Parallel 3D Split and Merge Segmentation with Oriented Boundary Graph

F. Baldacci
LaBRI, Université Bordeaux 1,
France
baldacci@labri.fr

P. Desbarats
LaBRI, Université Bordeaux 1,
France
desbarats@labri.fr

ABSTRACT

In this paper, we present how 3D split and merge segmentation using topological and geometrical structuring with an Oriented Boundary Graph may be optimized by parallel algorithms. This structuring allows to implement efficiently split and merge operations, but since these treatments have often to be applied with large images, we have studied how to improve performances by parallelizing this process. After a short description of the structuring model and its construction, we describe algorithms for parallelizing the construction of the structuring and describe how this model can be maintained while using parallel processes. We explain the way of partitioning data for use with multiprocessor systems, and extension for use with NUMA architectures and graphics processing units (GPU) is described. Exemples on two medical images of different sizes is presented and execution time will be given.

Keywords
Parallel segmentation, Split and Merge 3D, Topological structuring, Topological graph, NUMA architecture, Medical image segmentation.

1 INTRODUCTION

Image segmentation is a process that leads to its partition into regions. Two main approaches exists : contour detection algorithms and region-based methods. Contour detection algorithms are based on the idea that contours delimitate rapid changing in image intensities which can be detected for example by using derivative operators [Can86, Der87] while region-based methods regroup all neighboring 2D pixels (or voxels in 3D) according to an homogeneity criterion (such as region intensity mean or variance) [Har85, Bis94].

Split and merge algorithms belong to the latter one [Hor74]. Their goal is to alternately split non-homogeneous regions or merge adjacent regions with the same criterion value. An efficient implementation of related algorithms needs a structuring of the image. This structuring have to facilitate the retrieval of geometrical features (such as the domain corresponding to a region) and topological ones (such as the list of adjacent regions for a given region or the regions included into it). Several models have been proposed in 2D and 3D [Bra99, Fio96, Ber00, Bra99, Bra03]. In order to capture both the geometry and the topology of the image, most of these models are based on the cellular decomposition of the discrete space and on combinatorial maps. If such a structuring is straightforward in 2D, topological problems arise in 3D and leads to either non minimal representations or non fully geometrically embedded models. Furthermore, combinatorial maps require to decompose region boundary into surface elements homeomorphic to a topological disc, which needs extra processing on surfaces resulting from a segmentation. For this reason, while 2D split and merge algorithms based on this kind of structuring are efficient [Bra98], the first 3D split and merge algorithms developed are fully functionnal but not optimal [Dam02, Bra01].

As most of segmentation algorithms do not require the whole topology of the image, we have proposed an other model [Bal08], lighter than models based on combinatorial maps. It is especially designed to be efficient with split and merge algorithms. We will recall this model and its construction in the first part of this article.

Another problem occurs when dealing with 3D medical images, industrial scans or microscanner acquisitions. The new generation of acquisition devices produces high resolution images leading to huge amount of data. For example, a classical 3D X-Ray microscanner acquisition at a $25\mu m$ spatial resolution of a lumbar vertebra weights almost 4 Gb of data. It is thus very difficult to process such images as a whole in physical memory and computation times for segmen-

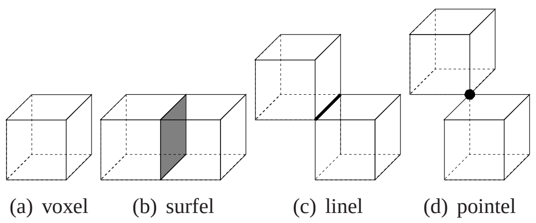


Figure 1: Elements of the cellular decomposition of 3D discrete space.

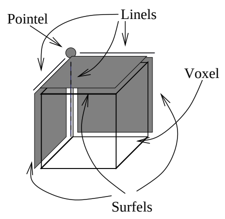


Figure 2: A voxel and its corresponding encoded intervoxel elements in the boundary image.

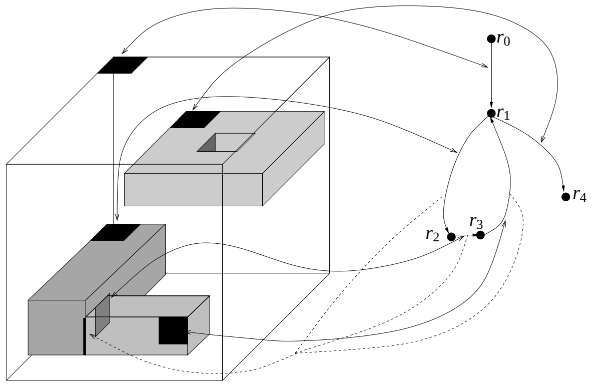


Figure 3: Example of image with the corresponding representation in our model.

tation will be crippling. A classical method to address this problem is to partition the image and parallelize treatments. In the field of image segmentation, several parallel implementations of 2D or 3D algorithms have been presented (see for example [Bad96, Mog97, Til89, Kho90, Aln91, Mon03] or [Pit93] for applications on MIMD parallel machines). They mainly differ in the way they partition image data and in their parallel machine-dependant implementations. Nowadays, multi-core and multiprocessors computers are available as working stations, which democratizes and facilitates the use of lightly-parallelized algorithms. Our approach is slightly different from the common one. We chose to use multi-core/multiprocessor implementation and the parallelization is not done on the segmentation algorithm but on the construction of the structuring model it needs to be efficient.

In this article, our goal is to present a feasibility study on how to implement an efficient parallel 3D split and merge algorithm, while using our structuring model, in oder to show that our model do not prevent from using parallel algorithms. The parallelization will be proposed on the construction of the structuring model needed. We present first a recall of our structuring model, then we explain our data partitionning method. After a presentation of our results on different multicore/multiprocessor computers, we will conclude about the feasability of such algorithms and discuss about the future of our work.

2 STRUCTURING MODEL DESCRIPTION AND CONSTRUCTION

A segmented image is a partition of the image into regions according to criteria. A Region is a set of 6-connected voxels. Regions intersections define separating surfaces which need the encoding of intervoxel elements to be represented. Intervoxel representation lays on the elements of the cellular decomposition of 3D discrete space (Figure 1) which are voxels (elements of dimension 3), surfels (intersection of two voxels), linels (surfels intersection) and pointels (linels intersection) [Kov89, Kov03].

Our model [Bal08], called *Oriented Boundary Graph* (*OBG*), is composed of a topological level encoding region adjacency relation and surface adjacency relation, and a geometrical level encoding intervoxel elements. Links between the two levels are necessary to efficiently update the representation. The link is done by associating an oriented surfel to each surface in the *OBG*.

The topological level is a multiple region adjacency graph, encoding each region with a node and each separating surface with an edge. Furthermore, edges corresponding to adjacent surfaces are linked in the graph. This is necessary to efficiently compute the merge operation that can lead to adjacent surfaces merging.

The geometrical level is a matrix, called boundary image, encoding the presence of three surfels, three linels and one pointel on one byte, corresponding to the voxel with same coordinates in the image (Figure 2).

In the topological level, each edge encodes an embedding surfel allowing to retrieve its corresponding surface in the boundary image, and which incident voxel belongs to the regions corresponding to the extremity nodes. Those representative surfels are also marked in the geometrical level and represent entry points to the graph from the geometry. Figure 3 shows an example of a segmented image and its representation with the different types of links.

The construction of the model is done using the split operation considering the region to split is the entire image. We consider that criteria needed for this split operation is given by an *Oracle* function. This function answers if two given voxels belong to the same region and is sufficient to describe the segmentation of an image. The split operation (Algorithm 1) consists in labelling identically voxels belonging to a same region and surfels belonging to a same surface. Two different regions or surfaces can not have the same label. For each voxel label a node is added in the graph and for each surfel label an edge is added. Those labelling are done by traversing the image and labelling voxels and surfels. Linels, which are needed to traverse single surfaces faster, are created in the same time. Image traversing is done with a scanline method to avoid some tests, and thus a region or a surface can have several labels. In this case their corresponding topological elements are merged and an indirection table is maintained to avoid an other labelling operation.

Algorithm 1 Split Algorithm

```
Require: An OBG, A domain D and an Oracle function
1:  for all Voxel $v \in D$ do
2:    for all labelled voxel $v_i$ neighbors of $v$ do
3:      if $Oracle(v_i, v)$ then
4:        SaveLabel(list, $v_i$)
5:      else
6:        addSurfelBetween($v_i, v$)
7:      end if
8:    end for
9:    if NumberOfLabel(list) $> 1$ then
10:     MergeRegion(list)
11:   else
12:     if NumberOfLabel(list) == 0 then
13:       CreateNewRegion(NewLabel)
14:     end if
15:   end if
16:   for all 4 labelled voxel cuboid containing $v$ do
17:     if more than 2 surfels are present then
18:       AddLinel(x, y, z)
19:     end if
20:     LabelAddedLinels()
21:     LabelSurfels()
22:   end for
23: end for
```

Merging two regions (Algorithm 2) consists in deleting all the surfels belonging to surfaces shared by both regions from the boundary image, and the corresponding edges from the graph. It can result that several surfaces have to be merged if only two surfaces remain incident to a linel. This condition can be detected during the geometrical traverse of surfaces by looking at encountered linels.

3 PARALLELIZATION

We propose in this paper a method to study the possibility to reduce the split operation computation time. This analysis does not pretend to be exhaustive as it is only a feasibility study driven by the need to process large 3D images. Note that the merge operation can also be parallelized with few constraints on our structuring model.

Algorithm 2 Merge Algorithm

```
Require: 2 regions $r_1$ and $r_2$ to merge
Ensure: The new region $r = r_1 \cup r_2$
1:  for each surface of $r_1$ do
2:    if it separate $r_1$ from $r_2$ then
3:      linellist $\leftarrow$ deleteSurfaceFromGeometry()
4:      DeleteSurfaceFromTopology()
5:    end if
6:  end for
7:  for each linel of linellist do
8:    if its degree is 2 then
9:      DeleteLinelFromGeometry()
10:     if it is a representative linel then
11:       MergeSurfaces(linel)
12:     end if
13:   else
14:     if its degree is less than 2 then
15:       DeleteLinelFromGeometry()
16:     end if
17:   end if
18: end for
```

The first tests on classical images show that the merge process is far less time consuming than the split one. For example merging 100000 regions takes about 1 second on our test platform with only one thread. Future parallelization of the merge operation will be discussed in section 5, but similar results than with the split operation can be expected.

We use a eight cores machine and a sixteen cores machine with NUMA (Non Uniform Memory Access) [Bol91, Nie96] architecture (see Section 4 for more details). Our goal is to launch several threads on independent data and then to merge the different results. Since they do not manipulate the same labels, threads can be executed in parallel without constraint. Label management have to be protected in order to ensure that the label distribution is done without collision, and the outside of the image must have a different label for each thread. The fact that two threads do not have access to a same label ensures that each thread can build the *OBG* corresponding to its part independently, and that each part can be glued to the others only by analysing its border. It might be possible for each thread to manage its label by itself for an adaptation on some architectures where asking for a label is not efficient (for example computer clusters). In this case the gluing operation induces to label one of the two graphs with new labels, and induces some extra treatments to analyse the junction where different surfaces can have the same label.

Our approach consists in cutting the data into blocks such as the union of all the blocks corresponds to the entire image. Each thread has its own label for the outside of the image and computes its *OBG* by treating all the elements of its block except surfels and linels belong-

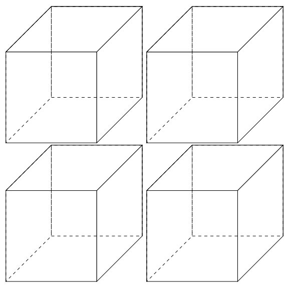


Figure 4: Optimal data cutting

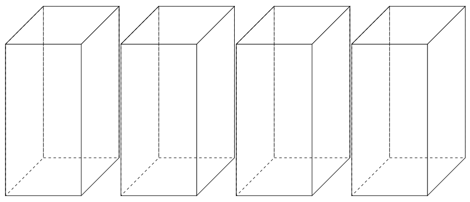


Figure 5: Implemented data cutting

ing to the intersection with another block. When two adjacent blocks are processed, a thread could be launch to glue them. The gluing operation consists in merging in the graph the two nodes corresponding to the outside of the image, and then to test for each surfel of the intersection if it belongs to a region boundary (and thus labelling it and eventually merging some surfaces by looking at its neighbors), or if the regions of the two incident voxels have to be merged in the graph. Linels are processed in the same manner. Note that a block can not simultaneously be glued with several other blocks, due to the impossibility to access a same label from several threads.

The optimal way for making independent blocks is achieved by minimizing the number of surfels that have to be post-processed. In order to have an easier gluing operation we have implemented only a cutting along one axis. This leads to more post-processings but it is sufficient for testing possible improvements induced by parallelization. Figure 4 shows the optimal cutting to use with four threads, and the Figure 5 our implemented cutting. It can lead to some variations on the execution time that have to be considered in the analysis. For example, on a 512x512x512 image with sixteen blocks, the optimal cutting needs 1310720 surfels to be post-processed, whereas our implemented cutting method needs 3932160 surfels to be post-processed. Even if this operation is parallelized, it needs more tests for treating a surfel than during the normal process.

4 RESULTS

The implementation have been tested on two machines. The first one is equipped with two intel Xeon Quad core at 2,33Ghz, and the second one is equipped with eight AMD Opteron Dual core at 1,8Ghz but with a NUMA architecture. Although our method is suitable for use with a NUMA architecture (because each thread has its own data and so it is possible to address each processor with a partition of the data), our partition implementation is not optimized for such architecture.

	Femur	Brain
size	512x512x475	256x256x256
Regions	456396	73453
Surfaces	1638726	291337

Figure 6: Tested Images Specification

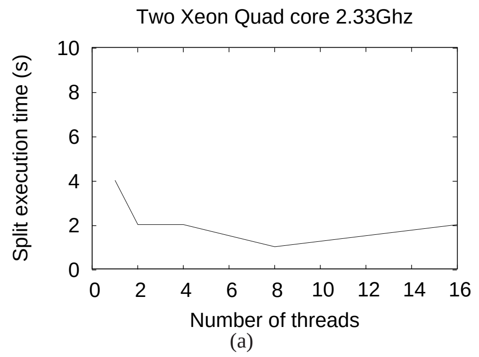
Our split criterion is a classifier that returns the class of a voxel according to its value. Two adjacent voxels are considered to be in the same region if they belong to the same class according to this classifier.

We used two medical images to make our tests (Figure 6). The first one is a MR anatomical image of a brain of size 256x256x256 voxels. The first split operation needed by the model construction gives 73453 regions with 291337 surfaces. The second image represents a CT scanned femur and its size is 512x512x475 voxels. The split operation give 456396 regions and 1638726 surfaces.

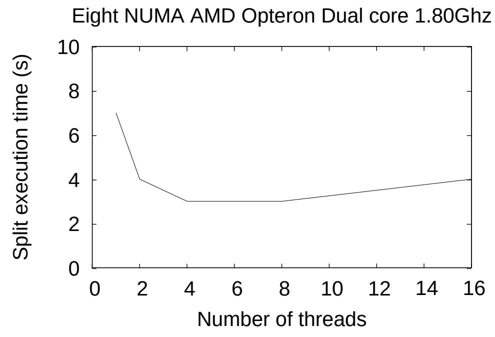
Before analysing the results it is important to recall that our implementation is a testing one that could be easily improved by making an optimal cutting of the data, and an adapted distribution of the memory for the NUMA architecture.

Execution time for the split operation on both images according to the number of threads are presented Figure 7 and 8. First, we can notice that the NUMA architecture is two time slower than the other one, partially due to the lower core frequency and also due to our inappropriate implementation. Using sixteen threads is slower than using eight threads due to our unoptimal data cutting method. In general, it is not efficient to make too small blocks of data, because it leads to more gluing operations. This means that the number of threads has to be adapted according to the image size. Despite this drawbacks, we can observe that our testing implementation is sufficient to significantly reduce computation times. Using eight threads reduces the execution time by a factor three on both architectures and with both images. This results show the possibility of reducing computation time for split operation in a segmentation process even with the use of our high level structuring of the image. This is an important result for the validation of our model and the proof of its usability with large images.

Merge operation can be parallelized in a same way. In order to maintain our structure we have to impose the same constraints as for the split operation. This means that a region cannot be simultaneously merged with several other regions. It can be easily controlled by using a protected boolean table encoding which regions are in a merging process. This is an important

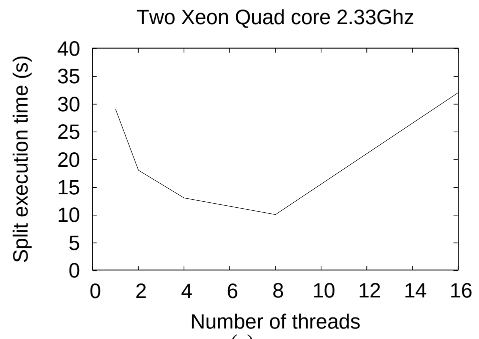


(a)

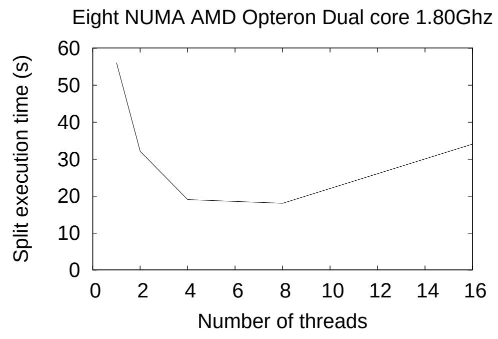


(b)

Figure 7: Execution time on the brain image.



(a)



(b)

Figure 8: Execution time on the femur image.

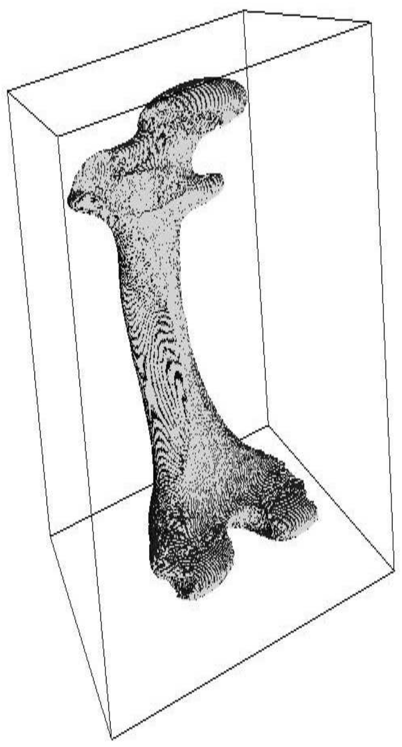


Figure 9: Segmentation result on the femur image

extension necessary for large images on which deleting surfaces will be a slow operation.

Figure 9 shows the segmentation result after computing merge operations on the result of the initial split. First we have merged little isolated regions with their unique neighbor. Then we have used criteria based on the mean difference and size of adjacent region (both criteria have been computed during the split operation and are included in our computational time results). This merge operation only takes few seconds and so not have been parallelized. The result image is only composed of eight regions. It means that the bone has been reconstructed while noise and unnecessary regions (as the scan table supporting the femur) have been removed.

5 DISCUSSION

Since the implemented algorithm was designed in order to study the improvements achievable by parallelizing split and merge operations with the constraint of maintaining a topological structuring of the segmented image, some optimizations and architecture adaptations still have to be done. The encouraging results obtained by our tests show that our structuring model can be used with large size images which need parallel treatments in order to be segmented in a reasonable time. Furthermore, the manner in which the data is partitioned by giving each thread a separate part of the data to analyze allows its usability on every type of parallel architecture. Thus the same algorithm can be efficiently adapted and implemented on NUMA architecture or computer clusters using a high-performance network.

The interest of our method to separate data and protect the access to our representation is that it can be used to parallelize also the merge operation. Even if the parallelization of the merge algorithm is not in the scope of this article, we can expect by looking at the algorithm given at the end of section 5 that we will have similar results than for the split algorithm. Note that in the case of a parallelized merge process, merging criteria can also be computed in parallel as this computation is done by traversing each region. Thus a major improvement in data partionning will be not to cut the image uniformly but to use our structure by adressing regions to threads.

The possibility of building our global structuring by dividing its construction to threads applied on separated data is mandatory to apply the algorithm on Graphical Processing Units (GPUs). New generation of GPUs have been designed to be data-parallel computing devices. There are multi-core based with very high memory bandwidth. For example, NVidia's GeForce 8800 is composed of 16 multiprocessors. Each multiprocessor being composed of 8 processors, the GPU is theoretically able to process 128 threads at once [Nvi07]. For the "Ultra" GPU of this serie, the memory bandwidth is 103,7 Gbits/s and the memory size is 768 Mb. Furthermore, processes can be parallelized for more than one GPU. In this case, even if the global memory is not shared, GPUs can communicate with each other. In order to fully exploit this kind of architecture, the partitioning of the image will once again be a critical step.

A GPU specific labelling algorithm can be designed for our purpose. It will be a type of multi-agent system [Kab02, Ben07] with the aim of labelling all voxels and surfels of the image. The first type of agent will label identically all the voxel of a region while detecting surfels composing its border. The same kind of agent will be defined for labelling surfels composing a surface. Indirection can eventually be envisaged depending on the size of shared memory. Using agent corresponding to very light-weighted threads may be a good way to exploit the parallel computability of GPU, since we know it is possible to directly extract the *OBG* from the results of those labelling.

6 CONCLUSION

An efficient parallel 3D split and merge algorithm has been presented in this article. Our goal was to demonstrate the feasibility of such a parallelization on our 3D structuring model construction.

The first tests we made gave encouraging results. As these tests were made on different architectures, they allow us to have a good overview on this kind of parallelization problems. Optimizations will first consist in adapting our data partitioning method to the targeted computer architecture and then to parallelize the rest of the processes (the merge and the segmentation algorithm itself). We also plan to port our algorithms on the new GPU generation to investigate its performance for our purpose.

ACKNOWLEDGEMENTS

The authors wish to thank Lionel Barrère, member of the SOD (Distributed Objects and Systems) research team of the LaBRI, for his knowledge in multi-core / multiprocessor machine and for his help in implementing the source code for our tests.

REFERENCES

[Bis94] Bischof L. Adams R. Seeded region growing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16:641–647, 1994.

[Aln91] H.M. Alnuweiri and V.K. Prasanna Kumar. Fast image labeling using local operators on mesh-connected computers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(2):202–207, 1991.

[Bad96] D.A. Bader, J. Jájá, D. Harwood, and L.S. Davis. Parallel algorithms for image enhancement and segmentation by region growing with an experimental study. *J. Supercomputing*, 10(2), 1996.

[Bal08] F. Baldacci, A. Braquelaire, P. Desbarats, and Domenger J.P. 3d image topological structuring with an oriented boundary graph for split and merge segmentation. 2008. Accepted 14th IAPR International Conference on Discrete Geometry for Computer Imagery.

[Ben07] N. Benamrane and S. Nassane. Medical image segmentation by a multi-agent system approach. In *Multiagent System Technologies*, volume 4687 of *Lecture Notes in Computer Science*, 2007.

[Ber00] Y. Bertrand, G. Damiand, and C. Fiorio. Topological encoding of 3d segmented images. In *Proceedings of 9th Discrete Geometry for Computer Imagery*, volume 1953 of *LNCS*, pages 311–324, December 2000.

[Bol91] William J. Bolosky, Michael L. Scott, Robert P. Fitzgerald, Robert J. Fowler, and Alan L. Cox. Numa policies and their relation to memory architecture. *ACM SIGOPS Operating Systems Review, Special Issue, Proceedings of the 4th international conference on architectural support for programming languages and operating systems*, 25:212–221, 1991.

[Bra98] J.-P. Braquelaire and L. Brun. Image segmentation with topological maps and interpixel representation. *Journal of Visual Communication and Image Representation*, 9(1):62–79, mars 1998.

[Bra99] J.-P. Braquelaire and J.-P. Domenger. Representation of segmented images with discrete geometric maps. *Image Vision Comput.*, 17(10):715–735, 1999.

[Bra99] A. Braquelaire, P. Desbarats, J.-P. Domenger, and C. Wütrich. A topological structuring for aggregates of 3d discrete objects. In *2nd IAPR-TC-15 Workshop on Graph-based representation*, pages 193–202. OCG, 1999. ISBN 3-8580-126-2.

[Bra01] A. Braquelaire, P. Desbarats, and J.-P. Domenger. 3d split and merge with 3-maps. In *3rd IAPR-TC-15 Workshop on Graph-based representation*, pages 32–43. CUEN, 2001. ISBN 887146579-2.

[Bra03] A. Braquelaire, G. Damiand, J.-P. Domenger, and F. Vidil. Comparison and convergence of two topological models for 3d image segmentation. In *Proceedings of 4th IAPR-TC15 Workshop on Graph-Based Representations in Pattern Recognition*, volume 2726 of *LNCS*, pages 59–70, July 2003.

[Dam02] G. Damiand and P. Resch. Topological map based algorithms for 3d image segmentation. In *Proceedings of 10th Discrete Geometry for Computer Imagery*, volume 2301 of *LNCS*, pages 220–231, April 2002.

[Fio96] C. Fiorio. A topologically consistent representation for image analysis: the topological graph of frontiers. In *DCGA '96: Proceedings of the 6th International Workshop on Discrete Geometry for Computer Imagery*, pages 151–162. Springer-Verlag, 1996.

[Har85] R.M. Haralick and Shapiro L.G. Survey: Image segmentation techniques. *Computer Vision and Graphics: Image Processing*, 29, 1985.

[Hor74] S.L. Horowitz and T. Pavlidis. Picture segmentation by a directed split and merge procedure. In *ICPR74*, pages 424–433, 1974.

[Can86] Canny J. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–714, 1986.

[Kab02] Y. Kabir and A. Belhadj-Aissa. Distributed image segmentation system by a multi-agents approach (under pvm environment). In *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 9th European PVM/MPI Users' Group Meeting, Linz, Austria*, volume 2474 of *Lecture Notes in Computer Science*, pages 111–114, 2002.

[Kho90] A Khotanzad and A Bouarfa. Image segmentation by a parallel, non-parametric histogram based clustering algorithm. *Pattern Recognition*, 23(9):961–973, 1990.

[Kov89] V. Kovalevsky. Finite topology as applied to image analysis. *CVGIP*, 46(2):141–161, May 1989.

[Kov03] V. Kovalevsky. Multidimensional cell lists for investigating 3-manifolds. *Discrete Appl. Math.*, 125(1):25–43, 2003.

[Mog97] A. Moga and M. Gabbouj. Parallel image component labeling with watershed transformation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):441–450, 1997.

[Mon03] M.D.G. Montoya, C. Gil, and I. García. The load unbalancing problem for region growing image segmentation algorithms. *Journal of Parallel and Distributed Computing*, 63(4):387–395, 2003.

[Nie96] J. Nieplocha, R. J. Harrison, and R. J. Littlefield. Global arrays: A nonuniform memory access programming model for high-performance computers. *The Journal of Supercomputing*, 10(2):169–189, 1996.

[Nvi07] NVidia Corp. *NVidia CUDA (Compute Unified Device Architecture) Programming Guide version 1.0*, June 2007.

[Pit93] Ioannis Pitas, editor. *Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks*. John Wiley and Sons, 1993.

[Der87] Deriche R. Using canny's criteria to derive an optimal edge detector recursively implemented. *Int. J. Computer Vision*, 2:15–20, 1987.

[Til89] J.C. Tilton. Image segmentation by iterative parallel region growing and splitting. In *Geoscience and Remote Sensing Symposium, IGARSS'89. 12th Canadian Symposium on Remote Sensing*, pages 2420–2423, 1989.