Topological Encoding of 3D Segmented Images

Yves Bertrand, Guillaume Damiand, Christophe Fiorio

Topological Encoding of 3D Segmented Images

Yves Bertrand[1], Guillaume Damiand[2], and Christophe Fiorio[2]

[1] IRCOM-SIC, SP2MI
BP 179, F-86960 Futuroscope Cedex
bertrand@sic.sp2mi.univ-poitiers.fr
[2] LIRMM
161 rue Ada, F-34392 Montpellier Cedex 5
{damiand,fiorio}@lirmm.fr

Abstract. In this paper we define the *3d topological map* and give an optimal algorithm which computes it from a segmented image. This data structure encodes totally all the information given by the segmentation. More, it allows to continue segmentation either algorithmically or interactively. We propose an original approach which uses several levels of maps. This allows us to propose a reasonable and implementable solution where other approaches don't allow suitable solutions. Moreover our solution has been implemented and the theoretical results translate very well in practical applications.

1 Introduction

An early stage of an image analysis process is to group related information in the image into regions: it is segmentation. This problem has been widely studied and efficent algorithms which perform segmentation are well known. A related problem concerns the data structure used to represent the segmentation. Indeed, after segmentation, more complex algorithms (such as pattern recognition) need to be run in order to give useful results. These algorithms need a structuration of the data which represents completely all the information given by the segmentation. Moreover this structuration need to be efficiently computable. For example we may have to deal with the topological characteristics of the volume found, as well as to work interactively on the objects modelled.

Numerous data structures have been already proposed. For example, a simple one is to associate each pixel to the region it belongs to [15]. But this solution is not efficient if we have to cut or merge regions. There is also a solution using quadtrees or octrees [8,16] which are based on a recursive decomposition of the image. But this structure is difficult to modify and doesn't allow to check easily adjacency relations between regions. Same drawbacks can be reproached to the pyramidal structures [2,12].

Other structures focus on information related to the regions, for example the region adjacency graph [14], which codes the adjacency relation between regions. But it is not topologically consistent and doesn't relate the inclusion or multiple adjacency, so it doesn't allow a precise analysis.

G. Borgefors, I. Nyström, and G. Sanniti di Baja (Eds.): DGCI 2000, LNCS 1953, pp. 311–324, 2000.

Some other solutions try to use several of the formers in order to gather the advantages offered by each of them. But this lead to complex solutions, often non efficiently computable.

This is this lack of adequate solutions which has lead [10] to define the topological graph of frontiers. The interest of such a structure is to be consistent with the topology of the objects it represents. Moreover, this structure can be efficiently computed thanks to a linear algorithm in the number of pixels which proceeds in one scan of the image [9]. One can remark that this solution is close to the one of [3,4,5] which defines the 2d topological map, the difference being in the extraction algorithm.

But these solutions are only defined in dimension 2. The need to work in dimension 3 has lead [1] to propose the border map. This solution can easily be extended to dimension n. Moreover, they have proposed a very simple algorithm which extract this structure from a 3d segmented image. But the border map is not homogeneous. It mixes topological and geometrical information. Furthermore, the optimal algorithm proposed is not practicable in dimension 3 since it requires the definition of more than 4000 differents cases, which is impossible to achieve.

We then propose an evolution of the border map: the 3d topological map. This data structure is completely homogeneous and requires less memory than the border map. We also give an extraction algorithm which uses a limited number of cases and so allows an implementation. The topological and the geometrical information has been completely separated allowing to modify one type of characteristic on one object independently to the other.

In order to present the topological map, we define several maps levels, the border map becoming one of this levels. We start Sect. 2 by a short introduction on combinatorial maps. Then Sect. 3 we come back on the basis of border map. Section 4 we show the notion of topological map and the principle of the extraction algorithm in dimension 2. Then Sect. 5 we extend the precedent results to the dimension 3. At last Sect. 6 we show and analyse some results and we conclude this paper Sect. 7.

2 Combinatorial Maps

Combinatorial maps are a mathematical model of representation of space subdivisions in any dimension. They are consistent with the topology of the space as they also code the adjacency relation between all the subdivisions. For a complete definition see for example [6,7,11].

A combinatorial map of dimension n is called a *n-map*. The *n-map* allows to represent subdivisions of orientable n-manifolds. The notion of maps has been generalized in [13] for the representation of quasi-manifold cellular of dimension n, orientable or not.

In dimension n, a n-map is an $(n+1)$-tuple $M = (D, \beta_1, \beta_2, \ldots, \beta_n)$ such that D is the set of darts, β_1 is a permutation on D and the other β_i are

involutions[1]. Each β_i links two cells of dimension i. The β_i are called sews, and a dart b is said i-*sewn* to a dart c if $\beta_i(b) = c$. For each dimension the sets of darts corresponding to the i-cells are a partition of the set of all darts. Maps code the topology of objects, but it is not sufficient in order to represent them completely. Information on how drawing them are necessary: it's the notion of embedding. For a cells subdivision of space, this embedding associates to each i-cell a geometric object of dimension i.

For example in dimension 3, the embedding of a surface consists in relating a point to each vertex, an open curve to each edge and an open surface to each face. An open face is associated to each cell in order to avoid duplication of information. Indeed, border of an i-cell is an $(i-1)$-cell to which an embedding is already associated.

But according to our needs or constraints, we can choose not to embed each cell and compute the missing embedding. For example, in dimension 2, we can embed only the edge with closed curves. In this case, the embedding of the vertices can be retrieved by taken the extremities of one incident edge. It is then necessary to ensure the coherence between topological and geometrical information: if two edges are incident to the same vertex, there embeddings have to have the same extremity.

3 Border Maps

We give, here, a brief recall on the border map defined in [1]. Note that in this paper we use combinatorial map instead of generalized map, this, in order to save memory.

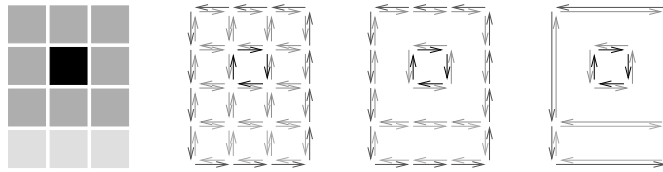
Definition 1. *The* border map *is a combinatorial map which codes the interpixel contours of a segmented image. Each edge of the map corresponds to a segment of the interpixel contours of the image.*

The border map codes the topology of the contours but also the geometry. So it contains all the information given by the segmentation about the objects found.

A level 0 map is the complete map obtained by sewing $n_1 \times n_2$ squares between them. Then level i map is the map obtained from i different sorts of merging. For dimension 2, we get the level 1 map by merging adjacent faces of the same region. Level 2 map is then obtained by merging adjacent edges on the same line incident to a degree 2 vertex[2]. This level 2 map is in fact the border map.

We can see Fig. 1, the border map of an image and also the two intermediate levels of maps. We can see on this example, that after the merging of faces, the map can be disconnected. This problem is solved by adding an inclusion tree to the connected components of the map. This tree can easily be computed by a linear algorithm in the number of darts of the map [1].

[1] An application f is an involution if and only if $f \circ f$ is the identity application.
[2] A vertex is of degree 2 if and only if it's incident to exactly two edges.



a. An image b. Level 0 c. Level 1 d. Level 2 : the border map

Fig. 1. The border map

For dimension 3, we proceed on a similar way to obtain the border map. But an additionnal level is necessary (see Algo. 1).

Algorithm 1: The border map in dimension 3

Input: An image I of $n_1 \times n_2 \times n_3$ pixels
Output: The border map of I.

1 Build a 3-map C of $n_1 \times n_2 \times n_3$ cubes sewn in between them;
2 Merge all couple of adjacent volumes having the same region;
3 Merge all couple of coplanar faces incident to a degree 2 edge;
4 Merge all couple of aligned edge incident to a degree 2 vertex;

The extraction algorithm in dimension 3 is simply the extension of the naive extraction algorithm of the border map in dimension 2. We now have three sorts of merging:

- Merging of adjacent volumes which consists in deleting the faces in between the volumes and to update the sews in order to have only one volume (see Fig. 2.*b*): we get the Level 1.
- Merging couple of coplanar faces which is equivalent to two merging of faces in dimension 2 (see Fig. 2.*a*): we get the Level 2.
- At last, the merging of edges on the same line which is equivalent to do, one by one, several merging of edges in dimension 2: we then get the Level 3 which is also the border map.

a. Face merging b. Volume merging

Fig. 2. Face and volume merging in dimension 3

As for dimension 2, an inclusion tree of the regions is needed. But it is not sufficient since we can also have non-connected faces. Indeed when the merging of coplanar faces is done, the border of faces can be disconnected and so the map. In order to solve this new problem we can add an inclusion tree of faces which, for each face, gives the list of faces included.

In order to embed a border map in dimension 2 or 3, it is sufficient to give the coordinates of each vertex of the map. Indeed only elements on the same plane or line are merged, so edges are only straight segments and faces are coplanar to these edges. So the embedding of the vertices allows to compute the embedding of an edge from the embeddings of the two incident vertices, and the embedding of a face from the embedding of all incident edges. This choice of embedding facilitates process and extraction of the border map.

But this leads to an important problem: the same topological information can be coded differently according to the geometry of the image : two topologically equivalent objects could have two different border maps.

The border map mixes topological and geometrical information. The notion of topological map has been introduced to solve this problem. We first present it in dimension 2 in order to help us to understand the idea of the algorithms and the relation in the differents precodes of our different levels. But the goal remains the definition of a 3d topological map and this is presented in Sect. 5.

4 2d Topological Map: Last Level of an Hierarchy of Maps

In the border map, some merging are done accordingly to the embedding of the associated cells. So, two topologically equivalent images, but with different embeddings will have two different border maps.

So we have to do merging only on topological criteria. Practically for dimension 2, in addition to the merging of edges on the same line and incident to degree 2 vertices, we also have to merge edges incident to degree 2 vertices but not on the same line. Edges embeddings are no longer line segment and embed only vertices as for the border map is no longer sufficient.

Definition 2. *The* topological map *is a combinatorial map which codes the interpixel contours of a segmented image. Each edge of the map corresponds to an entire interpixel frontier of the image.*

If we forget for a moment the problem of embedding, computing the topological map is not really a problem, there is just another sort of merging to add after the computation of the border map: the merging of each couple of edges incidents to a degree 2 vertex. This Level 3 map is the topological map. If we take the example of Fig. 1 showing maps of Level 0 to 2, the topological map is then the one of Fig. 3. On this figure, the darts are drawn without taking into account their embedding. In fact the embedding of each edge should be coded by an open curve. An interesting way of doing this is to use a 1-map representing the

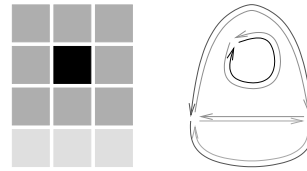


Fig. 3. Topological map without embedding

curve corresponding to the embedding of the edge. We then get an hierarchical structure where i-maps are used to code the embedding of i-cells.

We can see on Fig. 3 that there is exactly one edge for one frontier between two regions, so a frontier is always coded by exactly two darts. This imply that the topological map requires less memory than the border map. This can be very important to deal with 3d images. In the topological map only topological information darts have been kept. So two topologically equivalent images will give the same topological map, except for the embedding.

We can easily deduce from the definition of the topological map a first extraction algorithm: building the different levels, starting from level 0 and then each level is built from the former by doing the supplementary mergings. Indeed deducing one level from the former level is very simple. Moreover, the same method can easily be applied for the definition of the topological map and its associated extraction algorithm for any dimension.

But this algotihm is not optimal since it needs several image scans and it creates many darts which are in the following deleted. In order to improve this algorithm, [1] gives an algorithm based on precodes which computes directly the border map. We have extended this algorithm in order to compute the map of any level directly and in one scan.

4.1 Extraction of the Level 1 Map in One Scan

This algorithm scans the image with a 2×2 window and does the operations given by the configuration of the frontiers (called a precode) in the 2×2 window. So the map is built in one scan and no more than the exact number of darts are created.

The idea of the algorithm is that during a scan from top to bottom and from the left to the right, the face corresponding to the current pixel can only be merged with the one above or the one on the left. Then, We only have four possible cases which corresponds to all the combinations of merging this face with the two others. Figure 4 shows this four face merging precodes.

The algorithm uses two invariants which say that before processing one pixel:

- The map corresponding to the image already scanned is built.
- The dart corresponding to the right border of the last processed pixel is known.

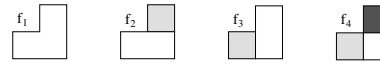


Fig. 4. The four precodes of face fusion

These two invariants insure us that we can proceed by modifying the map. Indeed we have to know how to link the map associated to the current pixel to the global map. So before scanning the image we have to build a map done with n_1 horizontal darts, corresponding to the upper border of the image, and 1 vertical dart to the left, all these darts being sewn by β_1. So the invariant is also verified for the first pixel and for the first line of the image.

During the scanning, we consider that the image is on a cylinder. Thus, when the algorithm treats the last pixel of a line, it can create the dart for the left border of the next line. Special treatments for borders of the image are so avoided. We only have to define the treatment associated for each of the four precode and the Level 1 map extraction algorithm will be defined. In fact this is easily done as we can see on the example of Fig. 5 and on Algo. 2 associated.

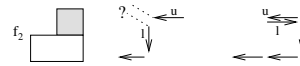


Fig. 5. Precode f_2: map before processing this precode and map to be obtained

Algorithm 2: Processing of precode f_2

$t_1 \leftarrow$ a new dart; $t_2 \leftarrow$ a new dart;
$\beta_1 - sew(t_2, \beta_1(l))$; $\beta_1 - sew(t_1, t_2)$;
$\beta_1 - sew(l, t_1)$; $\beta_2 - sew(l, u)$;
$(x, y) \leftarrow$ the embedding of vertex incident to u;
Embedding of vertex incident to $t_2 \leftarrow (x + 1, y + 1)$;
return *t1*

We can see on this example that not all the local configuration is known (the dart sewn to the dart u is not known), but it is not a problem since the operations to be done are independent of the fourth pixel. This precode returns the dart which will be the dart to the left for the next precode. At last we can see that the treatment is very simple, this is the same thing for the other three precodes.

Extraction algorithm of Level 1 map in one scan is very simple and is described in Algo. 3. This algorithm is in fact similar for all maps level, it is just necessary to define the precodes corresponding to the level concerned. So we will now focus on the precodes for the other levels of the map.

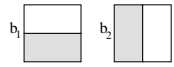
Algorithm 3: Extraction algorithm in one scan

Entrée: An image I of $n_1 \times n_2$ pixels
Sortie: Level i map of I.

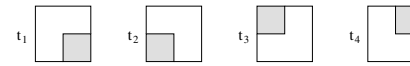
```
1 Build the upper border of the map;
2 for j = 1 n_2 + 1 do
      for i = 1 to n_1 + 1 do
          Process the precode (i, j);
```

4.2 Extraction of Level 2 Map in One Scan

For level 2 map, we have to add the merging of edges incident to a degree 2 vertex and on the same line. We can easily check that only two precodes are concerned by this merging: these are the precodes of Fig. 6.*a*.



a. The two precodes for level 2 map.



b. The four precodes for topological map.

Fig. 6. Precodes for level 2 and 3 maps

Finally we have six precodes for the Level 2 map: the four of the level 1 and these two of Fig. 6.*a*. The extraction algorithm is similar, we just have to look at the six precodes.

4.3 Extraction of the Topological Map in One Scan

For the topological map, in addition to the precedent merging, we have to merge the edges incident to a degree 2 vertex. These edges are necessary not on the same line, else they have been processed by one of the two precedent precodes for level 2 map. It is easy to see that there are only four precodes concerned by this sort of merging. These precodes are shown on Fig. 6.*b*.

In addition, this level requires an embedding for the edges, which can be represented by a 1-map. The general algorithm is also similar to Algo. 3 but ten precodes have to be checked. Nevertheless, this algorithm is still simple and can easily be implemented. It is also optimal and built directly the topological map.

Here we see the advantage of having defined several map levels: we have a relation between the number of precodes and the level of the map we want to extract in one scan. It allows to choose to extract directly Level i map accordingly to the need or development constraints, without paying more in processing time. This approach is particularly interesting in the 3d case where the number of precodes is huge.

5 3d Topological Map

As for dimension 2, we have to do merging according to topological criteria and without taking into account the embedding associated to the cells. We have seen, Sect. 3 that the border map of dimension 3 is the Level 3 map where adjacent volumes of the same region, coplanar faces incident to degree 2 edge and same line edges incident to degree 2 vertex have been merged. The level 4 map is then the map obtained from the Level 3 map where incident to degree 2 edge faces are merged.

As in dimension 2 where an edge embedding was added, this merging of non coplanar faces will require to add a face embedding. We can also use here a map for coding the embedding. Of course this map will be a 2d topological map where their edge embedding will be 1d map.

The topological map of dimension 3 (Level 5 map) is defined algorithmitically by merging in the Level 4 map each couple of edges incident to a degree 2 vertex: all the redundant topological information has been merged.

The simple algorithm which builds the map of Level i consists now to build a complete map (Level 0 map) of $n_1 \times n_2 \times n_3$ voxels sewn, then to build level after level the different maps. The Level $i+1$ map is obtained from the Level i map by doing the adequate additional merging.

[1] have tried to apply the same method as for dimension 2 in order to bring a more efficient algorithm. But they have calculated, thanks to the Stirling numbers, that it could be necessary to define 4140 different precodes in order to be able to compute optimally and in one scan the border map! In fact this number denotes the total number of possible precodes in dimension 3.

But we are interested only in manifolds, so in a first time, we have tried to calculate the number of manifold precodes. But we didn't manage to achieve a formula giving us this result. We finally computed it by the help of a program and found 958 manifold precodes. This number is more encouraging but the implementation of such a number of precodes is not yet easy. In fact the decomposition of the problem in different levels of map will bring us a reasonable solution.

5.1 8 Precodes for Level 1 Map

The scanning of the image starts from up to down, from behind to front of and from left to right. So a voxel can only be merged with the upper voxel, the voxel behind and the voxel to the left. We have then 8 possibility of volume merging and we get the 8 precodes of Fig. 7. More Generally, in dimension n, we get 2^n precodes for the Level 1 map. Indeed, the current element has exactly n neighbours already scan.

The extraction algorithm requires a similar invariant as for the dimension 2 version which ensure us that there exists in the map a face to the left, a face behind and a face with the top of the current voxel. The definition of the operations for each precode is not a lot more difficult that in dimension 2. The difficulties provide from the number of darts in one precodes, and from the fact that this

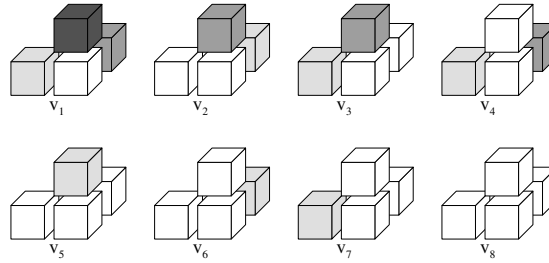


Fig. 7. 8 Precodes of Level 1 Map

is not easy to represent and visualize the object in dimension 3. The general algorithm is similar as the one of dimension 2, it requires only an additional loop for the additional dimension.

5.2 18 Precodes for Level 2 Map

At this level we have to do the merging of coplanar faces incident to degree 2 edge. Intuitively, in order to merge coplanar faces, the two voxels "containing" this faces are to be of the same region, and that the two front of voxels are of another region but the same for the two. We can see Fig. 8.*a* some examples of such precodes.

a. Two precodes for level 2 map. Two precodes for level 3 map.

Fig. 8. Examples of precodes for Level 2 and 3 Map

The low number of cases allows us to find exhaustively the 18 precodes for this sort of merging. We, then, immediately get the optimal extraction algorithm in one scan of Level 2 map: it's enough to check for each voxel in which of the 26 (the 18 for the coplanar face merging plus the 8 volume merging precodes) cases (precodes) it is and then to apply to corresponding operations on the map.

5.3 27 Precodes for the Border Map

For this level, we have to look at the merging of edges incident to a degree 2 vertex and on the same line. Intuitively, this merging can only be done if the vertex in the center of the precode is of degree 2. It means that all the voxels are two by two of the same region. Fig. 8.*b* shows some examples of such precodes.

In this case also, the low combinatorial allows us to find easily all the 27 precodes involved. So the total number of precodes to extract directly and in one scan the border map is of 53. We obtain a number which is far away from the 4140 first claimed in [1], and far away from the 958 cases of manifold. This last result is obtained thanks to our new approach which considers different levels of map.

But the topological map, which is our final objective, requires the implementation of Level 4 and 5.

5.4 Level 4 Map and Topological Map

Until the Level 3, only on the same line or on the same plane elements are merged. So the number of precodes remains limited and we have been able to determine all the precodes. But from level 4, things become more complicated and finding all the precodes is a lot more difficult and a long work. So we have just tried to determine the number of precodes for each level. This was done by the help of a program which, for each precode among the 958, builds the corresponding map, then does the merging required by the Level i considered. It is then sufficient to group the precodes with the same merging and count the number of equivalence classes.

The numbers of 8, 18 and 27 precodes for the first three levels of map have been confirmed. Then we get 98 precodes for the Level 4 map and 216 precodes more for the topological map. We recall that these numbers are the numbers of precodes to add to the precedent level. So the total number for an algorithm which would compute the topological map directly and in one scan would require the definition of 367 precodes! Our approach allows us to divide by half the total number of precodes required. But even if we are still far away from the 958 or better from the 4140 initially claimed, there are still too many precodes for a human.

But an intermediate solution exists. It consists in computing directly the border map by the way of the 53 precodes, and then to proceed as for the general algorithm and compute Level 4 and Level 5 (the topological map) by doing the additional merging on the border map. This solution is the one we have implemented and is an interesting compromise between the running time, the development time and the memory space required.

6 Experiments and Analysis

In dimension 2, we have implemented the extraction of maps for each level with the precode based algorithm. Results about memory needed can be seen on Table 1. The running time doesn't change a lot for the different levels and is about 0.20 seconds for a 512×512 on a 600 MHz PIII with 256 Mb of memory, included the computation of the inclusion tree.

In dimension 3, we can see on Table 2, for each level, the differences between the number of darts and the memory space. The method implemented is the

Table 1. Comparison of the different levels in dimension 2

	Level 1	Level 2	Level 3
Lena,	58 724 2d darts	31 648 2d darts	1 160 2d darts
256 region	-	-	15 244 1d darts
	1 650 004 Bytes	891 876 Bytes	350 948 Bytes
Lena,	220 900 darts 2d	139 848 darts 2d	67 112 darts 2d
14218 regions	-	-	36 368 darts 1d
	6 478 564 Bytes	4 209 108 Bytes	3 095 524Bytes

Table 2. Comparison of the different levels in dimension 3

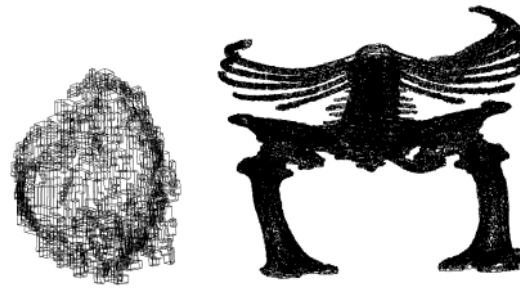
	Level 1	Level 2	Level 3	Level 5	Unit
Heart,	245 296	33 936	23 900	1 408	3d darts
$64 \times 64 \times 64$	-	-	-	16 435	2d darts
12 regions	-	-	-	225	1d darts
	7 998 920	1 235 400	854 032	598 068	Bytes
Skeleton,	10 109 680	1 354 412	922 072	44 300	3d darts
$512 \times 512 \times 177$	-	-	-	632 396	2d darts
4 regions	-	-	-	14 828	1d darts
	329 647 688	49 479 112	33 050 192	22 905 824	Bytes
Legs,	17 869 952	5 152 672	3 282 702	710 518	3d darts
$512 \times 512 \times 55$	-	-	-	1 667 613	2d darts
13926 regions	-	-	-	184 554	1d darts
	584 564 284	177 611 324	117 772 284	82 155 560	Bytes

one described at the end of Sect. 5.The computation of the Level 5 map takes about 3 seconds for the heart (Fig. 9.*a*), 470 seconds for the skeleton (Fig. 9.*b*) and about 200 seconds for the legs. The running time of the other levels is a little more important, due to much more dart creations and so much more memory allocation calls.

We can see on Table 2 that the gain in memory is very significant. These first results confirm that the 3d topological map is well suited for applications on 3d images. Indeed in dimension 3 the big amount of memory required could limit the size of the images processed. For example, for the legs where all regions are considered, only 55 slices have been processed and the level 1 map requires too much memory space to be used in a basic computer. In the contrary the topological map only needs about $80Mb$ which is reasonable and can be used on a classical computer.

7 Conclusion

In this paper we have presented a data structure and an algorithm which reconstruct a 3d segmented image. This structure, the 3*d topological map* represents



a. Heart b. Part of a skeleton

Fig. 9. Topological maps extracted from a segmented image

totally the objects found by the segmentation. It allows to continue the segmentation either algorithmically or interactively, since based on combinatorial maps used in geometric modeling. This structure can be used to compute geometric properties as well as topological properties of the objects represented.

The approach proposed which uses different levels of simplification has allowed to define on a simple manner the structure and to propose a simple, linear (but not optimal) extraction algorithm. In order to achieve an optimal solution, we have introduced a new algorithm based on precodes. This algorithm builds the different levels of map in one scan of the image and is then very efficient. Nevertheless, it remains very simple since it proceeds by checking a configuration, called a precode, and realizes the operations associated to the met configuration.

In order to implement this algorithm, it is then sufficient to define these operations. A first study gave 4140 possibles cases, but we have proven here that there are only 958 precodes suitable for the non-manifold case which we are interested in, and moreover that our approach gives the possibility to study only one half, i.e. 367 precodes, for the last level of map. But the intermediate levels require only 8, 18, 27 and 53 precodes and then can easy be implemented.

This level approach allows to choose an hybrid solution in order to avoid the 367 precodes: compute directly an intermediate level of map and then build the last levels from this one. This is the solution we have implemented and tested. The border map have been computed directly with the 53 precodes and then, level 4 and the topological map has been built from the border map by doing appropriate merging of elements of the map.

At last our experiments prove that the topological map requires less memory than the other map levels, which is an important point for 3d images. Moreover in order to apply segmentation algorithm on a structure it is important that it doesn't code redundant information. In fact, adjacency relation, that is topological information, is the key point of such algorithms. Our structure is then well-adapted, not only to reconstruction and modeling, but also to image analysis .

Acknowledgements

We wish to thank Lionel Gral for useful comments and careful reading of this paper.

References

1. Y. Bertrand, C. Fiorio, and Y. Pennaneach. Border map : a topological representation for nd image analysis. In G. Bertrand, M. Crouprie, and L. Perroton, editors, *Discrete Geometry for Computer Imagery*, number 1568 in Lecture Notes in Computer Science, pages 242–257, Marne-la-vallée, France, March 1999. 312, 313, 316, 319, 321
2. M. Bister, J. Cornelius, and A. Rosenfeld. A critical view of pyramid segmentation algorithms. *Pattern recognition Letters*, 9(11):605–617, 1990. 311
3. J. P. Braquelaire and L. Brun. Image segmentation with topological maps and inter-pixel representation. *Journal of Visual Communication and Image Representation*, 1(9):62–79, 1998. 312
4. J. P. Braquelaire and J. P. Domenger. Representation of segmented images with discrete geometric maps. *Image and Vision Computing*, pages 715–735, 1999. 312
5. L. Brun. *Segmentation d'images couleur à base Topologique.* Thèse de doctorat, Université Bordeaux I, décembre 1996. 312
6. R. Cori. *Un code pourles graphes planaires et ses applications.* PhD thesis, Université Paris VII, 1973. 312
7. R. Cori. Un code pour les graphes planaires et ses applications. In *Astérisque*, volume 27. Soc. Math. de France, Paris, France, 1975. 312
8. R. C. Dyer, A. Rosenfeld, and H. Samet. Region representation : boundary codes for quadtrees. In *ACM 23*, pages 171–179, 1980. 311
9. C. Fiorio. *Approche interpixel en analyse d'images : une topologie et des algorithmes de segmentation.* PhD thesis, Université Montpellier II, 1995. 312
10. C. Fiorio. A topologically consistent representation for image analysis: the frontiers topological graph. In S. Miguet, A. Montanvert, and S. Ubeda, editors, *6th International Workshop, DGCI'96*, number 1176 in Lecture Notes in Computer Sciences, pages 151–162, Lyon, France, November 1996. 312
11. A. Jacques. Constellations et graphes topologiques. In *Combinatorial Theory and Applications*, pages 657–673, Budapest, 1970. 312
12. W. G. Kropatsch. Building irregular pyramids by dual graph contraction. In *IEE Proceedings : Vision, Image and Signal Processing*, volume 142(6), pages 366–374, 1995. 311
13. P. Lienhardt. Subdivision of n-dimensional spaces and n-dimensional generalized maps. In 5^{th} *Annual ACM Symposium on Computational Geometry*, pages 228–236, Saarbrücken, Germany, 1989. 312
14. A. Rosenfeld. Adjacency in digital pictures. *Inform. and Control*, 26:24–33, 1974. 311
15. A. Rosenfeld and A. C. Kak. *Digital Picture Processing*, volume 2. Academic Press, New York, 1982. 311
16. H. Samet. Region representation : quadtrees from boundary codes. In *ACM 23*, pages 163–170, 1980. 311