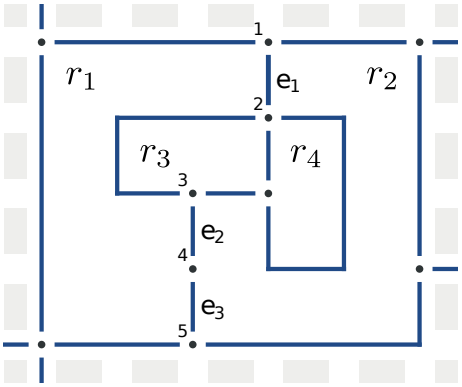


Fig. 11 Example of inner cells in a 2D topological map. The inner cells of the union of regions $r_1 \cup r_2$ are the edges e_1, e_2, e_3, and the vertex 4. Vertices 1, 2, 3, and 5 belong to the boundary of the inner border and thus are not qualified as inner cells.

Now, we implement the border counting mechanism which allows to compute the number of borders of the union of two adjacent regions r_1 and r_2. The algorithm first runs through the external border of r_1 and marks as inner and processed each dart d such as $region(\beta_n(d))$ equals r_2. Then, we count the number of connected components of cells that contains a dart of r_1 without traversing any inner border. Using the previously computed and stored value for the number of borders of region r_1 and region r_2, the algorithm returns the number of borders of the union of r_1 and r_2.

Figure 12 illustrates the incremental border counting algorithm during the evaluation of the union of regions $r_1 \cup r_2$ on a 2D topological map. In the figure, the border counting algorithm, starting from a dart belonging to region r_2 and edge e_1

allows to discover two borders. The new internal border is surrounding regions r_3 and r_4 and the external border surrounding regions r_1 and r_2.

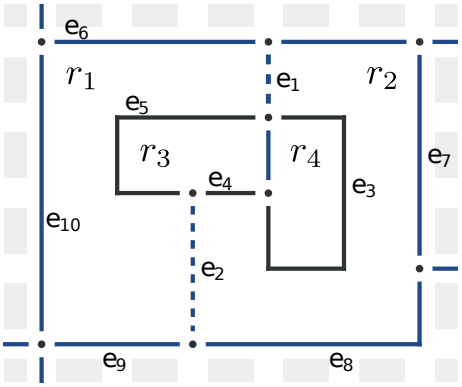


Fig. 12 Illustration of the incremental computation of the number of cavities for the union of regions $r_1 \cup r_2$. Edges e_1 and e_2 are inner cells for $r_1 \cup r_2$. For the purpose of this example, we suppose the starting dart belongs to region r_2 and to the edge e_1. The two borders discovered by the algorithm are the internal border composed of edges e_3, e_4, e_5, and the external border composed of edges e_6 through e_{10}.

Now we can use this algorithm to compute incrementally the number of cavities of two regions that are going to be merged. The differences between the number of cavities in 2D regions and the number of cavities in 3D regions are the cells used to describe the border of a region. In 2D topological maps, border cells are vertices and edges whereas in 3D the border cells are vertices, edges, and faces. The core algorithm does not change except for the orbit used in the border traversal algorithms. In the 2D case, the orbit $\langle \beta_1 \rangle$ gives the darts of the border of a region and the involution β_2 is used to retrieve the region on the other side of the border. In the 3D case, the orbit $\langle \beta_1, \beta_2 \rangle$ describes the border of a region and a dart of the region on the other side of the border is obtained by the involution β_3.

The time complexity of the incremental computation of the number of cavities of the union of regions $r_1 \cup r_2$ depends on the number of darts of the external border of r_2 and the number of darts of the border of r_1 in contact with region r_2.

4.3.2 Incremental Computation of the Number of Tunnels in 3D

The incremental computation of the number of tunnels of a region r represented in a 3D topological map consists in computing incrementally each member of the formula presented in Proposition 2. The number of connected components is constant, and since the number of cavities can be obtained incrementally using the method proposed in Sect. 4.3.1, we detail here the incremental computation of the Euler characteristic of the border of the union of r_1 and r_2, denoted $\chi'(r_1 \cup r_2)$.

In a 3D topological map, a border is a surface composed of faces, edges, and vertices. There may be one or several surfaces between two adjacent regions; these are the inner surfaces. We call inner cells the cells that entirely belong to the inner surfaces: that is cells such that the region of each dart representing the cell is either r_1 or r_2. Proposition 4 gives an incremental computation formula to obtain the Euler characteristic of the border of the union of two regions r_1 and r_2 from the Euler characteristic of the border of the two regions and the Euler characteristic of the inner surfaces between the two regions. Proof of Proposition 4 is available in [16].

Proposition 4. $\chi'(r_1 \cup r_2) = \chi'(r_1) + \chi'(r_2) - 2\chi(inner(r_1, r_2))$

Figure 13 presents an example of incremental computation of the number of tunnels for the union of regions $r_1 \cup r_2$ detailing the incremental computation of χ'. The resulting region has one tunnel and zero cavity.

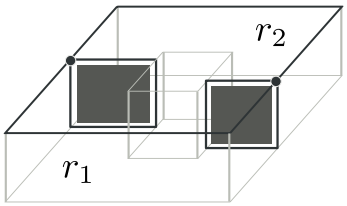


Fig. 13 Illustration of the incremental computation of the number of tunnels for the union of regions $r_1 \cup r_2$ in a 3D topological map. Edges drawn in gray show the shape of the region. The edges drawn in black are actual cells represented in the topological map. The faces drawn in dark gray are inner faces, $\chi(inner(r_1, r_2))$ is equal to two (two faces). The values of $\chi'(r_1)$ and $\chi'(r_2)$ are equal to two (two vertices, three edges, and three faces). The resulting value for $\chi'(r_1 \cup r_2)$ is equal to zero which corresponds to the expected value for a torus surface. The number of tunnels obtained after using the formula is one.

We can ignore cells that partially belong to inner surfaces: they form the border of the inner surfaces, and there is an equal number of edges and vertices. Since we are only interested in the Euler characteristic and not in the actual number of cells used to represent the union of regions r_1 and r_2 border, we can ignore these cells: the Euler characteristic of the border of inner surfaces is always zero.

The algorithm that computes incrementally $\chi'(r_1 \cup r_2)$ for the union of two regions r_1 and r_2 is defined as follow. The prerequisite for the incremental computation of χ' is to initialize the χ' value for each region in the first partition represented by a topological map. This is done either by counting cells and using the algorithm proposed in Sect. 4.2, or by using the incremental approach during the extraction of the topological map as presented in [13]. The χ' value is stored for each region.

The incremental algorithm is divided in two processing loops. In the first loop, we run through darts of the surface of r_1 that are in contact with r_2. Then each dart belonging to an inner face is marked accordingly. We also count the number of inner faces that are discovered. In the second loop, we count the number of the inner

vertices and the number of the inner edges. Using the fact that the darts belonging to inner faces are marked, we conclude that a cell is inner if all darts used to represent the cell are marked as inner. In the final step, we compute the Euler characteristic of the inner border. We retrieve the χ' value for both region r_1 and region r_2, and we use the formula from Proposition 4 to obtain the Euler characteristic of the border of the union of the two regions around the dart d. Then, with the incrementally computed value for the number of cavity, we use the formula from Proposition 2 to obtain the number of tunnels.

The algorithm has a linear time complexity with the number of darts belonging to $\langle \beta_1, \beta_2 \rangle(d)$ (a subset of the darts used to represent cells of the border of region r_1). In fact, each dart of the orbit is processed at most four times during the course of the algorithm. Darts belonging to the inner cells of $r_1 \cup r_2$ are also covered during the cell counting part of the algorithm, but their number is linearly proportional to the number of darts in the orbit.

4.4 Implementation of Topological Criteria in the Segmentation

The incremental computation of Betti numbers allows to anticipate the number of tunnels and cavities during a region merging process. Thus, we can use the number of cavities, or the number of tunnels as a removal criterion during the segmentation.

Weight functions can be defined according to topological properties, for instance, one can prioritize cells whose removal do not change the topology of the regions. However, due to region merging, a cell removal that does not change topological features such as the number of cavities in a simple two regions merging might do so after several other removals. It seems unlikely that such a weight function has any interest for the segmentation part, and thus we keep weight functions given in Sect. 3.3 based on intensity values of the image.

We propose to define two new criteria that handle topological properties like:

1. Forbidding topological changes for regions;
2. Reducing the number of cavities.

These criteria can be mixed with other criteria such as the ones presented in Sect. 3.3 to control some colorimetric properties of the resulting regions. In such a case, the idea is to check successively the different criteria and allowing the merge if all criteria return a true value.

The first criterion, known as constant topology criterion and given in Algo. 10, returns true if all the Betti numbers remain constant. That is, if each Betti number of the resulting region is equal to the sum of the same Betti numbers of the two initial regions (except for the number of connected components which is constant).

The second criterion, given in Algo. 11, returns true if the total number of cavities is reduced. That is if the number of cavities of the resulting region is smaller than the sum of the numbers of cavities of the initial regions.

Algorithm 10: Removal criterion based on constant Betti numbers

Input: A dart d of an nD topological map.
Result: `true` iff $c_{n-1}(d)$ must be removed.

$r_1 \leftarrow region(d)$; $r_2 \leftarrow region(\beta_n(d))$;
for $i \leftarrow 1$ *to* n **do**
  $b \leftarrow$ compute incrementally b_i for $r_1 \cup r_2$;
  if $b \neq b_i(r_1) + b_i(r_2)$ **then return** *false*;
  ;
return *true*;

Algorithm 11: Removal criterion based on decreasing number of cavities

Input: A dart d of a 3D topological map.
Result: `true` iff $c_2(d)$ must be removed.

$r_1 \leftarrow region(d)$; $r_2 \leftarrow region(\beta_3(d))$;
$C \leftarrow$ compute incrementally the number of cavities of $r_1 \cup r_2$;
return $C \leq \#cavities(r_1) + \#cavities(r_2)$;

Since we have a direct access to each information associated with darts ($region(b)$, $\beta_n(d)$) and to each Betti number ($b_i(r)$), the two algorithms have a time complexity equal to the time complexity of the incremental computation algorithms which are linearly proportional to the number of darts used to represent the cells of the border of the regions.

The incremental computation method implies that, if two regions r_1 and r_2 are merged during the symbolic merging, we update in constant time the Betti numbers. This is achieved by storing the values computed during the evaluation of the topological criteria and using them to update the values of the symbolic region $r_1 \cup r_2$ if the two regions are merged.

5 Experimental Results

We present some segmentation results obtained using the segmentation algorithm proposed in the previous section. First, we present some results obtained using different criteria with the generic segmentation algorithm and topological maps. Then, we introduce topological constraints inside the segmentation algorithm to obtain partitions that constraint the number of cavities or tunnels.

5.1 Generic Criteria

We show some segmentation results obtained using the different criteria presented Sect. 3.3. Figure 14 shows the segmentation of a classical 2D image using the intensity range criterion, and the segmentation of simulated body positron emission tomography (PET) 3D image using the contrast criterion. We also apply removal of small regions to obtain a cleaner partition. In Fig. 14(c), we process edges in a random order, that is without a meaningful weight function. The obtained partition has more regions and some regions, which have a similar intensity (like background regions) cannot be merged together due to the intensity range constraint.

(a) (b) (c) (d) (e)

Fig. 14 Sample partitions using the range and contrast criteria. (a) Original "fishing boat" image; (b) Partition of the "fishing boat" image using the range criterion (range=100); (c) Partition of the "fishing boat" image using the range criterion with a random weight function (range=100); (d) Slice of a simulated body PET scan (luminosity and contrast have been edited); (e) Slice of the simulated PET 3D image segmented using the contrast criterion with the removal of small regions (k=2500, minimum size for regions 500 voxels).

The small region criterion allows removing small regions that do not appear to be relevant to the obtained partition. An example of the use of such a criterion is show in Fig. 15 where the initial partition Fig. 15(b), obtained using the range criterion, contains lots of small regions. Using the small region removal criterion in conjunction with a weight function that orders the edges by the intensity difference between the two adjacent regions, we obtain the partition presented Fig. 15(c) where the regions are larger.

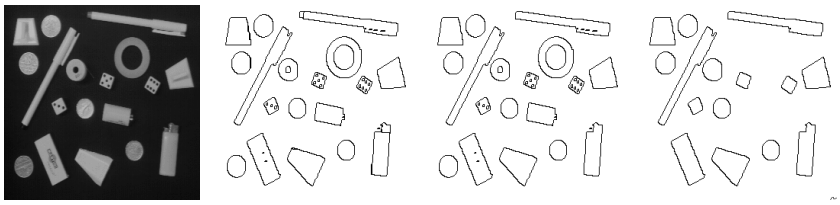
5.2 Constraint on Betti Numbers

In Fig. 16, we show the impact of topological constraints applied to a segmentation of the 2D image presented Fig. 16(a). We use the range criterion to obtain an initial segmentation shown in Fig. 16(b). In this segmentation there are regions with cavities. Starting with the initial partition, increasing the range threshold, and constraining the number of cavities by a constant, we obtain the segmentation proposed in Fig. 16(c). If the cavity count is allowed to decrease from the cavity count in the initial partition, the result is slightly different, as shown in Fig. 16(d). Some objects disappeared as the regions representing them are merged with the background.

(a) (b) (c)

Fig. 15 Partitions obtained from a 2D image using the small region criterion. (a) Original image; (b) Partition obtained using the range criterion with a threshold of 50; there are many small regions; (c) Partition obtained from the previous partition, by using the segmentation algorithm with a the small region threshold set to 25. There are no more small regions of size less than 25 in the partition.

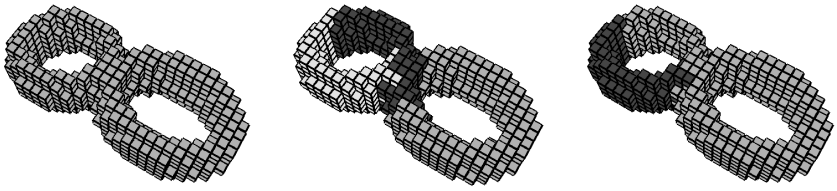
Moreover, some small cavities, like dots in the dice, are removed while they are preserved if the number of cavities remains constant.



(a) (b) (c) (d)

Fig. 16 Segmentation of a 2D image using a topological constraint to limit the number of cavities. (a) Original image; (b) Initial partition obtained with a range criterion (threshold set to 100); (c) Partition obtained from (b) if the number of cavities is constant with a range threshold set to 150; (d) Partition obtained from (b) if the number of cavities decreases with a range threshold set to 150.

Figure 17 illustrates the results obtained by a segmentation algorithm that controls the number of tunnels in objects. Without topological constraint, the segmentation using the intensity range criteria produces a 2-torus (a region with two tunnels) as seen in Fig. 17(a). If no tunnels are allowed, the segmentation result uses three regions to represent the same object, none of them having a tunnel (Fig. 17(b)). Finally, if one tunnel per region is allowed, the segmentation result uses two regions to represent the object: one of them has a tunnel and the other one has none. This is due to the ordering of the border of the segmentation algorithm that merges the regions (Fig. 17(c)).



(a) (b) (c)

Fig. 17 Segmentation of a 3D object using a topological constraint to limit the number of tunnels. (a) Original partition of the object without a topological constraint on the number of tunnels per region, the region has two tunnels; (b) Partition obtained if the number of tunnels is minimized: the same object is represented by three regions without a tunnel; (c) Partition obtained if the allowed number of tunnels does not excess one: the same object is represented by two regions, one having a tunnel while the other has none.

6 Open Problems and Discussion

In this chapter, we presented an efficient and generic image segmentation algorithm based on topological maps. The algorithm is generic because it is defined in any dimension, and it is controlled by two functions that change its behavior. In order to specify the segmentation process, we define several segmentation criteria using different kind of information associated with the regions or the cells represented by topological maps. A method of computation of topological invariants, the Betti numbers, based on the complete representation of cells and the relationship between them in topological maps, is presented. We introduce new topological criteria using Betti numbers that control the number of cavities and tunnels in regions of an image partition. This process is an example of topological control during image segmentation. We provide some experiments of 2D and 3D image segmentation using different criteria to enlighten the use of topological maps for image segmentation.

There are several open problems and possible extensions for image segmentation using topological maps. From a theoretical point of view, the main issue to tackle involves the definition of topological maps in any dimension. The problem is to construct a minimal combinatorial map that describes an nD image partition into regions. In 2D and 3D, we propose a constructive definition that is proved in [9, 8]. Starting from a combinatorial map describing all the elements of the image, the model is refined by successive simplifications using removal operations. The process produces the minimal map representing the partition of the image. The principal difficulty is to guarantee the preservation of all topological information during the cell removals while obtaining the minimal number of cells in the combinatorial map. One can directly extend the constructive definition in higher dimensions since combinatorial maps and removal operations are defined in any dimension. The constraints required to preserve all the topological information may also be stated for any dimension, but guaranteeing that the obtained combinatorial map is minimal, remains an open problem. This is not an issue for the particular topic of image seg-

mentation since the generic algorithm does also work in non-minimal combinatorial maps. However, this is not satisfying since two different combinatorial maps may describe the same partition.

Another theoretical problem is the computation of Betti numbers and their use as criteria for image segmentation in any dimension. Currently, we only have computation algorithms in case of 2D and 3D topological maps. This issue is related to the problem of combinatorial balls' characterization. Actually, Betti numbers are defined for a cellular complex where each cell is homeomorphic to a topological ball. As there is no combinatorial way to recognize if a cell is a topological ball, there is no easy way to compute Betti numbers. This is also an open problem in computational topology. In 2D and 3D, we solve this problem by using the Euler characteristic of each border of regions and the implicit cells, but this solution is not available in higher dimensions. An extension of this problem is the computation of other topological invariants such as the homology group generators for regions in a topological map.

Future works that do not raise theoretical questions might include the development of alternative segmentation methods using topological maps. In this work, we presented a bottom-up segmentation approach that uses the removal operation to group regions and obtain the final segmentation. Using the split operation, which divides a region in smaller regions, one can develop a top-down approach. Starting with the whole image in a unique region, the final partition is obtained by successive splits of regions. New criteria based on region properties should be developed to support the top-down approach as the method raises new issues. One aspect of such criteria is to define how regions are split from the geometrical and topological point of views. An example of topological criterion we envision is the split of a region based on the number of tunnels or cavities. An initial idea is to split a region to remove a tunnel or a cavity from that region: this criteria might gain from the computation of the homology group generators to guide the split operation. Another idea is to split a region with multiple cavities such as each of the resulting regions has exactly one cavity.

To broaden the use of topological maps in computer vision related works, we also want to tackle real world image segmentation issues. In preliminary experiments, we obtained interesting results regarding the use of Betti numbers as criteria for image segmentation. Using image segmentation with topological maps, in conjunction with topological criteria like constraints on Betti numbers, could ease the development of image segmentation tools. This should be fully demonstrated by applying image segmentation with topological maps to real use cases and compare results to existing approaches. Experts from the field should be brought along to define the goals of the segmentation tool and evaluate the results. New criteria, based on different parameters should also be introduced and mixed with the existing ones to produce better results.

References

1. Y. Bertrand, G. Damiand, and C. Fiorio. Topological map: Minimal encoding of 3d segmented images. In *Proc. of 3rd Workshop on Graph-Based Representation in Pattern Recognition*, pages 64–73, Ischia, Italy, May 2001.
2. J.P. Braquelaire and L. Brun. Image segmentation with topological maps and inter-pixel representation. *Journal of Visual Communication and Image Representation*, 9(1):62–79, march 1998.
3. J.P. Braquelaire, P. Desbarats, and J.P. Domenger. 3d split and merge with 3-maps. In *Proc. of International Workshop on Graph based representations*, pages 32–43, Ischia, Italy, may 2001. IAPR-TC15.
4. J.P. Braquelaire, P. Desbarats, J.P. Domenger, and C.A. Wüthrich. A topological structuring for aggregates of 3d discrete objects. In *Proc. of International Workshop on Graph based representations*, pages 193–202, Austria, may 1999. IAPR-TC15.
5. J.P. Braquelaire and J.P. Domenger. Representation of segmented images with discrete geometric maps. *Image and Vision Computing*, 17(10):715–735, 1999.
6. J.P. Braquelaire and P. Guitton. A model for image structuration. In *Proceedings of Computer Graphics International*, pages 426–435, 1988.
7. L. Brun. *Segmentation d'images couleur à base topologique*. Thèse de doctorat, Université Bordeaux 1, décembre 1996.
8. G. Damiand. Topological model for 3d image representation: Definition and incremental extraction algorithm. *Computer Vision and Image Understanding*, 109(3):260–289, March 2008.
9. 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.
10. G. Damiand, M. Dexet-Guiard, P. Lienhardt, and E. Andres. Removal and contraction operations to define combinatorial pyramids: Application to the design of a spatial modeler. *Image and Vision Computing*, 23(2):259–269, February 2005.
11. G. Damiand and P. Lienhardt. Removal and contraction for n-dimensional generalized maps. In *Proc. of International Conference on Discrete Geometry for Computer Imagery*, volume 2886 of *Lecture Notes in Computer Science*, pages 408–419, Naples, Italy, November 2003. Springer Berlin/Heidelberg.
12. G. Damiand and P. Resch. Topological map based algorithms for 3d image segmentation. In *Proc. of 10th International Conference on Discrete Geometry for Computer Imagery*, volume 2301 of *Lecture Notes in Computer Science*, pages 220–231, Bordeaux, France, April 2002. Springer Berlin/Heidelberg.
13. Guillaume Damiand, Samuel Peltier, Laurent Fuchs, and Pascal Lienhardt. Topological map: An efficient tool to compute incrementally topological features on 3d images. In Ralf Reulke, Ulrich Eckardt, Boris Flach, Uwe Knauer, and Konrad Polthier, editors, *IWCIA*, volume 4040 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.
14. J.P. Domenger. *Conception et implémentation du noyau graphique d'un environnement* $2D1/2$ *d'édition d'images discrètes*. Thèse de doctorat, Université Bordeaux 1, avril 1992.
15. A. Dupas and G. Damiand. First results for 3d image segmentation with topological map. In *Proc. of International Conference on Discrete Geometry for Computer Imagery*, volume 4992 of *Lecture Notes in Computer Science*, pages 507–518, Lyon, France, April 2008. Springer Berlin/Heidelberg.
16. A. Dupas and G. Damiand. Region merging with topological control. *Discrete Applied Mathematics*, 157(16):3435–3446, August 2009.
17. P. F. Felzenszwalb and D. P. Huttenlocher. Image segmentation using local variation. In *Proc. of Computer Vision and Pattern Recognition*, pages 98–104, June 1998.
18. P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

19. C. Fiorio. A topologically consistent representation for image analysis: the frontiers topological graph. In *Proc. of International Conference on Discrete Geometry for Computer Imagery*, volume 1176 of *Lecture Notes in Computer Science*, pages 151–162, Lyon, France, november 1996.
20. S. Fourey and L. Brun. A first step toward combinatorial pyramids in n-D spaces. In *Proc. of Graph-based Representations in Pattern Recognition*, volume 5534 of *Lecture Notes in Computer Sciences*, pages 304–313, Venice, Italy, May 2009. Springer.
21. S. Fourey and L. Brun. Efficient encoding of n-d combinatorial pyramids. In *Proc. of International Conference on Pattern Recognition*, ICPR '10, pages 1036–1039. IEEE Computer Society, 2010.
22. J. Freixenet, X. Muoz, D. Raba, J. Mart, and X. Cuf. Yet another survey on image segmentation: Region and boundary information integration. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2352 of *Lecture Notes in Computer Science*, pages 21–25. Springer Berlin / Heidelberg, 2002.
23. K.S. Fu and J.K. Mui. A survey on image segmentation. *Pattern Recognition*, 13(1):3–16, 1981.
24. Fraunhofer Itwm. Survey of 3d image segmentation methods. *Fraunhofer Institut Technound Wirtschaftsmathematik ITWM*, 123(123):Kaiserslautern, Germany, 2007.
25. E. Khalimsky, R. Kopperman, and P.R. Meyer. Computer graphics and connected topologies on finite ordered sets. *Topology and its Applications*, 36:1–17, 1990.
26. U. Köthe. Xpmaps and topological segmentation - a unified approach to finite topologies in the plane. In *Proc. of International Conference Discrete Geometry for Computer Imagery*, volume 2301 of *Lecture Notes in Computer Science*, pages 22–33, Bordeaux, France, april 2002.
27. V.A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 46:141–161, 1989.
28. W.G. Kropatsch and H. Macho. Finding the structure of connected components using dual irregular pyramids. In *Proc. of International Conference on Discrete Geometry for Computer Imagery*, pages 147–158, september 1995.
29. P. Lienhardt. Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Commputer Aided Design*, 23(1):59–82, 1991.
30. P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324, 1994.
31. James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, 1984.
32. Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.
33. A. Rosenfeld. Adjacency in digital pictures. *Information and Control*, 26(1):24–33, 1974.
34. R. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22(2):215–225, 1975.
35. D. Willersinn and W.G. Kropatsch. Dual graph contraction for irregular pyramids. In *Proceedings of 12th IAPR International Conference on Signal Processing*, volume 3, pages 251–256, Jerusalem, Israel, 1994.
36. Y.J. Zhang. Evaluation and comparison of different segmentation algorithms. *Pattern Recognition Letters*, 18(10):963–974, 1997.