3D Image Topological Structuring with an Oriented Boundary Graph for Split and Merge Segmentation

Fabien Baldacci, Achille Braquelaire, Pascal Desbarats, and Jean-Philippe Domenger

LaBRI, Université Bordeaux 1, 351 cours de la liberation F-33405 Talence cedex France
{baldacci,braquelaire,desbarats,domenger}@labri.fr

Abstract. In this paper, we present a new representation model for the topology and the geometry of a 3D segmented image. This model has been designed to provide main features and operations required by a 3D image segmentation library. It is mainly devoted to region based segmentation methods such as split and merge algorithms but is also convenient for contour based approaches. The model has been fully implemented and tested both on synthetic and real 3D images.

Keywords: 3D Images Segmentation. Topological graph. Image representation.

1 Introduction

Split and merge segmentation [1] basically consists in building and refining a partition of an image. The partition elements are called regions. The split operation consists in dividing a domain into regions, and the merge operation consists in fusing two or more adjacent regions. Split and merge algorithms consist in alternatively splitting and merging regions according to one or more criteria. Those methods have been extensively studied for 2D images and can be transposed to 3D images. Related algorithms require the extraction of some features to drive the segmentation process, such as the domain corresponding to a region (to compute geometrical features, region mean, region variance, etc.), the set of regions being adjacent to another one, or the set of regions included into another one. Boundary of a given region or boundary shared by two regions are also usually needed.

Efficiency of features extraction is a critical aspect of the development of 3D image segmentation methods. Feature extraction can be improved by structuring the segmented image. This structuring must represent both the topology and the geometry of the partition of a 3D image into 3D regions [2]. Several topological models have been proposed in 2D. One popular model is the model of *Region Adjacency Graph* (RAG) [3] which is simple to implement but does not capture the whole topology of the segmented image, such as the multiple adjacency of regions relation (related to the number of disconnected surfaces

D. Coeurjolly et al. (Eds.): DGCI 2008, LNCS 4992, pp. 541–552, 2008.

shared by two regions). Other popular and more sophisticated models are the models based on combinatorial maps [4,5]. Combinatorial maps can be defined in 3D [6] and two models using 3D combinatorial maps have been proposed: the *Hierarchical Local Embedding* (HLE) [7,8] and the *Geometrical Embedding* (GE) [9]. Both these models use 3D combinatorial maps associated with inclusion relations or inclusion tree, and an intervoxel representation of the geometry of regions boundary [10,11]. Intervoxel representation of geometry [12] lays on the cellular decomposition of 3D discrete space [13,14] in which elements are voxels (elements of dimension 3), surfels (intersection of two voxels), linels (intersection of two surfels) and pointels (intersection of two linels) (see Fig. 1).

Nevertheless, the development of these models for 3D images raises some unstraightforward problems. For instance, a GE representation is not minimal in term of topological elements, and some extra elements need to be added with some voxel configurations. A HLE representation uses fictive topological elements that cannot be retrieved in the geometrical level (this leads to some constraints on the update operations of the representation), and local geometry representation which may be inefficient for split and merge methods. These models have been compared in order to be merged [15], but the resulting model still uses fictive elements and related drawbacks remains the same. Furthermore, combinatorial maps require to decompose region boundary into surface elements homeomorphic to a topological disc, which need extra processing on surfaces resulting from a segmentation.

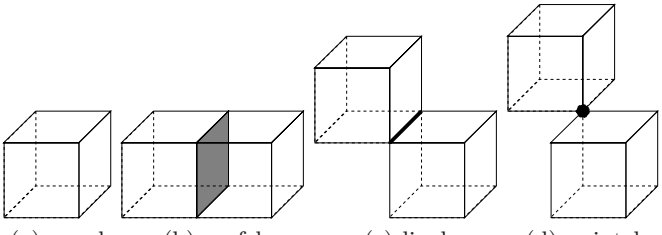


Fig. 1. Elements of the cellular decomposition of 3D discrete space

Models based on maps encode all topological features of a segmented image like regions neighborhood, regions inclusion, and higher level topological features such as Euler characteristic — and thus the number of regions' holes — and Betti numbers [16]. However many segmentation algorithms do not require such high level topological features and it is relevant to consider alternative topological models, less powerful than maps but powerful enough to implement most of split and merge methods. Thus we propose in this work a new model for structuring 3D segmented image which is lighter than models based on combinatorial maps, compatible with them (it is still possible to extract the combinatorial map of some regions of interest) and which avoids their drawbacks (the compromise is

that high level topological features require specific processing to be extracted from our model).

To sum up, the goal of this work is to develop a model which is on one hand not space consuming and more suitable than other models when considering large 3D images, and on the other hand which allows efficient implementation of 3D split and merge algorithms. We retained the following approach:

1. Identify the main geometrical and topological features and the representation updates required by region based segmentation algorithms.
2. Define a minimal set of functions satisfying these requirements (this set of functions can be seen as the API of a segmentation library).
3. Build a model with a minimum memory cost that allows efficient computation of these functions.

Those requirements and functions are listed in Section 2. The model is presented in Section 3, the related functions are detailed in Section 4. Finally examples and results are given in Section 5.

2 Requirements for a Segmentation Environment

A partition P of an image is a region set such as: $P = \{\{r_1...r_n\} \mid \cap_{i \neq j} r_i r_j = \emptyset, \cup_{1 \leq i \leq n} r_i = P\}$; a region is a set of 6-connected voxels. A segmented image is a partition of the image satisfying criteria.

The boundary of a region r_i is the set of surfels that separates r_i from other regions, it is denoted by ∂r_i. A region boundary consists of the surfels of the outside boundary (the shell of the region) and of the surfels of the region cavities. The common boundary of two adjacent regions r_i and r_j is decomposed into a set of connected surfaces $\{S_{ij}^1...S_{ij}^n\}$, such as each connected surface S_{ij}^k is the *k-nth* maximum set of connected surfels $s \in \partial r_i \cap \partial r_j$, called *S-set*. These surfaces are not necessarily simple (a simple surface is a surfce whithout hole). need any other processing for our representation.

Update primitives required by region based segmentation methods are basically the construction of the representation of a partitioned region (resulting from the splitting step of a split and merge method) and the merging of two or more adjacent regions. The first one requires the specification of a region partition. We chose to specify a region partition by using a function $Oracle(v_i, v_j)$ which indicates whether two adjacent voxels v_i and v_j belong or not to the same region. This function provides an abstract representation of a region (or an image) partition which is not dependent on the concrete representation of the segmented image (label array, boundary representation, etc.).

Let us now give a short description of the functions we have retained for a 3D image segmentation environment. The list of these functions is derived from an analogous work for 2D images [17]. The reader can refer to this work to find justifications of the choice of the selected function.

Updating functions. The two following functions respectively build the representation of a partitioned region and merge two adjacent regions.

Split$(r, Oracle)$**:** returns $\{r_i | \cup_i r_i = r\}$. This function refines the partition of the image by dividing r_i, according to the *Oracle* function which specifies the partition.

Merge(r_i, r_j)**:** returns $\{r | r = r_i \cup r_j\}$. This function merges two adjacent regions r_i and r_j.

Geometrical functions. The following functions extract features from the geometrical representation.

RegionDomain(r_i)**:** returns $\{v | v \in r_i\}$. This function returns a list of pairs of voxels defining a set of lines covering the region r_i: for each pair $\{v_i(x_i, y, z), v_j(x_j, y, z)\}$, voxels $\{v_k(x_k, y, z) | x_i < x_k < x_j\} \in r_i$.

RegionBoundary(r_i)**:** returns $\{s | s \in \partial r_i\}$. This function returns a list of closed surfaces. The number of returned surfaces is equal to the number of cavities of the region, plus one for the outside boundary.

CommonBoundary(r_i, r_j)**:** returns $\{s | s \in \partial r_i \cap \partial r_j\}$. This function returns a set of non simple surfaces $\{S_1 ... S_n\}$, such as $\forall s \in S_i, s' \in S_j, i \neq j, \forall surfel\ chain \{s ... s'\}$, $\exists\, s'' \in \{s ... s'\}$, $s'' \notin \partial r_i \cap \partial r_j$.

RegionAt(voxel v): returns $r | v \in r$.

Topological functions. The following functions extract topological features: neighborhood and inclusion relations. A region r_i is said to be included in another region r_j if each possible voxel chain (sequence of connected voxels) from any voxel of r_i to the outside of the image contains at least one voxel of r_j. The nearest region in which r_i is included is denoted by $Parent(r_i)$.

IncludedRegions(r_i)**:** returns $\{r | r_i = Parent(r)\}$.
Neighbors(r_i)**:** returns $\{r_j | \partial r_i \cap \partial r_j \neq \emptyset\}$.
OppositeRegion(Surface S_{ij}^k**, Region** r_i**):** returns r_j.
Parent(r_i)**:** returns $r | r = Parent(r_i)$.

This list of *required functions* can be seen as the basic API of a 3D image segmentation library. In the following section we describe the model we retained as a framework to implement these functions.

3 The Model

The model described in this section is based on the *S-sets* induced by a partition. It is made of two representation levels, a topological one and a geometrical one. These levels are linked together by associating an oriented surfel with each *S-set*. The geometrical level must at least encode the boundary surfels in order to be able to represent *S-sets*. It must also encode the edges of *S-set*, denoted by ∂S, which are sequences of linels. The encoding of such linels is necessary to traverse the boundary elements composing surface boundaries. We chose to use a boundary image [15] (also called inter-voxel matrix) to encode the geometrical elements. This matrix encodes the presence of surfels, linels and pointels corresponding to the image with seven bits per voxel (pointels are not used by our

model, but are necessary to the embedding of a combinatorial map). Another advantage of the boundary image is to provide a global representation of the geometry which allows the access to the topological representation from a geometrical element. Moreover such models are more convenient for implementing contour based methods.

From the required functions list of the previous section we know we have to provide an explicit encoding of two topological features: the region adjacency relation and the inclusion relation. Coding region adjacency is achieved by using an adjacency graph that we call *oriented boundary graph* (in short *OBG*). This adjacency graph, which is a multigraph, needs one edge per *S-set* to encode multiple adjacency. More precisely a graph $OBG(E, V)$ is defined on:

- a set of vertices $V = \{r_1, ...r_n\}$;
- a set of edges $E = \{e(r_i, r_j)^k | \exists S_{ij}^k$ with $\cup_k S_{ij}^k = \partial r_i \cap \partial r_j\}$.

This graph provides the **Neighbors** function simply by traversing all the incident edges of the given region. Since the *OBG* is a connected graph, it is possible to compute the inclusion relation without adding an extra structure such as an inclusion tree. The algorithm 1 is used to compute the **IncludedRegion** function. It returns the list of regions included in a given one. In this algorithm, the $ParentSurface$ is a *S-set* belonging to the outside boundary of the region. Then all *S-sets* of the outside boundary have to be marked, then unmarked *S-sets* correspond to region cavities. This algorithm needs to encode the *S-set* adjacency relation. This relation explicits a relation implicitly encodede in the *OBG* definition, and is encoded by linking each edge $e(r_i, r_j)^p$ with $e(r_l, r_m)^q$ if $\partial S_{ij}^p \cap \partial S_{lm}^q \neq \emptyset$. This link is done by a representative linel pointing to all edges corresponding to *S-set* adjacent to it. Note that if the inclusion relation has to be computed many times, it is possible to perform a lazy evaluation by storing the ParentSurface in order to avoid the breadth-first search in the *OBG*. The orientation of the *OBG* will be precised in the next paragraph.

The boundary image and the *OBG* allows to efficiently retrieve all features needed by the required functions list. Both structures now have to be linked

Algorithm 1. Inclusion

Require: An *OBG* and a region r_i
Returns: List of regions included in r_i

```
ParentSurface ← BreadthFirstSearchUntil(OBG, r_i)
Mark(ParentSurface)
for all Surface ∈ ∂r_i and connected to ParentSurface do
    Mark(Surface)
end for
for all Surface ∈ ∂r_i do
    if IsUnmarked(Surface) then
        AddToList(ResultList, OppositeRegion(Surface, r_i))
    end if
end for
```

together in order to provide efficient updates of the representation (**Split** and **Merge** functions). It is done by selecting a representative surfel for each *S-set*. This surfel is associated in the geometrical level with its corresponding edge in the topological level. It is thus possible to go from an element of the boundary image to the associated element of the *OBG*. Representative surfels are also stored in the *OBG* such that with each edge $e(r_i, r_j)^k$ is associated one surfel of its corresponding *S-set* S_{ij}^k. This allows to retrieve all surfaces from the *OBG*. Finding regions is done by orienting edges. Edges are oriented depending on their geometrical embedding. For each type of surfel s it is possible to define a positive voxel $Positive(s)$ and a negative one $Negative(s)$ according to the image orthonormal coordinates system. Those voxels belong to the two different regions the surface is separating. Edges are oriented such that with each edge $e(r_i, r_j)$ a surfel $s|Positive(s) \in r_i, Negative(s) \in r_j$ is associated. This orientation is not used for traversing the *OBG*.

An example is shown on Fig. 2. The region r_0 is the outside of the image, called the infinite region. There are four other regions, such as r_2, r_3 and r_4 are included in r_1. Regions are linked in the *OBG* by edges representing *S-sets*. Region r_1 is separated from region r_0 by one *S-set* which is the outside boundary of r_1. Region r_1 has two cavities: the connected component made of r_2 and r_3, and the one composed of r_4. The outside boundary of r_2 (as the outside boundary of r_3) is cut into two *S-set*: one common with r_1 and the other one common with r_3. Each edge is linked to the representative surfel of its corresponding *S-set*, and adjacent surfaces are linked together by a linel. Black side of a representative surfel s corresponds to their $Negative(s)$, and grey side corresponds to their $Positive(s)$.

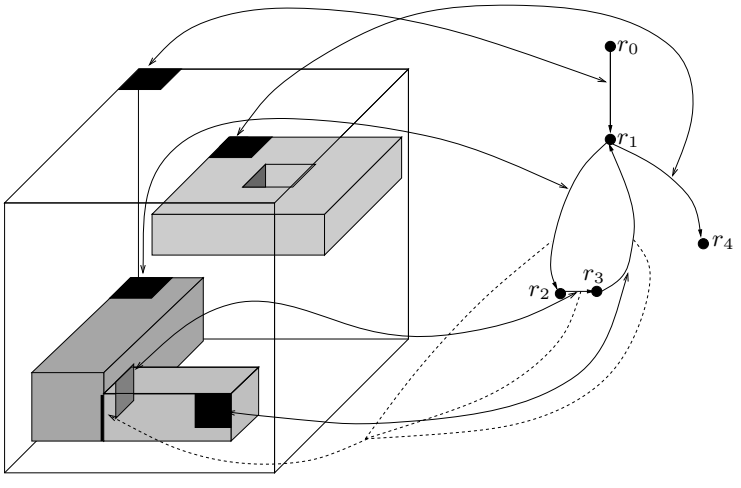


Fig. 2. Example of image with the corresponding representation with our model

4 Updating and Feature Extraction Functions

The construction of the representation of a partitioned region is done from the region domain and the *Oracle* function representing the segmentation. The region may be the whole image, for instance, at the first step of a split and merge process. The construction is based on a temporary labelling of each elements: voxels, surfels and linels, such as all the voxels (resp. surfels) of a region (resp. a *S-set*) have the same label, and two different regions (resp *S-set*) have different labels. Each voxel label is associated with one region in the *OBG*, and each surfel label is associated with one edge. Since traversing is done by a scan-line algorithm, a region can temporarily have different labels. When such a case is detected, the two related vertices of the *OBG* are merged and an indirection table is updated in order not to relabelling voxels.

The split function is described in Algorithm 2. Each voxel of the domain is compared with its labelled neighbors, and then is labelled with possibly the appropriate treatment (creating a node or merging two ones). The way the domain is traversed (by scan-line) allows to determine which neighbors are already labelled without any test. In the same time surfels are detected. When all the voxels of a 2x2x1 cuboid (Figure 3) are labelled, all the surfels contained in the cuboid have been detected and it is possible to decide whether a linel is present. Surfels and linels have to be labelled in the same way voxels are.

Construction and split is done by traversing all the voxels of the image once, and since there is no constraint on the configuration of treated voxels for treating a new one, it is possible to parallelize this operation [18].

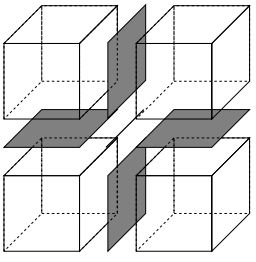


Fig. 3. A 2x2x1 voxel cuboid and its contained surfels and linel

Merging two regions r_i and r_j is done by deleting all the *S-sets* S_{ij}^k from the boundary image and their corresponding edges $e(r_i, r_j)^k$ from the *OBG*. Some *S-sets* may have to be merged. Such *S-sets* are detected by checking all the encountered linels: if they bound only two *S-sets*, then those *S-sets* have to be merged (*S-sets* to be merged are retrieved directly from the representative linel). The merging of two regions is described by Algorithm 3.

CommonBoundary of two regions can be retrieved since each edge is pointing to a representative surfel from which the *S-set* can be traversed by looking

Algorithm 2. Split Algorithm

Require: An OBG, A domain D and an $Oracle$ function
for all Voxel $v \in D$ **do**
  for all labelled voxel v_i neighbors of v **do**
    if $Oracle(v_i, v)$ **then**
      SaveLabel(list, v_i)
    else
      addSurfelBetween(v_i, v)
    end if
  end for
  if NumberOfLabel(list) > 1 **then**
    MergeRegion(list)
  else
    if NumberOfLabel(list) $==$ 0 **then**
      CreateNewRegion(NewLabel)
    end if
  end if
  for all 4 labelled voxel cuboid containing v **do**
    if more than 2 surfels are present **then**
      AddLinel(x, y, z)
    end if
    LabelAddedLinels()
    LabelSurfels()
  end for
end for

Algorithm 3. Merge Algorithm

Require: 2 regions r_1 and r_2 to merge
Returns: The new region $r = r_1 \cup r_2$
for each surface of r_1 **do**
  if it separates r_1 from r_2 **then**
    linellist $\leftarrow$ deleteSurfaceFromGeometry()
    DeleteSurfaceFromTopology()
  end if
end for
for each linel of linellist **do**
  if its degree is 2 **then**
    DeleteLinelFromGeometry()
    if it is a representative linel **then**
      MergeSurfaces(linel)
    end if
  else
    if its degree is less than 2 **then**
      DeleteLinelFromGeometry()
    end if
  end if
end for

for connected surfels up to encountering linels. Adjacent *S-sets* can be traversed together (to compute the **RegionBoundary**) without stopping on linels by using the surfel orientation in order to stay in a same region during the traversing. **RegionDomain** can be built from the boundary by sorting each encountered surfels of same type, each surfel pair of the results bounds a voxel line of the region.

Some regions features are additive for the split and the merge operation: size (in number of voxels), sum of values, sum of squares values, etc. It is possible to save those values with each region to compute features extraction such as mean and variance of regions. Furthermore, the model allows the use of active contours, useful for example to smooth the surface of a region.

5 An Example of Segmentation Algorithm

In this section we present an example of segmentation algorithm (Algorithm 4) using functions of the proposed framework. Note that this algorithm is not supposed to give accurate results but to show how the proposed model can be used to specify and implement a split and merge algorithm based on topological and geometrical features.

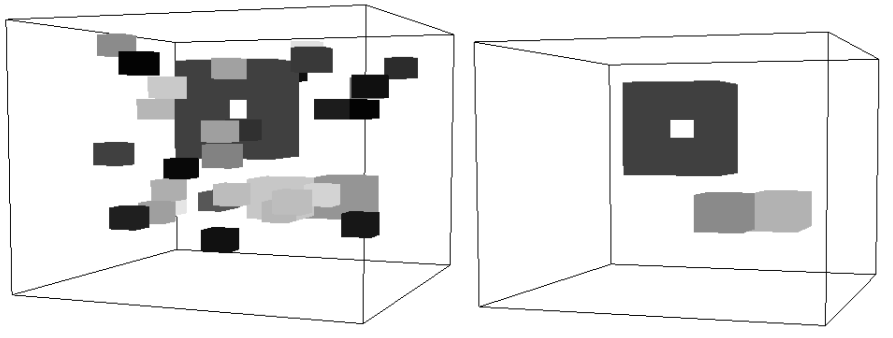
This algorithm first creates the partition of an image by classifying the voxels into classes according to given thresholds (on the examples we use five thresholds

Algorithm 4. Simple Segmentation Algorithm

Require: An image I, a size minsize and a value minmean
Split(I, Classify($\{s_i\}$))
for all region r **do**
  if Size(r) < minsize **then**
    if Parent(r) ∈ **Neighbors**(r) **then**
      Merge(r, **Parent**(r)
    end if
  end if
end for
for all region r **do**
  if Parent(r) ∈ **Neighbors**(r) **then**
    if abs(Mean(Parent(r)) - Mean(r)) < minmean **then**
      Merge(r, Parent(r))
    end if
  end if
end for
for all region r_i **do**
  for all region r_j ∈ **Neighbors**(r_i) **do**
    if abs(Mean(r_i) - Mean(r_j)) < minmean **then**
      Merge(r_i, r_j)
    end if
  end for
end for

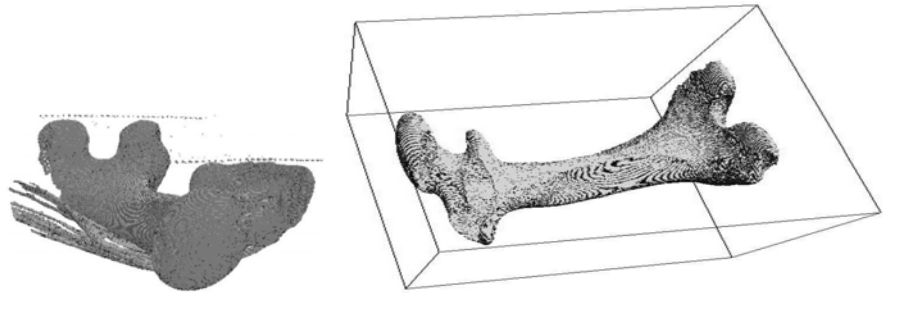
and six classes). Then it uses three criteria for merging the regions: first it merges small isolated regions with their parent (this corresponds to noise removing). Then it applies two kind of merges on adjacent regions that are similar enough according to segmentation criteria (mean difference, inclusion relation).

We have tested our representation both on synthetic and real images with the segmentation algorithm 4. Figure 4 shows the obtained result on synthetic noisy image. Figure 4(a) represent the obtained result after the split operation, and Figure 4(b) the obtained result at the end of the algorithm.



(a) (b)

Fig. 4. Example of segmentation on a noised synthetic image



(a) (b)

Fig. 5. Example of segmentation on a real medical image of size 512x512x475 voxels

Figure 5 shows the result obtained on a real medical image (representing a femur) sizing 512x512x475 voxels. We obtain from the split operation more than 200000 regions. Figure 5(a) is a capture during the segmentation process and it contains a lot of insignificant regions. Merges operations lead to the Figure 5(b) which is composed of 8 regions (outside of the image, the bone, the background and some regions inside the bone). This example shows the possibility of using our model to represent and update a partition with large size images.

6 Conclusion

The proposed model has been defined according to a set of topological and geometrical functions required by a segmentation environment. On one hand it is simpler than models based on combinatorial maps, and on the other hand it is convenient for implementing a library for 3D image segmentation. This model does not explicitly encode the whole topology of a segmented image as combinatorial maps do but it avoids the drawbacks of map based models. We chose to encode less information, in order to have a lighter and a more efficient model for all segmentation applications which do not use high level topological features of the regions. Nevertheless we think that both this model and map based models are interesting for image segmentation, depending on the kind of segmentation methods to implement. Thus the model has been designed in order to allow local extraction of a combinatorial map when it appears necessary to explicit the encoding of higher topological features. Furthermore it could be possible to use contour based methods in order to refine the result of a region based segmentation.

The model have been implemented and is fully functional with real images. The first version was a prototype implemented in Python language in order to experimentally validate the model, that is the reason why time an memory consumption cannot be given. A second optimised version in C language is currently under development. This development is achieved for the construction of the model, and for a 512x512x475 image with more than 1 million regions, it takes about 30 seconds to build the model on a 2,33Ghz intel Xeon machine.

Future works consist in studying the extraction of combinatorial maps from our model, in order to use high level topological features on some regions, and then to update our model after treatments. A fast surface traversing algorithm optimising the representative surfel search may be developed using two ideas: placing the representative surfels with priority to some directions, or saving a distance map with the boundary image. A further work will consist in parallelising most of the operations (especially the split one, for which a first result is presented in [18]).

References

1. Horowitz, S., Pavlidis, T.: Picture segmentation by a directed split and merge procedure. In: ICPR 1974, pp. 424–433 (1974)
2. Braquelaire, J.P., Brun, L.: Image segmentation with topological maps and interpixel representation. Journal of Visual Communication and Image Representation 9(1), 62–79 (1998)
3. Rosenfeld, A.: Adjacency in digital pictures. InfoControl 26 (1974)
4. Braquelaire, J.-P., Domenger, J.P.: Representation of segmented images with discrete geometric maps. Image Vision Comput. 17(10), 715–735 (1999)
5. Braquelaire, A., Desbarats, P., Domenger, J.P.: 3d split and merge with 3-maps. In: 3rd IAPR-TC-15 Workshop on Graph-based representation. CUEN, pp. 32–43 (2001) ISBN 887146579-2

6. Lienhardt, P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. Comput. Aided Des. 23(1), 59–82 (1991)
7. Bertrand, Y., Damiand, G., Fiorio, C.: Topological encoding of 3d segmented images. In: Nyström, I., Sanniti di Baja, G., Borgefors, G. (eds.) DGCI 2000. LNCS, vol. 1953, pp. 311–324. Springer, Heidelberg (2000)
8. Damiand, G.: Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d. PhD thesis, PhD Thesis, Montpellier II University (2001)
9. Desbarats, P.: Structuration d'images segmentées 3D discrètes. PhD thesis, PhD Thesis, Bordeaux I University (2001)
10. Braquelaire, A., Desbarats, P., Domenger, J.P., Wütrich, C.: A topological structuring for aggregates of 3d discrete objects. In: 2nd IAPR-TC-15 Workshop on Graph-based representation, pp. 193–202. OCG (1999) ISBN 3-8580-126-2
11. Damiand, G., Resch, P.: Topological map based algorithms for 3d image segmentation. In: Braquelaire, A., Lachaud, J.-O., Vialard, A. (eds.) DGCI 2002. LNCS, vol. 2301, pp. 220–231. Springer, Heidelberg (2002)
12. Brice, C.R., L., F.C.: Scene analysis using regions. Artif. Intell. 1(3), 205–226 (1970)
13. Kovalevsky, V.: Finite topology as applied to image analysis. CVGIP 46(2), 141–161 (1989)
14. Kovalevsky, V.: Multidimensional cell lists for investigating 3-manifolds. Discrete Appl. Math. 125(1), 25–43 (2003)
15. Braquelaire, A., Damiand, G., Domenger, J.P., Vidil, F.: Comparison and convergence of two topological models for 3d image segmentation. In: Hancock, E.R., Vento, M. (eds.) GbRPR 2003. LNCS, vol. 2726, pp. 59–70. Springer, Heidelberg (2003)
16. Desbarats, P., Domenger, J.P.: Retrieving and using topological characteristics from 3D discrete images. In: Proceedings of the 7th Computer Vision Winter Workshop, PRIP-TR-72, pp. 130–139 (2002)
17. Braquelaire, A.: Representing and segmenting 2d images by means of planar maps with discrete embeddings: From model to applications. In: Brun, L., Vento, M. (eds.) GbRPR 2005. LNCS, vol. 3434, pp. 92–121. Springer, Heidelberg (2005)
18. Baldacci, F., Desbarats, P.: Parallel 3d split and merge segmentation with oriented boundary graph. In: Proceedings of The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (accepted, 2008)