Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

VOLUME SEGMENTATION OF 3-DIMENSIONAL IMAGES

by

CHRISTOPHE FIORIO   JENS GUSTEDT

NO. 515/1996

Volume Segmentation of 3-dimensional Images

Christophe Fiorio[*]
TU Berlin, Sekr MA 6-1
10623 Berlin, Germany
fiorio@math.tu-berlin.de

Jens Gustedt
TU Berlin, Sekr MA 6-1
10623 Berlin, Germany
gustedt@math.tu-berlin.de

Abstract

We present a practical method to segment large medical images that takes the whole 3-dimensional structure into account. We use a Union-Find data structure to record and maintain the necessary information during the segmentation process. Due to the large data size, we are forced to divide our process in two parts: a "*weak segmentation*" of the individual sections and a global integration of all the data. This method shows good results on computer tomographies.

1 Introduction and Overview

In this paper we present a method to perform real 3-dimensional segmentation of tomography images (for a more general issue on medical imaging, see e.g. [Her83]). The data in these images is a set of cross-sectional (slice) digital images of a part of a human body. Certain parts of this image, such as organs, bones, ..., may need to be visualized and manipulated in order to study the patient. But the extraction of such parts, as well as the definition of their boundaries is fundamental to visualization, manipulation and analysis.

In the 2-dimensional case, the problem of the definition of boundaries and relative notions as connectedness have long been studied, see [KR89] for a survey. In the 3-dimensional case, the topology of images is still well-studied, see e.g. [Kov89, KKM90, Lat93, AAF95]. Once the model of boundaries and connectedness is chosen, an algorithm which realizes the extraction of the boundaries of an object is necessary. There are many works on this problem, see e.g. [AFH81, KU92, RKW91, Udu94]. But all these algorithms take a binary digital image as input such that the "lightened" voxels are those which belong to the object to be described., i.e. the object in question has already been extracted.

Several methods which perform such extraction exist, see [UH91], but the common and non-satisfactory approach is to segment each slice separately and identify in each of them the region(s) or the edges which match the object (organ in our case) that is to be extracted, see e.g. [GL93]. There is another approach described in [MBA93] which also take a binary image but where a 3D-edge segmentation is assumed to be done (as for example the one described in [MDMC90]) and perform a topological reconstruction.

Our approach is slightly different, since we will take the 3-dimensional image into account and then identify volumes in this image as a whole. So afterwards, only the volume that is to be displayed must be selected, no additional operation to retrieve the 3 information is necessary.

We define a volume in the classical way as a set of connected voxels, where the connectivity relation is the 6-connectivity relation (see [KR89] for a definition).

Our method will perform *volume growing*, by extending the region growing in the 2D case. We are starting with small volumes and we merge them together according to a predicate. This predicate will use statistical informations of the volume to take its decision. Note that our method allows to use local criteria as well as global ones. Indeed our structure provide us with a connected component labeling, so for a given pair of voxels we have local informations related to the voxels themselves, and global informations relative to the segment[1] they belong to.

First we shortly present the basics of Union-Find and how it can be applied to image segmentation. Then, our approach is presented: we justify the use of a procedure in 2-parts , *"weak segmentation"* and show how we deal with with the entire 3-dimensional image. It follows a description of an application to computer tomographies and a presentation of results. Finally some perspectives are given.

[*]The research presented in this paper was made possible by a visiting grant of DIMANET per Christophe Fiorio at Berlin.

[1]which can be retrieve by a Find operation.

2 Using Union-Find for Segmentation

The use of Union-Find data structures for image segmentation is immediate: segments are just sets if we suppress the connectivity for a moment; the Union operation then corresponds to the merging of two segments; the Find to the identification of a particular segment. To add the connectivity constraint, we only have to ensure that only pairs of adjacent pixels are taken in order to ask for the merging of two segments. Then the corresponding segment are guaranted to be adjacent and the merging of the two is indeed connected.

2.1 Basics

The general Union-Find problem, or more precisely *the disjoint set-union problem*, can be formulated as follows. Given is a set S, the ground-set, of elements that form one-element subsets at the beginning; the goal us to perform arbitrary sequences of Union and Find operations in the best time complexity possible. Here a Union works on two disjoint subsets fusing them into one, a Find identifies the subset a certain element belongs to. For an introduction and overview to Union-Find see e.g. [Meh84, GI91]; for recent results see [vKO93].

Efficient implementations of the Union-Find problem use tree data structures to represent sets (i.e. segments in our case), the root of the tree being the representative of the region. To perform an Union operation it suffices to link the two roots of the corresponding tree, creating thereby a new tree. The Find operation identifies the region by finding the root of the tree in an iterative pointer search.

In the general case, the best complexity known has been first obtained by Tarjan, see [Tar75], who has shown that general Union-Find algorithms perform in $O(\alpha(n,m)m)$ where α is a very slowly growing function. and $n < m$ are the amounts of calls to a a Union and Find operation respectively. In [Gus95] it is shown that the Union-Find problem can been solve in linear time on a RAM for special classes of graphs, in particular for d-dimensional grids for fixed d and 8-neighborhood graphs of a 2 dimensional grid.

So theoretically there is no problem since an image can be considered as a 2-dimensional grid and a 3-dimensional image as a 3-dimensional grid. But the algorithms described in [Gus95] are very complex and the application to image segmentation is not trivial. But there exist linear algorithms (see [DST92, FG96]) that perform in linear time in the special case of 2-dimensional image segmentation. The theoretical complexity of these algorithms translate very well in short running-time. The method described here is based on the algorithms given in [FG96]. Let us now present shortly how the Union-Find problem can be applied to image segmentation and more precisely to 3-dimensional image segmentation.

2.2 Application to 3-dimensional Images

Let us first examine the 2D case. Clearly at the beginning of the process of region growing all the pixels form regions of only one element. So you have to choose a scanning strategy (line-by-line, linear quad-tree, random), examine all pairs of pixels of the image, and for each pair decide whether or not to perform the merging of the two regions the pixels belong to. Note that the scanning strategy plays an important role in the process and its outcome. A deterministic strategy will lead to better efficiency, but will influence the shape of the obtained regions in particular line-by-line strategy. An ideal strategy would be a random scan ensuring an homogeneous growing of the regions.

Note that Union-Find segmentation has the particularity to be driven by the pixels. So we can use both local (i.e. relative to the pixels) and global (i.e. relative to the regions) merging criteria, this leads to a contour-region cooperation. Moreover, since we can identify the region each pixel belongs to, such a method, in parallel to the segmentation process, provides us with a connected component labeling.

This method can – in theory – easily be generalized to 3-dimensional images. The same technique applies, just replace pixel by voxel in the above text. But to implement such an approach, there is a problem due to the limitation of memory. A Union-Find data structure is a greedily space consuming. At the beginning, each voxel is a segment, so one may have the structure for a segment for each voxel. And in addition to the grey level of the pixel and the pointer to the father in the tree, this structure must record all the data needed for the merging criteria, such as e.g. number of pixels in the region, sum of the grey levels of all the pixels of the region, etc. For example, in our application the size of such a structure is about 16 bytes. So it is difficult to handle all the 3-dimensional image in memory[2] at once, that is why we have used a 2-parts procedure as described in the following.

3 Description of our Approach

As we have already mentioned, we cannot create a Union-Find structure for the entire 3-dimensional im-

[2]In our application, the size of the image is $512 \times 512 \times 178$, so all the image structure will require about 710 MB of memory!

age at once and load it into memory. So we will proceed slice by slice, from the first to the last. For each slice a weak segmentation will reduce the number of segments to be integrated in the global Union-Find structure, and it will then be possible to feed the entire structure obtained into memory. This scanning that is actually chosen is not the only possible but it is the simplest. For a discussion of the scanning order, see section 5. The general procedure can be summarize as follow:

Algorithm 1: Volume Segmentation

```
  for $i = 1$ to *number of slices* do
1     load slice $S_i$;
2     *weak segmentation* of $S_i$;
3     write $S_i$ to disk;
4     integration of the surviving segments into the
      global data structure;
      if $i \neq 1$ then
5         *volume segmentation* according to $S_{i-1}$;
6         free memory occupied by $S_{i-1}$;
      end
  end
7 save the result slice by slice;
```

The steps 1,2 and 3 form the first part of our algorithm. Its goal is to reduce the amount of data in each slice in order to be able to deal with the entire 3-dimensional image.

The second part consist in the step 4,5 and 6. It realizes the global integration of the data as a whole by managing the global Union-Find structure and creating volumes by the merge of the segments of the slices.

Our algorithm handles two Union-Find structures at two different levels, a ”slice level” and a general level, i.e. the global 3-dimensional structure. The so called ”slice structure” is a classical Union-Find structure for a 2D image as described for example in [FG96]. A record is created for each voxel of the slice and so the slice structure is relatively big compared to the 2D image.

After such a slice has been integrated into the global data structure its segment information is written on disk to free the memory that was used by it and to make the saving slice by slice of the final result possible in the step 7.

The global structure will keep all the segments found after a weak-segmentation of each slice. It is dynamically constructed, slice after slice. So the number of elements of this structure will be equal to the total number of segments extracted by the weak segmentation of all the slice. In our case it is about the same size as an individual slice. So this structure will be not more greedy in memory consumption as a slice structure by itself.

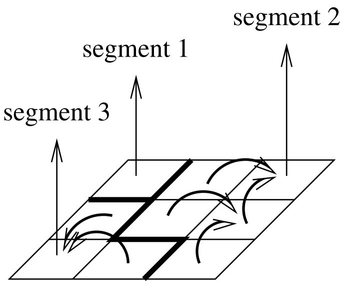


Figure 1: after a weak-segmentation

3.1 Weak Segmentation of Individual Slices

Since we need to reduce the amount of data, we are forced to segment each slice. But we choose parameters of the merging criteria such that the regions obtained are still relatively small. This is what we call a *“weak segmentation”*.

The intention for this is that we want to do a real 3-dimensional segmentation in order to obtain a volume description of the image. By doing a complete segmentation of each slice, we would not take the 3-dimensional information of the image into account. In fact we would fall back into classical methods which segment each slice separately and would have to perform a 3d-reconstruction of the image afterwards.

So we use the segmentation process in each slice as a preprocessing tool performing a filtering on the original image. In keeping regions small we ensure that:

- no abusive merging will be performed on any slice,
- the 3-dimensional information is still pertinent.

For a discussion of the merging criteria that was used in out application and that in fact was able to guarantee the properties we want see the discussion below.

The segmentation process used is fast and provides a connected component labeling (see [FG96]). This last point is important since we will need it when doing the 3-dimensional segmentation.

Now, after the weak segmentation, the number of regions in a slice is substantially reduced. We have the data structure like the one represented in Figure 1.

Before integrate these data into the global structure, we will perform an additional process which will ensure a better complexity in the final algorithm and thus a better performance. This process, called ”flatten the slice” consists in linking all the voxels directly to the segment they belong to. This realizes the component labeling of the slice effectively.

Note that the complexity of such a process is linear in the number of voxel of the slice. Indeed, for each

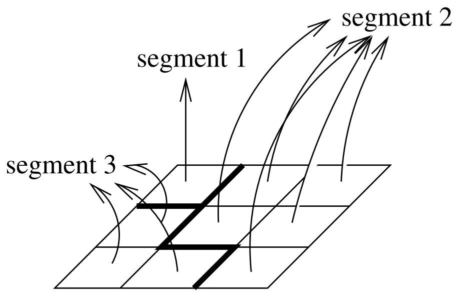


Figure 2: structure obtained after a flatten

voxel we do an iterative pointer search of the root, and on the path followed we link all the voxel directly to the root (path compression). So each edge of the tree which is not link to the root or a leaf, will be visited only one time since after an iterative pointer search all other voxels which are linked to a voxel of this path will find directly the root from it. Each extremal edge (i.e. edge linked to a leaf or the root) will be used one time for each leaf. So the total number of pointer jumps is at most equal to two times the number of edges in the tree which is equal to the number of element of the tree minus one, that's give us a linear time complexity. For a complete discussion see [FG96].

Figure 2 presents the structure obtained.

Note that the structure of a slice is still big since you always have a structure for each voxel. But we will not keep such a structure for each slice of the 3-dimensional image, in fact we will have at most 2 slice at the same time in the memory as it will be shown in the following.

3.2 Global Integration of all the Data

The surviving segments of the precedent part will be introduced in the global data structure. They are now considered as volumes of the image, but they are not totally integrated in the segmentation since we have not yet tried to merge them with the volume previously determined to which they are adjacent. In order to realize this, we always keep the last slice involved in the process in addition to the new one.

Two such structures can be combined to perform a volume segmentation. Thanks to the connected component labeling, one just has to match corresponding voxels to find for each segment those volumes to which it is adjacent. For each pair of matching voxel we decide whether or not merge the segment they belong to (see Figure 3). Clearly after that we don't need the "old slice" anymore and we can free it. Thus, we always have at most 2 slice-structures in memory. In our practical application the weak segmentation of each slice give us about the same number of segments in all the slices

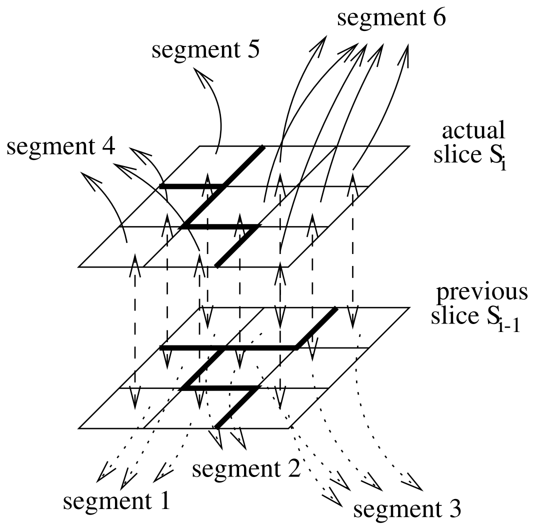


Figure 3: matching of two segmented slices

as the size of one individual slice. In section 4.3 on the next page you will give some information about the memory demand in our application.

At the end of the algorithm we get an Union-Find structure which describes the volume segmentation of the image. We must now save this information. After each weak segmentation, the slice is saved on the disk. For each voxel of the slice we keep the segment identificator it belong to. Now, at the end of the 3-dimension segmentation, we always have all these segments and for each of them we know which volume it is a part of. So we load each slice, and for each voxel replace the segment identificator by the volume identificator.

Thus we have realized a volume segmentation of this image and furthermore we are able to say for each voxel which volume it belongs to.

4 Application to Computer Tomographies

In this section, we present some results obtained by our algorithm on a computer tomography image of a part of a human body, see Figure 6 for a look at the skeleton found inside this part.

4.1 Description of the Data

We have 178 slices of a computer tomography starting at the 8th chest vertebra and going down to the knee. Each slice is a 256 grey levels image of size of 512×512. Figure 4 on the following page shows the 28th slice. It represents a cut through the body at about

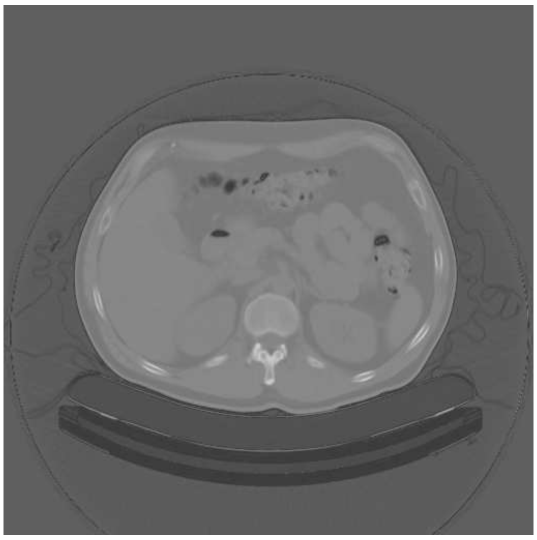


Figure 4: 28th slice of the image

the 11th chest vertebra. In this 2D image one can easily identifies the vertebra, the kidneys at the right and left of it and the liver.

The real distance between two slices is not the same as the one between two pixels of a slice (about 2 times). So our voxels are not cubic but parallelepipedic.

We thus have a $512 \times 512 \times 178$ image, that is 46.661.632 voxels.

4.2 The Merging Criteria

In our application we ensure that the properties needed to have a good performance by choosing a global merging criterion that gives a tight bound on the variance of the regions obtained. Since we also want to avoid that large regions, as the background, arbitrarily eat up small ones, such a guarantee on the variance alone would not suffice. So we use a new criterion that combines these two prerequisites but which to describe extends the limited scope of this paper.

Besides that, we use two other simple local criteria. They are two threshold values which help us to distinguish the background and a great part of the bones. A more pertinent local information would be a 3-dimensional edge, i.e. the presence of local discontinuities as gived by such an algorithm described in [MDMC90]. We intend to include such more sophisticated local criteria in the near future to obtain segmentations that respect boundaries of organs even better.

We use the same criteria for the weak segmentation as for the volume segmentation, but the parameter chosen are more restrictive in order to merge less.

4.3 Statistics (Data Size, Processing Time)

The following table summarizes some statistics. The numbers given are mean values or approximations. The images are saved in a compressed TIFF format.

	time	segments	memory	disk
orig. slice	n/a	260.000	n/a	78 Kb
for one slice (step 1,2,3)	9s.	700	4 MB	24 Kb
volume (step 4,5,6)	4s.	220	2 MB	n/a
"Total" (after step 7)	42 mn.	40.000	10 MB	4 MB

Observe that the total number of volumes is still important. It is only $\frac{1}{3}$ of the total number of segments in all the slices after a weak-segmentation. And the mean size of volumes (without taking into account the background) is about 600 voxels. This is due to our restrictive merging criteria. But the important parts (as bones or organs) are only formed by a very few number of volumes, generally only one. This will be shown in the results presented in the section 4.4. This means that there are numerous small volumes in the rest of the image. The merging of these volumes needs more criteria. We think that the use of a Volume Adjacency Graph (VAG for short) will solve this problem. We discuss this improvement in the conclusion of this paper.

The final segmented 3-dimensional image is about 4 MBytes disk space. Note that the original image takes about 13 MBytes.

The total computation time is about 42 minutes, but if you already have the weak segmented slices, the volume segmentation takes only about 15 minutes, this time includes the disk access, i.e. load and save each of the 178 slices.

4.4 Some Results

The following images are the projection of selected volumes found by our algorithm. We have used a home maid tool to do the rendering, since it is not our specialty, the images are not so good as with a professional rendering software.

The Figure 5 on page 7 presents the 28th slice after the weak-segmentation. Note that here the grey levels are not the mean of the grey levels of the region. In fact we use an RGB format to save the number of the region[3]. The image has been converted in grey levels in order to be printable. The pixels of a same region have the same grey level but due to the limitation on the number of grey levels (256), some different regions

[3]We use the 2 Bytes of the Green and Blue components as an integer.

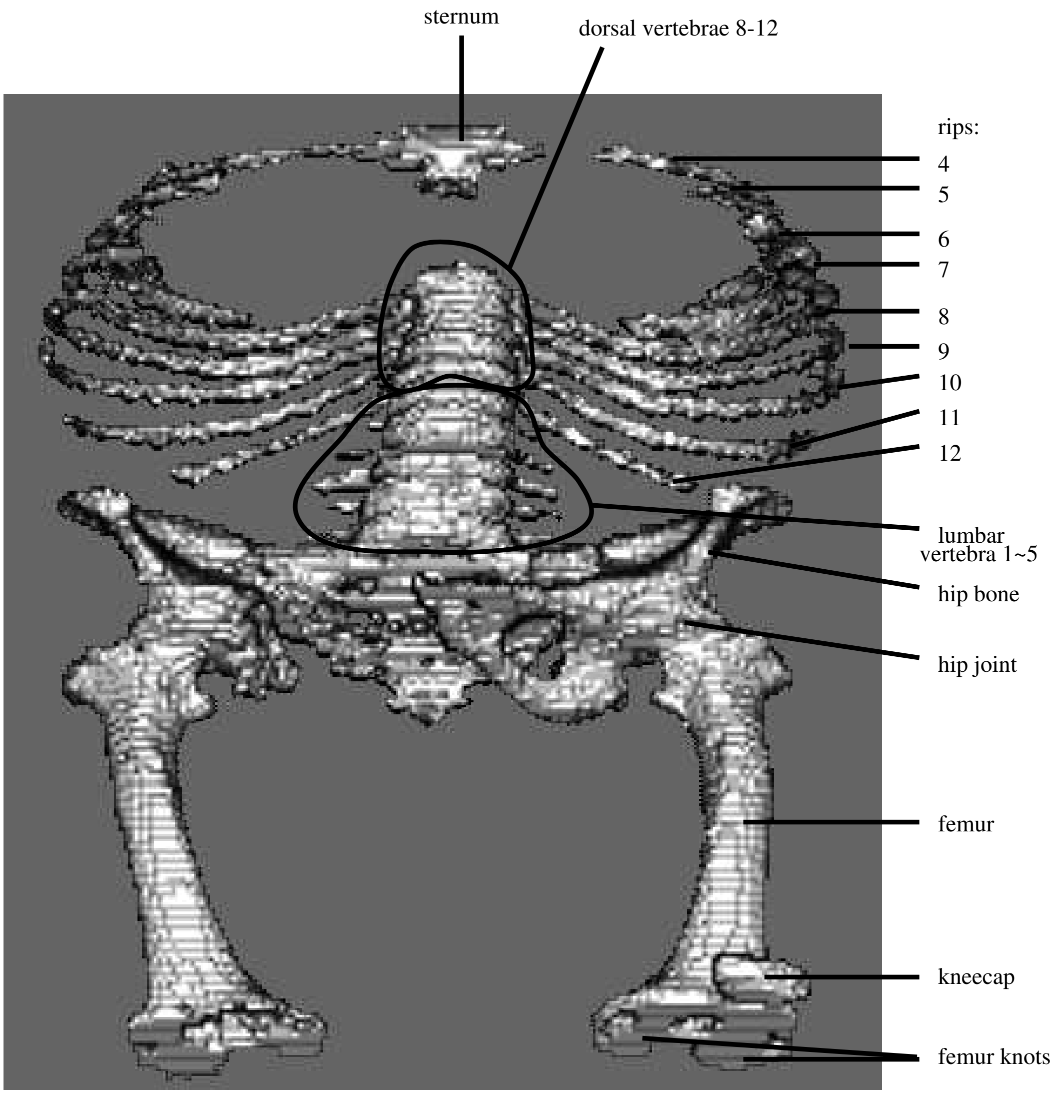


Figure 6: Skeleton found (the body is tilted by 67°)

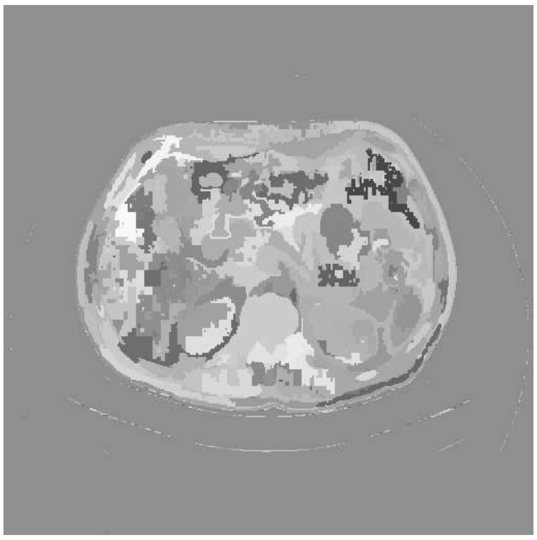


Figure 5: 28th slice after the weak-segmentation

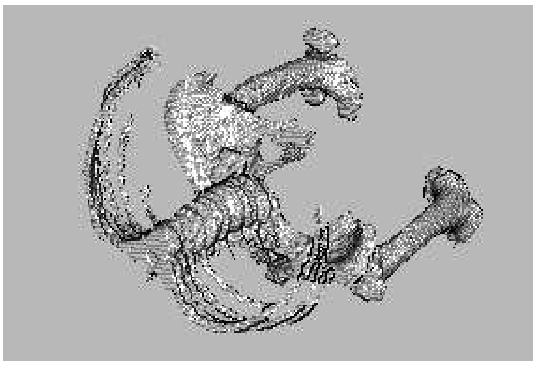


Figure 7: part of the skeleton found in a single volume

can have the same grey level, but they should not be adjacent.

The Figure 6 on the preceding page shows the skeleton found in the image. This image displays a set of 13 volumes. In fact in one single volume we get the most part of all the bones, see Figure 7. Here we have an additional volume for the sternum, since it is not connected to the rips. Moreover the image is only a part of a body, and the cut starts at the 8^{th} rip. So all the rips above, are not in entirely present and thus not connected to the spinal column. So it is impossible to get them into the same volume as the rest of the bones.

Our algorithm succeeds not only in getting the bones but some organs too. For example, Figure 8 shows the stomach which is found as one volume.

Figure 9 shows the two kidneys we have isolated, each in one volume. You can see that some "leaks" occurs in the right kidney, Figure 9.(b). This is due to some little volumes that have been improperly merged.

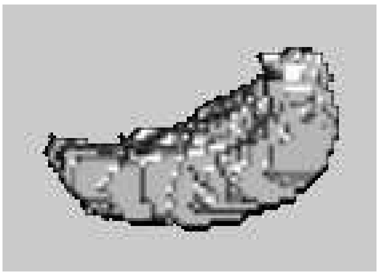
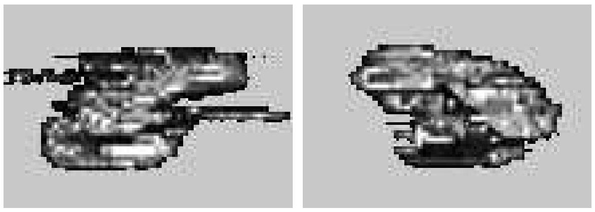


Figure 8: The stomach



(b) right (a) left

Figure 9: The kidneys

We think that the use of more sophisticated local criteria such as a 3-dimensional edge (see e.g. [MDMC90]) will solve this sort of problems.

5 Conclusion and Perspectives

In this paper we have presented a new approach to do 3D segmentation. The algorithm shows good results on a computer tomography image. This approach use a Union-Find data structure in order to deal with the data as a whole. Moreover it provides us with a connected component labeling and we are able to use local and global criteria.

No reconstruction are necessary afterwards, since we have already identified volumes and are able to say for each voxel the volume it belongs to. So to display an organ, for example, it is sufficient to point to a voxel in the image which belongs to it, in order to display the volume representing this organ.

Some improvements are in study, as for example different scanning orders. The slice-by-slice scanning used is the simplest but has a default: the volumes to be merged are disproportionate, especially at the end of the process. We think that a binary-tree order would be better. Moreover this scanning order can lead to a parallelization of the process.

The way to do the merging can also be improved. As we have already mentioned, the local criteria we used are quite simple and it will be easy to add more sophisticated criteria based on local discontinuities, such the one described in [MDMC90]. Note that the computation time will increase since it will be necessary to precompute the map of the 3D-edges.

Another perspective is the use of a Volume Adjacency Graph. This graph can be constructed during the volume segmentation and maintained in parallel to the Union-Find structure. Then after the volume segmentation as described in this paper, we would be able to merge volumes or set of volumes according to the graph. This sort of approach has been already used with success in the 2-dimensional case. So it would be not very difficult to implement it in the 3-dimensional case. The only problem could be the memory usage. But since we have already reduced considerably the memory consuming (only 10 MBytes), it could be done without problem.

Aknowledgement

We are grateful to people of the SFB Hyperthermie supplying us with the computer tomographie that forms the base of the pratical results presented in this paper.

References

[AAF95] Ehoud Ahronovitz, Jean-Pierre Aubert, and Christophe Fiorio. The star-topology: a topology for image analysis. In 5th *Discrete Geometry for Computer Imagery, Proceedings*, pages 107–116. Groupe GDR PRC/AMI du CNRS, september 1995.

[AFH81] E. Artzy, G. Frieder, and G.T. Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. *Computer Graphics and Image Process.*, 15:1–24, 1981.

[DST92] Michael B. Dillencourt, Hanan Samet, and Markku Tamminen. A general approach to connected-component labeling for arbitrary image representation. *J. of the Association for Computing Machinery*, 39(2):253–280, April 1992.

[FG96] Christophe Fiorio and Jens Gustedt. Two linear time Union-Find strategies for image processing. *Theoretical Computer Science*, 154:165–181, 1996.

[GI91] Zvi Galil and Giuseppe F. Italiano. Data structures and algorithms for disjoint set union problems. *ACM Computing Surveys*, 23(3):319–344, September 1991.

[GL93] Muhittin Gökmen and Ching-Chung Li. Edge detection and surface reconstruction using refined regularization. *IEEE trans. Pattern Analysis and Machine Intelligence*, 15(5):492–499, may 1993.

[Gus95] Jens Gustedt. Efficient union-find for planar graphs and other sparse graph classes. Preprint Reihe Mathematik 476/1995, TU Berlin, 1995.

[Her83] G.T. Herman. Special issue on computerized tomography. *IEEE Proc.*, 71:291–435, 1983.

[KKM90] Efim Khalimsky, Ralph Kopperman, and Paul R. Meyer. Boundaries in digital planes. *J. of Applied Mathematics and Stochastic Analysis*, 3(1):27–55, 1990.

[Kov89] V.A. Kovalevsky. Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing.*, 46:141–161, 1989.

[KR89] T.Y. Kong and A. Rosenfeld. Digital topology: introduction and survey. *Computer Vision, Graphics, and Image Processing.*, 48:357–393, 1989.

[KU92] T.Y. Kong and J.K. Udupa. A justification of a fast surface tracking algorithm. *CVGIP: Graphical Models and Image Processing*, 54:162–170, 1992.

[Lat93] L. Latecki. Topological connectedness and 8-connectedness in digital pictures. *CVGIP: Image Understanding*, 57(2):261–262, 1993.

[MBA93] Grégoire Malandain, Gilles Bertrand, and Nicolas Ayache. Topological segmentation of discrete surfaces. *Int. J. of Computer Vision*, 10(2):182–197, 1993.

[MDMC90] O. Monga, R. Deriche, G. Malandain, and J.P. Cocquerez. Recursive filtering and edge closing: two primary tools for 3d edge detection. In *Proc. 1st European Conference on Computer Vision*, volume 427, Nice, France, April 1990. Lecture Notes in Computer Science, Springer Verlag: New York.

[Meh84] Kurt Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*. Springer, 1984.

[RKW91] A. Rosenfeld, T.Y. Kong, and A.Y. Wu. Digital surfaces. *CVGIP: Graphical Models and Image Processing*, 53(4):305–312, July 1991.

[Tar75] Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. of the Association for Computing Machinery*, 22:215–225, 1975.

[Udu94] J.K. Udupa. Multidimensional digital boudaries. *CVGIP: Graphical Models and Image Processing*, 54(4):311–323, July 1994.

[UH91] J.K. Udupa and G.T. Herman. *3D Imaging in Medicine*. CRC Press, Boca Raton, FL, 1991.

[vKO93] Marc J. van Kreveld and Mark H. Overmars. Union-copy structures and dynamic segment trees. *J. of the Association for Computing Machinery*, 40(3):635–652, July 1993.

Reports from the group

“Algorithmic Discrete Mathematics”

of the Department of Mathematics, TU Berlin

515/1996 *Christophe Fiorio and Jens Gustedt:* Volume Segmentation of 3-dimensional Images

514/1996 *Martin Skutella:* Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem

506/1996 *Rolf H. Möhring and Markus W. Schäffter:* A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

505/1996 *Rolf H. Möhring and Dorothea Wagner:* Combinatorial Topics in VLSI Design: An Annotated Bibliography

504/1996 *Uta Wille:* The Role of Synthetic Geometry in Representational Measurement Theory

502/1996 *Nina Amenta and Günter M.Ziegler:* Deformed Products and Maximal Shadows of Polytopes

498/1996 *Ewa Malesinska, Alessandro Panconesi:* On the Hardness of Allocating Frequencies for Hybrid Networks

496/1996 *Jörg Rambau:* Triangulations of Cyclic Polytopes and higher Bruhat Orders

489/1995 *Eva Maria Feichtner and Dmitry N. Kozlov:* On Subspace Arrangements of Type $\mathcal{D}$

483/1995 *Rolf H. Möhring and Markus W. Schäffter:* Approximation Algorithms for Scheduling Series-Parallel Orders Subject to Unit Time Communication Delays

478/1995 *Sven G. Bartels:* The complexity of Yamnitsky and Levin’s simplices algorithm

477/1995 *Jens Gustedt, Michel Morvan and Laurent Viennot:* A compact data structure and parallel algorithms for permutation graphs, to appear in : Nagl et al., editors, *Graph-Theoretic Concepts in Computer Science*, Procceedings of the 20th International Workshop WG ’95.

476/1995 *Jens Gustedt:* Efficient Union-Find for Planar Graphs and other Sparse Graph Classes

475/1995 *Ross McConnell and Jeremy Spinrad:* Modular decomposition and transitive orientation

474/1995 *Andreas S. Schulz:* Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP–Based Heuristics and Lower Bounds

473/1995 *Günter M. Ziegler:* Shelling Polyhedral 3-Balls and 4-Polytopes

472/1995 *Martin Henk, Jürgen Richter-Gebert and Günter M. Ziegler:* Basic Properties of Convex Polytopes

471/1995 *Jürgen Richter-Gebert and Günter M. Ziegler:* Oriented Matroids

465/1995 *Ulrich Betke and Martin Henk:* Finite Packings of Spheres

463/1995 *Ulrich Hund:* Every simplicial polytope with at most $d+4$ vertices is a quotient of a neighborly polytope, to appear in Discrete & Computational Geometry

462/1995 *Markus W. Schäffter:* Scheduling with Forbidden Sets

461/1995 *Markus W. Schäffter:* Drawing Graphs on Rectangular Grids with at most 2 Bends per Edge

458/1995 *Ewa Malesińska:* List Coloring and Optimization Criteria for a Channel Assignment Problem

447/1995 *Martin Henk:* Minkowski's second theorem on successive minima

441/1995 *Andreas S. Schulz, Robert Weismantel, Günter M. Ziegler:* 0/1–Integer Programming: Optimization and Augmentation are Equivalent, appeared in Paul Spirakis (ed.): Algorithms – ESA '95, Lecture Notes in Computer Science 979, Springer: Berlin, 1995, pp. 473-483

440/1995 *Maurice Queyranne, Andreas S. Schulz:* Scheduling Unit Jobs with Compatible Release Dates on Parallel Machines with Nonstationary Speeds, appeared in Egon Balas and Jens Clausen (eds.): Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 920, Springer: Berlin, 1995, pp. 307-320

439/1995 *Rolf H. Möhring, Matthias Müller-Hannemann:* Cardinality Matching: Heuristic Search for Augmenting Paths

436/1995 *Andreas Parra, Petra Scheffler:* Treewidth Equals Bandwidth for AT-Free Claw-Free Graphs

432/1995 *Volkmar Welker, Günter M. Ziegler, Rade T. Živaljević:* Comparison Lemmas and Applications for Diagrams of Spaces

431/1995 *Jürgen Richter-Gebert, Günter M. Ziegler:* Realization Spaces of 4-Polytopes are Universal, to appear in *Bulletin of the American Mathematical Society*, October 1995.

430/1995 *Martin Henk:* Note on Shortest and Nearest Lattice Vectors

429/1995 *Jörg Rambau, Günter M. Ziegler:* Projections of Polytopes and the Generalized Baues Conjecture

428/1995 *David B. Massey, Rodica Simion, Richard P. Stanley, Dirk Vertigan, Dominic J. A. Welsh, Günter M. Ziegler:* Lê Numbers of Arrangements and Matroid Identities

408/1994 *Maurice Queyranne, Andreas S. Schulz:* Polyhedral Approaches to Machine Scheduling

407/1994 *Andreas Parra, Petra Scheffler:* How to Use the Minimal Separators of a Graph for Its Chordal Triangulation

401/1994 *Rudolf Müller, Andreas S. Schulz:* The Interval Order Polytope of a Digraph, appeared in Egon Balas and Jens Clausen (eds.): Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 920, Springer: Berlin, 1995, pp. 50-64

396/1994 *Petra Scheffler:* A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree-width

394/1994 *Jens Gustedt:* The General Two-Path Problem in time $O(m \log n)$, extended abstract

393/1994 *Maurice Queyranne:* A Combinatorial Algorithm for Minimizing Symmetric Submodular Functions

392/1994 *Andreas Parra:* Triangulating Multitolerance Graphs

390/1994 *Karsten Weihe:* Maximum (s,t)–Flows in Planar Networks in $O(|V| \log |V|)$ Time

386/1994 *Annelie von Arnim, Andreas S. Schulz:* Facets of the Generalized Permutahedron of a Poset, to appear in *Discrete Applied Mathematics*

383/1994 *Karsten Weihe:* Kurzeinführung in C++

377/1994 *Rolf H. Möhring, Matthias Müller-Hannemann, Karsten Weihe:* Using Network Flows for Surface Modeling

376/1994 *Valeska Naumann:* Measuring the Distance to Series-Parallelity by Path Expressions

375/1994 *Christophe Fiorio, Jens Gustedt:* Two Linear Time Union-Find Strategies for Image Processing

374/1994 *Karsten Weihe:* Edge–Disjoint (s,t)–Paths in Undirected Planar graphs in Linear Time

373/1994 *Andreas S. Schulz:* A Note on the Permutahedron of Series–Parallel Posets, appeared in *Discrete Applied Mathematics* 57 (1995), pp. 85-90

371/1994 *Heike Ripphausen–Lipa, Dorothea Wagner, Karsten Weihe:* Efficient Algorithms for Disjoint Paths in Planar Graphs