Flexible mesh generation for segmented 2D and 3D images containing multiple materials

Guntram Berti, C&C Research Laboratories, NEC Europe Ltd.
St. Augustin, Germany

Abstract

We present a hierarchical, dimension-independent approach to mesh generation from segmented images, based on adaptive spacetree subdivision. The central algorithm is extended to generate non-uniform tetrahedral meshes or hybrid meshes containing hexahedra, pyramids, and tetrahedra. We present a volumetric marching tetrahedra algorithm permitting vertices to lie on the separating surface, enabling d-linear separating functions and vertex snapping. This algorithm is extended to multiple material, and a new separating function based on solid angles is proposed. Finally, smoothing in the presence of internal boundaries is discussed.

1 Introduction

Building a geometric representation (that is, a volume mesh) from a medical image is a necessary prerequisite for many interesting numerical analyzes like computational structural mechanics or fluid dynamics, which use finite element or finite volume algorithms [1].

A segmented (or labeled) medical image can already be regarded as such a geometric model by viewing the individual voxels as cells (elements) of a (subset of a) structured mesh. However, such a mesh gives us a fixed resolution and a non-smooth axis-aligned boundary approximation. While this is appropriate for some problems, often one needs meshes that are better adapted to the geometry, for instance by providing smooth approximations of the boundary or permitting local cell spacing in order to control resolution and problem size.

In this paper, we show how to refine the basic structured mesh represented by a multi-material segmented image into a non-uniform unstructured mesh with smooth internal and external boundaries. The algorithms are essentially dimension independent, except some parts of the marching simplices method which have to be adapted for each image dimension. The algorithms can be controlled in a flexible way, allowing one e.g. to treat different materials, interfaces or locations differently.

The meshing paradigm used in this paper we have called *volume-oriented meshing*, as it directly operates on a volumetric representation of the geometry, and a boundary surface approximation is created in the course of the volume meshing process. It has been used by a number of researchers in the context of medical image data, for instance [2, 3]. Similar approaches have been used at least since 1983 [4].

A competing meshing approach is *surface-oriented meshing*, where first a surface mesh is generated and used as a starting point for volume meshing. Well-known examples are Delaunay triangulation and the advancing front method, see [5] for an overview.

We feel that in the context of geometries represented by segmented volume images, the volume-oriented approach has some advantages over the surface oriented approach. It is fast, stable, and easy to implement. It also seems to be more natural for volumetric data that do not contain an explicit and unambiguous boundary definition. The latter is found as a by-product. whereas the surface-oriented approach needs a surface mesh as starting point. However, in the volume-oriented approach, special attention has to be paid to mesh quality near the boundary.

This paper is composed as follows: In the first section, we give an overview on the work flow of the volumetric meshing approach, going from segmented images to hierarchical spacetrees to non-uniform simplicial or hybrid meshes and finally to smoothed boundary approximations. Here, we introduce a generalization of the volumetric marching simplices algorithm to permit vertices lying on the separating surface, and discuss smoothing in the presence of internal boundaries. In Section 3, we present an extension of marching simplices to the case of multiple materials. Finally, we discuss the methods presented and point to further research.

2 The Volume-oriented Meshing Pipeline

The volume-oriented meshing approach operates in stages (cf. Fig. 1), described briefly in the sequel. More detail can be found in [6]. We start with a segmented d-dimensional image, which has been obtained by assigning material labels to the voxels of a raw (gray-scale) image. Here, we do not consider the segmentation process itself part of the meshing. We only note that the segmented image may result in a loss of geometric precision, which may make it advantageous to use the initial raw image as additional source of information in later meshing stages, for instance for boundary approximation.

Spacetree generation

The next step transforms the segmented image, which may be regarded as a Cartesian grid, into a hierarchical structure called *spacetree* [7] (known as quadtree in 2D and octree in 3D). We may use different refinement patterns besides the classical bisection pattern. This transformation allows to adapt mesh size locally, e.g. according to material and interfaces. The spacetree may be balanced allowing arbitrary levels of imbalance. See Fig. 2 for examples.

Non-uniform conforming meshing So far, the mesh consists of axis-aligned cubes of different sizes (Fig. 1 (b)). While this may already be usable for some simulation algorithms [7], most numerical methods require conforming interfaces, i.e. without so-called hanging nodes. Therefore, we now tessellate the spacetree in order to obtain a conforming mesh, consisting either of simplices only or containing simplices, pyramids and cubes. In order to achieve a conforming tessellation into simplices, we apply a recursive midpoint-rule: First, each boundary facet of a cube is subdivided into $d-1$-dimensional simplices, and then, each boundary simplex is connected to the midpoint of the cube, creating a d-dimensional simplex. For cubes which are regular (that is, have no hanging nodes on their boundary), we use a template tessellation which produces much less simplices. See Fig. 1 (c) for a 2D example and Fig. 3 for a 3D example. This approach can easily extended to hybrid mesh generation, leaving regular cubes and generating pyramids as transition elements between cubes and simplices.

Marching simplices The resulting mesh is conforming, but still lacks boundary smoothness. The basic idea of marching simplices is to define a separating surface between two regions and split each simplex intersecting this surface. The problems which have to be solved are the following:

1. How to ensure a conforming triangulation after splitting?

2. How to define a smooth separating surface?

3. How to cope with vertices very close to or on the separating surface?

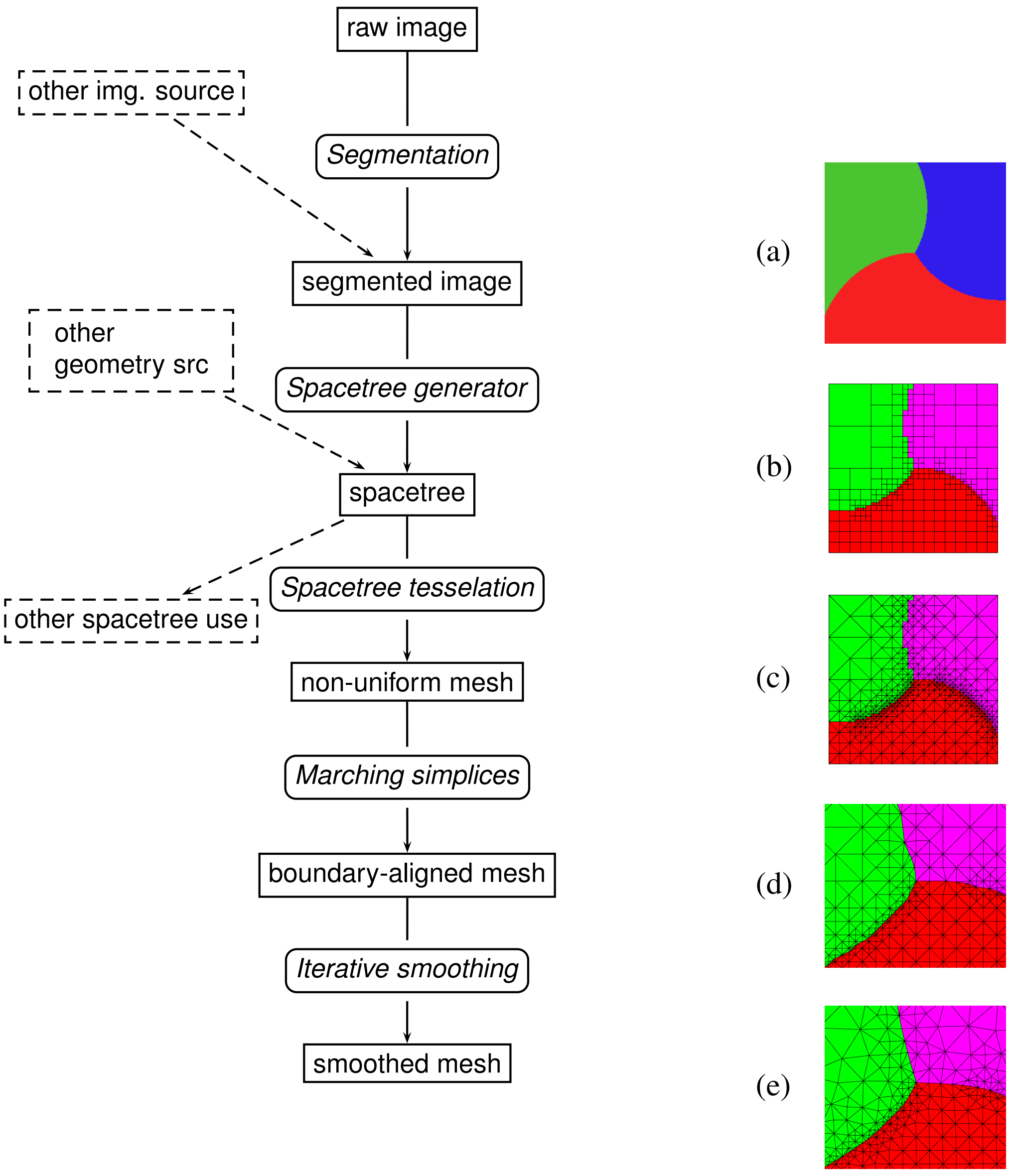

Figure 1: The volume-oriented mesh generation pipeline. (a) The segmented image, consisting of 3 materials. (b) First a spacetree is constructed from the image. Here, each material and each interface has its own resolution. (c) Next, a non-uniform simplicial (or hybrid) mesh is generated, by using midpoint-based subdivision. Most of the uniformly sized cells in the bottom region are subdivided by a template. (d) A variant of the volumetric marching simplices algorithm is applied to obtain smoother boundaries (zoom). (e) Finally, additional iterative smoothing gives better boundaries (zoom).

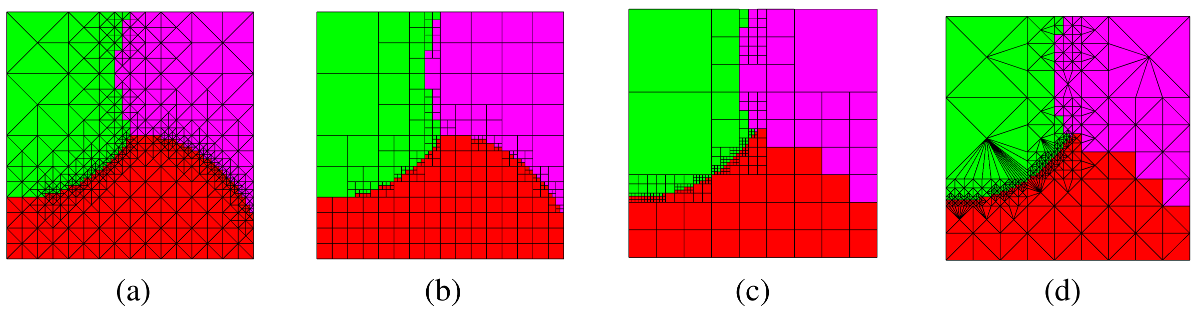

Figure 2: Different options for spacetree generation. (a) 2×2 pattern, 1 level balancing. (b) 2×2 pattern, 2 levels balancing. (c) 3×3 pattern, no balancing. The interfaces and materials are meshed with different resolutions. (d) Non-uniform mesh resulting from (c). Imbalanced spacetrees result in skinny simplices.

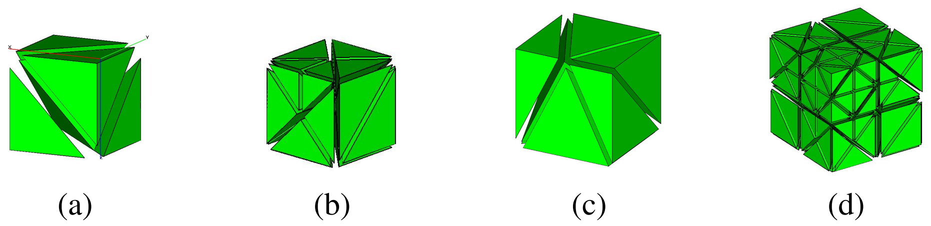

Figure 3: The non-uniform meshing algorithm in 3D. (a) Template subdivision of a cube (regular spacetree cell) into 5 tetrahedra. (b) Midpoint subdivision of a non-regular spacetree cell. The left and the top facets are subdivided according to the midpoint rule, whereas the subdivision of the right facet uses a template. The rule for choosing the diagonal uses only information associated to the facet in order to guarantee a consistent triangulation of two adjacent cells. (c) Extended template for the case in (b), which avoids the generation of additional nodes. (d) Full 3D example.

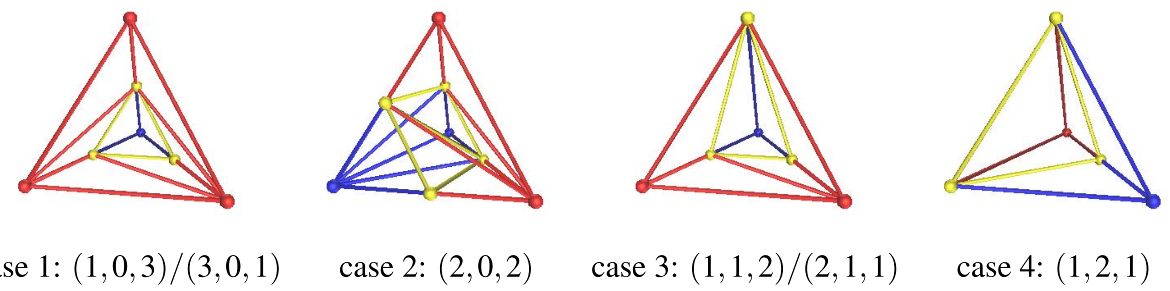

Figure 4: We can classify simplices by a triple (n_+, n_0, n_-), meaning the number of vertices v with $f(v) > 0$, $f(v) = 0$, and $f(v) < 0$. There are 4 essential cases for vertex signs 3D. Blue (dark gray) is positive, red (medium gray) is negative, yellow (light gray) is zero. The new triangulation edges are also shown, assuming the "smallest" vertex is always the right front one. The real intersection points are found be linear interpolation of f along each edge.

4. How to cope with multiple materials?

In [3], the authors present a volumetric marching tetrahedra algorithm which solves problem 1 by using a vertex ordering rule to disambiguate the splitting process.

A general approach to obtain a separating surface is to define a scalar function f on the image domain and use the zero isosurface of f. Probably the simplest way to define f is to use d-linear interpolation of the binary voxel values. This produces visible staircase artefacts (cf. Fig. 6 left), which can be smoothed afterwards. Other possibilities are the use of the gray values in the original raw image, or some higher-order interpolation.

A problem may be vertices which are close to or even on the separating surface. For example, when using d-linear interpolation, we obtain only a finite number of different function values for f; thus, a number of vertices lie on the surface. Also, for arbitrary separating functions, it may happen that a vertex is very close to the surface, resulting in very thin simplices. In such a case, it is preferable to move ("snap") the vertex to the surface. Therefore, the simplex splitting patterns in [3] has been extended to handle the "degenerate" case of vertices lying on the surface.

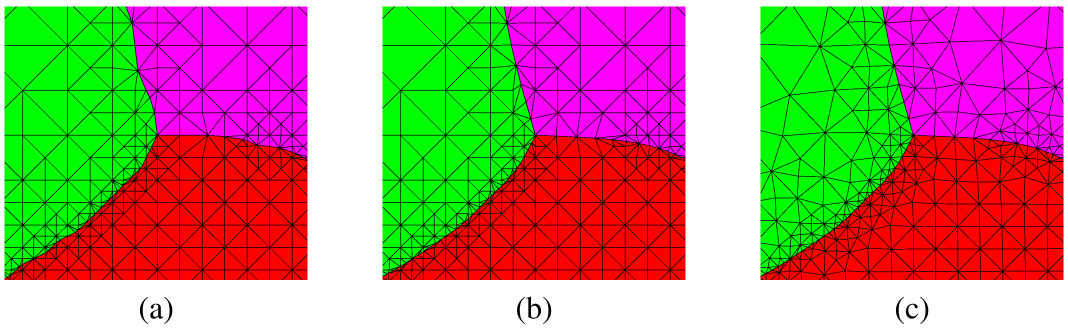
(a) (b) (c)

Figure 5: Different stages of surface smoothing: (a) Only marching simplices, no iterative smoothing (b) Smoothing on material interfaces only (c) Smoothing on material interfaces, followed by volumetric smoothing. Vertices on material interfaces are kept fixed during volumetric smoothing.

Finally, the separating surface approach does not directly extend from the binary case to the multi-material case. In the following section, we present a simple iterative technique using a new separating function which can be derived from arbitrary unstructured meshes.

Mesh smoothing Depending on the smoothness of the separating surface used, the boundaries of the mesh produced by the marching simplices algorithm still contain visible staircase artefacts stemming from the non-smooth approximation of the geometry in the underlying segmented image. These artefacts can be reduced significantly by smoothing the boundary surfaces. The basic idea here is to use Laplacian smoothing of boundary vertices. However, Laplacian smoothing will shrink a surface towards its barycenter [8]. Thus, vertex movement has to be constrained, or at least the shrinking effect has to be reduced by a clever use of parameters based on signal processing theory [8].

Smoothing the surface may have an adverse effect on the quality of incident volume element. To mitigate this effect, we first smooth the surface vertices, and then perform a volumetric Laplacian smoothing, keeping surface vertices fixed. In 3D, we first perform smoothing on material edges, that is, on vertices where 3 interior surfaces meet, then surface smoothing, and finally volume smoothing. For each stage, only neighbor vertices in the same subset (material edge, interface or interior) are considered; vertices in lower-dimension subsets (which have already been smoothed) are kept fixed (cf. Fig. 5).

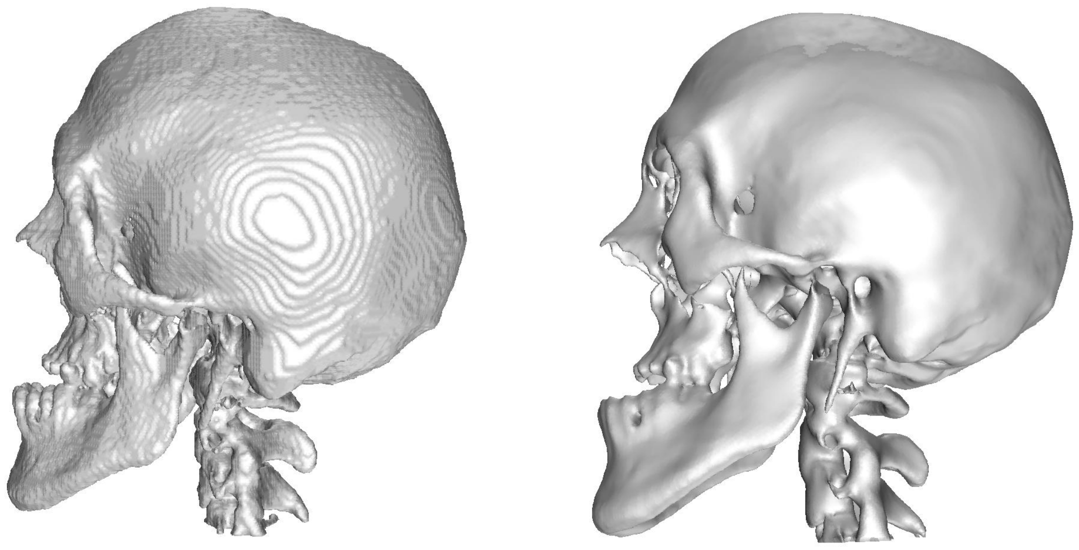

Figure 6: Surface mesh of a human skull. Left: Result of the marching simplices with 3-linear interpolation. The staircase artefacts resulting from the binary classification are still clearly visible. Right: Same mesh after additional smoothing has been applied.

3 Multiple Materials and Solid Angles as Separating Function

The approaches presented in the preceding section to not generalize directly to the case where $n > 2$ materials are present. If we view an image as a Cartesian grid with the voxels playing the role of cells, we may classify the vertices according to the number of different materials present in the incident cells. Two-material point lie on a *material surface* and can be handled by the algorithms presented before. However, three-material points lie on *material edges* and are more difficult to deal with. A basic idea is to replace the scalar separating function f with a vector $(f_1,\ldots,f_n)$ of material inclusion ratios or probabilities summing up to 1.

Several possibilities to handle multiple materials were suggested. Nielson and Franke [9] present a simple extension of the two-material scheme [3]. They assume a labeling of the vertices with an (integer or fractional) inclusion probability. The interfaces are reported to lack smoothness, especially in irregular meshes. This may be due to the use of integer probabilities stemming from assigning vertices to materials.

Bonnell et al [10] consider material interface reconstruction in the context of computational fluid dynamics. They interpret material-inclusion ratios on vertices as barycentric coordinates in a higher-dimensional space and project the intersection with a corresponding Voronoi tessellation back to the physical space to obtain material boundaries. However, they do not treat the splitting of volume cells. Hege et al [11] extend the Marching Cubes algorithm to handle multiple materials. They assign a material-inclusion ratio vector to vertices (assumed integer for efficiency, that is, the probability f_m equals 1 for some m) and use a sub-voxel Cartesian subdivision plus a detailed case analysis to subdivide each cube (voxel). Thus, their method is restricted to Cartesian grids.

None of the research cited above considers the case of separating surfaces passing through vertices.

Our basic idea for generalizing the marching simplices algorithm to multiple materials is to apply it iteratively: If we have a set of material $M = \{m_1,\ldots,m_n\}$, we first form the sets

$M_1 = \{m_1\}$ and $M_2 = M \setminus M_1$ and apply the two-material algorithm. We repeat this for each material m_j instead of m_1. Thus, each material interface is visited twice.

If we use the iterated approach, we have to operate on an unstructured mesh. In order to get a smooth separating function defined for the internal boundaries of an unstructured mesh partitioned into several materials, we have to assign the material inclusion probabilities for each vertex to each material.

When a vertex is surrounded by cells of a single material m, its inclusion probability $f_m(v)$ is 1 with respect to this material and 0 with respect to all other materials. When a vertex is surrounded by cells of different materials, we use the *solid angle* of these cells to measure the contribution of each material. We define the solid angle ratio $a_c(v)$ of a vertex v of a polytope c as the ratio $S_{\text{cone}}/S_{\text{sphere}}$ of the surface S_c which the cone at that vertex cuts out on a small sphere surrounding the vertex to the surface S_s of the sphere. We have $0 \leq a_c(v) \leq 1$ and

$$\sum_{\text{Cells } c \text{ incident to } v} a_c(v) = 1 \tag{1}$$

for interior vertices v. Setting

$$f_m(v) = \sum_{\text{Cells } c \text{ of material } m \text{ incident to } v} a_c(v) \tag{2}$$

we use the iso-value $f_m = 1/2$ to separate M_1 from M_2 in the mth iteration.

When using the iterative approach, a problem occurs when splitting a cell which is part of the single material set M_1. In this case, one part of the split cell belongs to the multi-material set M_2, and the question arises which material should be assigned to the new cell(s) of this part. We propose a simple but sufficient approach which uses the material of the majority of cell neighbors, weighted by the corresponding facet area.

A problem with the iterated approach is its dependence on the ordering of the materials, which may lead to non-optimal results at material edges in 3D (three or more material interfaces meeting). Also, while it works quite well in 2D (cf. Fig. 1 (d) and (e)), there are some problems with material edges in 3D. Perhaps a combination of the template approach in [9] with our smoother vertex probabilities would be appropriate. However, in order to continue the special treatment of “degenerate” vertices (lying on material surfaces or even edges), a considerable number of cases would have to be treated. A complete treatment of configurations for arbitrary inclusion properties entail a large number of cases, as the maximum probability for m material may change m times along an edge. Giving a special treatment to “degenerate” vertices (lying on material surfaces or even edges) would further increase the number of cases. Therefore, a consistent way of reducing the needed cases is preferable.

4 Conclusion

We have outlined a flexible approach for volume-oriented mesh generation, applying different independent steps in a pipelined workflow. The non-uniform meshing algorithm and the marching simplices algorithm with d-linear separating function can guarantee a minimum mesh quality [6]. Subsequent mesh smoothing allows to get smoother boundaries, yet the impact on mesh quality has to be investigated more thoroughly.

The iterative extension of the volumetric marching simplices algorithm to multiple materials looks promising, but more work has to be put into the issue of material edges where three or more material interfaces meet. In the future, we will investigate additional subdivision templates for the multi-material case.

Acknowledgments

I thank my current and former colleagues at NEC C&C Research Laboratories, Jochen Fingberg, Jens Georg Schmidt and Ulrich Hartmann, for their ongoing support and constructive discussions.

An earlier version of the meshing software was implemented by U. Hartmann and F. Kruggel at the Max-Planck-Institute of Cognitive Neuroscience at Leipzig and continued by U. Hartmann and the author at C&C Research Laboratories and is available at [12].

A CT image used for the meshes displayed in Fig. 6 was kindly provided by Dr. Hierl, Department of Oral and Maxillofacial Surgery, University Clinic Leipzig, and was segmented using software from F. Kruggel, MPI CNS, Leipzig.

The author was partially supported by the European Union under research grants IST-1999-10378 and IST-2001-37153 (IST projects SimBio and GEMSS).

References

[1] Jens Georg Schmidt, Guntram Berti, Jochen Fingberg, Gert Wollny, and Junwei Cao. A finite-element based tool chain for the planning and simulation of maxillo-facial surgery. In P. Neittaanmäki, T. Rossi, K. Majava, and O. Pironneau, editors, *European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2004*. University of Jyväskylä, Department of Mathematical Information Technology, 2004.

[2] R. Müller and P. Rüegsegger. Three-dimensional finite element modelling of non-invasively assessed trabecular bone structure. *Med. Eng. Phys*, 17(2):126–133, 1995.

[3] G. M. Nielson and Junwon Sung. Interval volume tetrahedrization. In *IEEE Visualization '97 (VIS '97)*, pages 221–228, Washington - Brussels - Tokyo, October 1997. IEEE.

[4] Mark A. Yerry and Mark S. Shephard. A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications*, 3(1):39–46, January — February 1983.

[5] Mark S. Shephard, Hugues L. de Cougny, Robert M. O'Bara, and Mark W. Beall. *Handbook of grid generation.* CRC Press, 1999.

[6] Guntram Berti. Image-based unstructured 3d mesh generation for medical applications. In P. Neittaanmäki, T. Rossi, K. Majava, and O. Pironneau, editors, *European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2004*. University of Jyväskylä, Department of Mathematical Information Technology, 2004.

[7] Michael Bader, Hans-Joachim Bungartz, Anton Frank, and Ralf Mundani. Space tree structures for pde solution. In Peter M. A. Sloot, C. J. Kenneth Tan, Jack J. Dongarra, and Alfons G. Hoekstra, editors, *Proceedings of ICCS 2002, part 3*, volume 2331 of *LNCS*. Springer, 2002.

[8] Gabriel Taubin. Geometric signal processing on polygonal meshes. In *Eurographics 2000 State of the Art Report*, 2000.

[9] Gregory Nielson and Richard Franke. Computing segmented volumes. In Hagen, Nielson, and Post, editors, *Proceedings of Dagstuhl Seminar, Scientific Visualization 1997*, pages 251–256, 1997.

[10] Kathleen S. Bonnell, Mark A. Duchaineau, Daniel A. Schikore, Bernd Hamann, and Kenneth I. Joy. Material interface reconstruction. In *IEEE Transactions on Visualization and Computer Graphics*. IEEE, 2003.

[11] H.-C. Hege, M. Seebaß, D. Stalling, and M. Zöckler. A generalized marching cubes algorithm based on non-binary classifications. Technical Report SC-97-05, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 1997.

[12] The SimBio-Vgrid mesh generator homepage. `http://www.ccrl-nece.de/vgrid`, 2004.