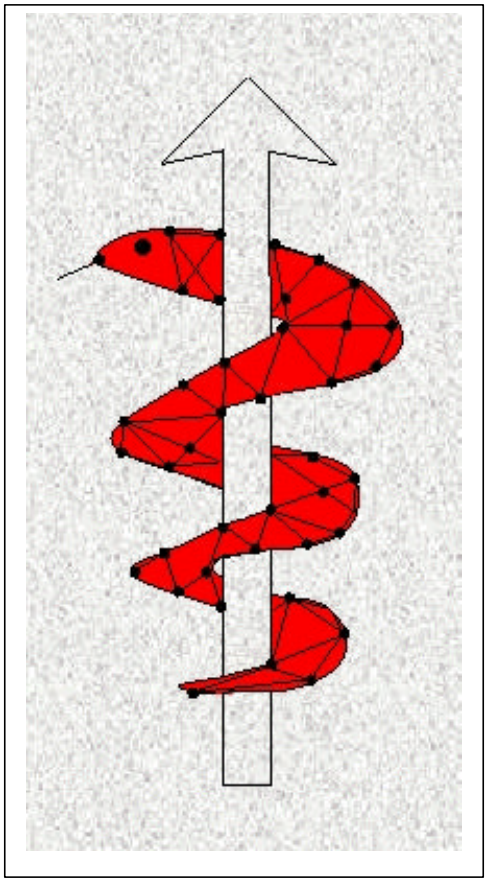
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D1.2a
Mesh Generation Design Report

Status:	Final
Version:	1.0
Security:	Public
Responsible:	NEC
Authoring Partners:	NEC, MPI, USFD, ESI

Release History

Version	Date
0.1	31.08.00
1.0	29.09.00

The SimBio Consortium :
NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
K.U. Leuven R&D – Belgium
ESI Group – France
Smith & Nephew - UK
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
CNRS-DR18 – France
Sheffield University – UK

1. Introduction

The finite element (FE) method for the numerical solution of partial differential equations (PDEs) has a large field of applications in its classical domain, the engineering sciences. Recently, this method gained interest in the medical field.

The finite element method is used for

- locating electromagnetical sources in the human brain [buc97],
- simulating the mechanical response of the head [hart98],
- for predicting the human facial shape after maxillofacial surgery [kee98] and
- for optimising the design of meniscal prostheses for the human knee [ho98, ho99]

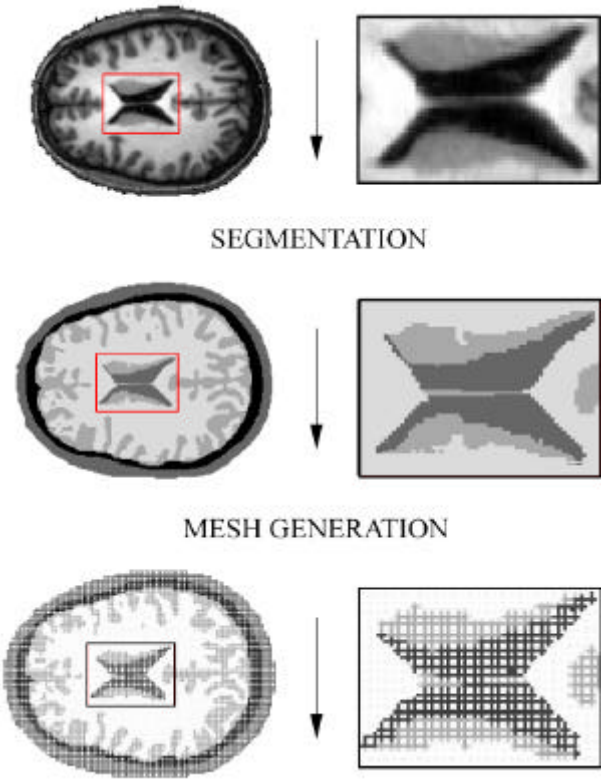


Figure I: From MRI data to finite elements: By means of image processing a segmented image is obtained which can be used as starting point for the mesh generation process.

FE models for medical problems are frequently based on tomographic examinations from magnetic resonance (MR) or X-ray scanners. A proper segmentation of the medical images yields objects (i.e. bone, soft tissue, anatomical structures) as homogeneous regions to which tissue properties (e.g. electrical conductivity or mechanical elasticity) are assigned (see Design Report of ST1.1). Then, a finite element description of those objects has to be introduced by a mesh generation procedure (see Figure 1).

By considering the connectivity of the elements and their tissue properties, a linear equation system is set up and solved by appropriate numerical techniques (see Design Report of WP3). The solution of an FE calculation yields a physical property depending on the underlying PDE. Spatial displacements or electrical potentials at the mesh nodes are typical results of a finite element analysis in the medical field.

The problem of 3D mesh generation is a current research topic within the field of computational geometry. All the solutions proposed can be roughly subdivided into two classes.

The most common approach to create a 3D mesh starts with a boundary representation of a polyhedral object [gei93, boe94, joe91, joe92, lo91a, lo91b]. During the mesh generation, often so-called "Steiner points" are set in the interior of the object. A Delaunay tesselation is then applied to generate the finite elements [ede87, cig94]. Frequently, a post-processing step is required to either remove degenerate elements or reconfigure the mesh, so that the elements satisfy predefined quality criteria (for definition, see [liu94, gol94]).

A second approach is based on a recursive spatial decomposition in the geometrical domain, in which a 3D region is subdivided into cubes or octants [sch90, she91, yue91, per89, mit92, yer94]. This subdivision may result in an isotropic or anisotropic mesh, imposed by object boundaries or resolution limits as stopping criteria [sch96, sch95].

Object descriptions for both types of algorithms are hard to generate from medical datasets. Thus, the application of these mesh generators poses a number of problems:

- Tomographic examinations yield voxel datasets, which first have to be segmented into objects.
- Imperfections in the imaging process (non-linear behaviour of the scanner, motion artefacts, partial-volume effect, noise, etc.) will lead to segmentation errors and likewise to an incompletely defined object.
- Segmentation errors lead to incompletely defined or roughly approximated object boundaries.
- Shapes of anatomical objects are generally complex and highly non-convex. This will (i) complicate geometry-based algorithms and (ii) result in a large number of elements generated.
- Multiple objects of different tissue types are present in a tomogram and must be handled simultaneously.

The following drawbacks arise with the application of the abovementioned methods to medical scan data:

- Given the highly complex structures in the brain, a few of the larger objects exhibit 10^4-10^5 boundary vertices, which lead to numerical stability problems in the Delaunay tesselation algorithms.
- On incompletely defined boundaries, the subdivision approach generates a huge number of small elements or even fails completely.
- The resulting FE meshes do not provide smooth surface representations.
- The subdivision approach has considerable requirements in terms of memory and computational resources.

Because of these disadvantages the cited approaches are not optimal for the generation of FE meshes based on medical datasets.

The basic algorithm we describe in this deliverable takes advantage of the discretization of space inherent in the medical scan data. As a first step the image dataset is transformed into an isotropic hexahedral mesh of user-defined spatial resolution. The cubic elements of this mesh are then subdivided into tetrahedra.
This approach also enables the generation of smooth representations of object boundaries. Smooth boundaries are highly important in nonlinear finite element analyses. Imagine, for instance, two FE meshes (e.g. representing skull and brain or meniscus and bone) whose surfaces are very jagged. If, during a simulation, the meshes are sliding against each other, the rough surfaces cause additional friction. Obviously, this artificial friction will negatively affect the accuracy of the simulation results. In order to avoid this annoying effect we propose a hybrid algorithm that combines features of the spatial decomposition technique and the boundary representation method. By introducing a triangular surface into a volumetric tetrahedral mesh we are able to create smooth surfaces maintaining high mesh quality with regard to numerical stability of the solution process.

The remainder of this deliverable is organised as follows. Some definitions will be given in the following chapter. The concept of a basic meshing algorithms will be described in detail in Chapter 3, alternative approaches are presented as well. Chapter 3 describes the general ideas and discusses the properties of the mesh generators. Chapter 4 illustrates the current status of implementation and presents meshes obtained with the algorithms of Chapter 3. Chapter 5 focuses on the planning of further technical developments of the meshing software to be realised in the SimBio project.

2. Definitions

2.1 3D Image Datasets

Three-dimensional medical images can be thought of as a set of addresses with each address being labeled with a defined gray value. The set of addresses which is often referred to as the image lattice can be written formally as:

$$A = \{ (s, r, c) \mid 0 \le s < nslices,\ 0 \le r < nrows,\ 0 \le c < ncolumns \}, \tag{1}$$

where *nslices*, *nrows*, *ncolumns* denote the number of slices, rows and columns of the 3D image. Each address can be interpreted as a point in the Euclidean space (depicted as circles in Figure 2). The assignment of labels to the elements of *A* can be formulated as a mapping

$$f_{image} : A \rightarrow G\ , \tag{2}$$

where G for MRI images is normally chosen as

$$G = \{ 0, 1, ..., 255 \}. \tag{3}$$

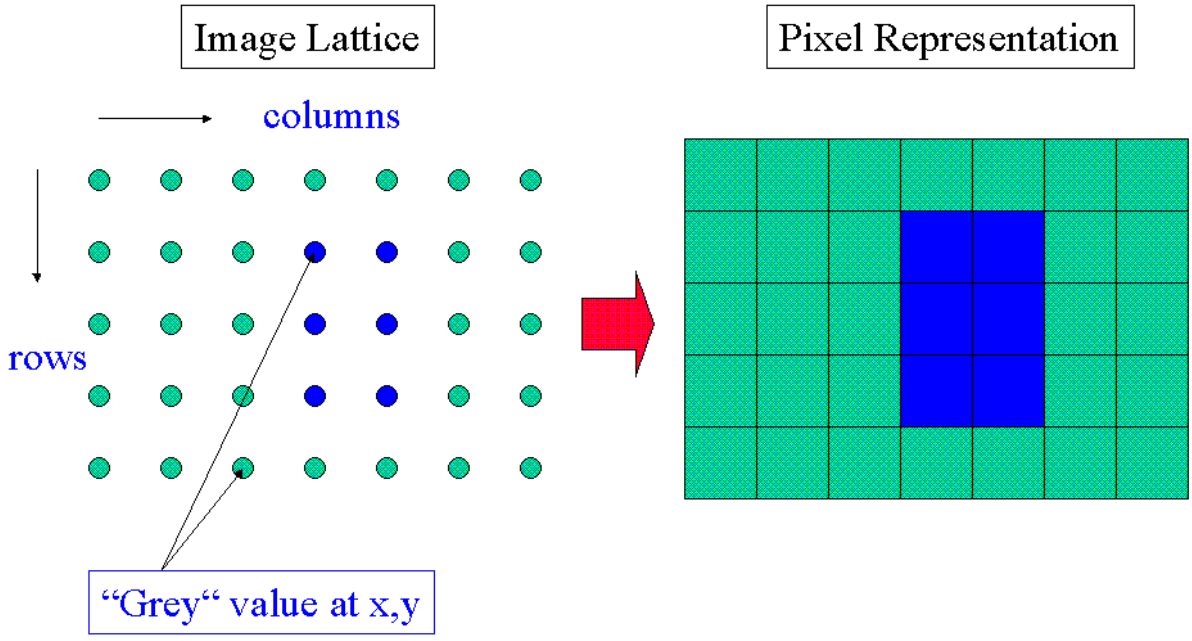


Figure 2: From image lattice to pixel representation. For 3D data the transformation is analogue.

Within this framework a voxel v can be defined as an element of the set A with a label l ∈ G given by the mapping (2). Voxels are in almost any case visualised as intensity coded cubes defined in the Euclidean space (see Figure 2 showing the 2D case). Therefore we will use the denotation *voxel* for a labelled image lattice point as well as for its (mostly) cubic representation in 3D space. If we apply a function $f_{SEGMENT}$ to the set of voxels V that classifies its elements v depending on their intensity l and

then assign a unique label u to each member of a class we obtain a segmented image L. Mathematically written, the voxels of the original image are subject of the mapping

$$f_{SEGMENT} : V \rightarrow U , \tag{4}$$

where U = {0, 1, ..., u_{MAX}} denotes the set of all assigned labels u. A detailed design description of segmentation algorithms used in SimBio is given in D1.1a.

2.2 Finite Element Meshes

The buildings blocks of a finite element are nodes and the connections between these nodes. In the language of geometry, the nodes define the corners of a polyhedron and the connections between the nodes define its edges. The mesh generator described in this design report is capable of creating hexahedral, tetrahedral and hybrid meshes. By definition, a tetrahedral finite element is delimited by four, a cubical finite element by eight nodal (corner) points. The latter element will be referred to as *brick* or *brick element* in the remainder of this text.

The concept of a geometrical entity that we term *a cell* plays an important role in our algorithm. We define a cell as a cubus with eight corner nodes and possibly additional nodes on its faces and edges. Thus, we can regard a brick as a special case of a cell. Theoretically, a cell can be directly transformed into a finite element. However, in our approach the troublesome setup of a variety of elements with nodes on edges and faces is not necessary.
The set of nodes belonging to the i-th cell will be denoted as N_i and the j-th member of N_i will be referred to as the node n^j_i. The set of all cells is called C and its number of members w will be dependent on the spatial mesh resolution chosen by the user. C is defined as

$$C = \{N_1 \bigcup N_2 \bigcup \quad ... \quad \bigcup N_w\} . \tag{5}$$

Finite elements are created whose nodes are a subset of the set of *corner points* of the cubic voxel representations (see Figure 3 showing a 2D mesh). This set will be referred to as B.

In the case depicted in Figure 3 the labeling of the finite elements is naturally given by the corresponding voxel labels. For FE meshes with lower resolution than that above (e.g. an FE element with an edge length of three voxels, see example Figure 4) the area that is covered by an element has to be scanned by the algorithm. A decision which label is to be assigned to the element is then based on the result of this scanning process. In our implementation the majority label of the pixels that are covered by an element will become the material label of the finite element.

It should be emphasised that the node colouring in Figure 3 is not arbitrary. Each node is assigned the colour of the pixel whose upper left corner is represented by the node. The colouring of edge nodes is arbitrary because the edges of an image are normally labelled as background. Background is not taken into account in the meshing process, i.e. no finite element is created with a background label. Though for simple hexahedral and tetrahedral meshes the colours of the nodes do not have any impact on the mesh generation procedure (see Section 3.2), node colouring becomes important when the special algorithm for generating smooth object boundaries is invoked (see Section 3.4.2)

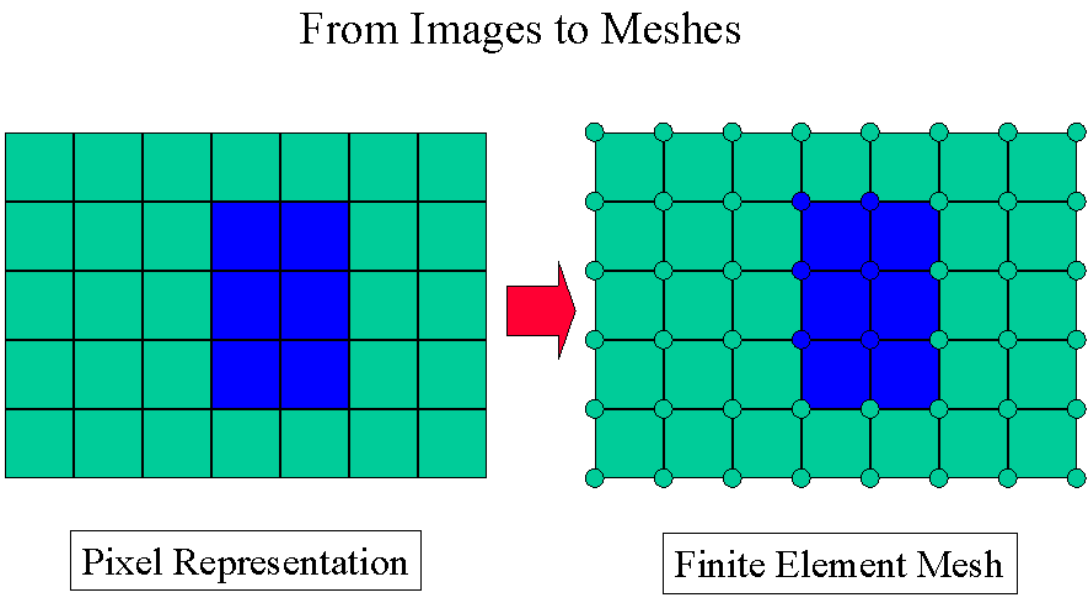


Figure 3: From images to meshes: The corner points of a pixel/voxel are the nodes of the FE mesh.

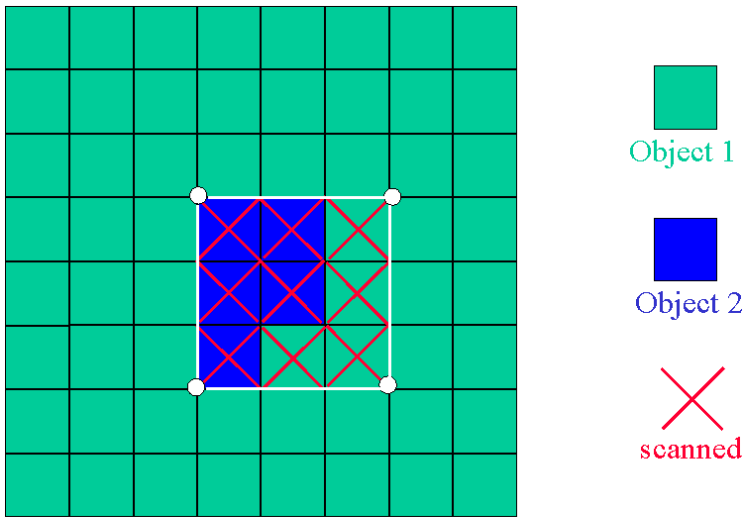


Figure 4: This image shows an embedded pixel object (blue) and a finite element (white) that is assigned the material label „blue“ as the majority of the pixels covered by this element is blue.The pixels marked with a red cross are scanned by the algorithm for finding the appropriate material label.

Now we can mathematically describe the set of nodes M that constitutes a mesh. For mesh type I the set M is simply written as

$$M_I = C \bigcap A, \tag{6}$$

the intersection of the cell set C and the set of image lattice points A.
For mesh type II the set of nodes M is the intersection of the cell set and all possible corner nodes of a voxel grid (set B)

$$M_{II} = C \bigcap B. \tag{7}$$

By definition the number of elements of B is

$$n^3 + n * (2n - 1), \tag{8}$$

where n is the dimension of the isotropic voxel grid (in the SimBio project n normally equals to 256).

In the Section 3.4.2 we will see that for smoothing mesh surfaces it becomes necessary to introduce additional nodes into the mesh that is described by the set M. These additional nodes are collected in the set S whose cardinality depends on the number and the shape of the objects and the spatial mesh resolution. Thus a smoothed mesh is spanned by the set of nodes K

$$K = M \bigcup S. \tag{9}$$

The elements of K are numbered with the labels k_1 to k_n.

Finally we want to give a formal description of a finite element mesh:
An FE mesh is represented by the set K in combination with a set of elements E. A member of E is described by several integer numbers, e.g. for a tetrahedron, four numbers suffice:

$$k_1 \quad k_2 \quad k_3 \quad k_4. \tag{10}$$

The four numbers are the labels of the elements of K that define the corner points of a tetrahedron. For brick elements the description is similar except for the number of integer numbers defining the finite element.

3. Algorithms

3.1. Introduction

Without loss of generality we will assume a 3D segmented MRI dataset L as input for our mesh generator. Voxels with the label u=0 are interpreted as image background and the creation of background elements is suppressed. In our problem domain, the segmentation labels $u \neq 0$ correspond to different objects (i.e. soft tissue, hard tissue, etc.) and/or regions with specific material properties (i.e. electrical conductivity, stiffness).

3.2 The Basic Algorithm

The basic idea of our algorithm is to simplify the task of meshing by exploiting the discretisation of 3D space that is a given for any scanned medical dataset. Introducing cells of different size into the 3D dataset yields anisotropic meshes with a significantly reduced number of nodes. In the final step the

cells are sequentially processed according to the number of nodes a cell contains (i.e. the cardinality of its corresponding set C) to generate a tetrahedral mesh. The basic algorithm evolves in three steps:

1. Isotropic subdivision of the input image into bricks (see image I in Figure 5).
2. Collection of bricks to form cells (see image II in Figure 5).
3. Generation of tetrahedral elements (see image III in Figure 5).

Figure 5 shows the intermediate results of the three phases for a selection of slices of a segmented skull-brain dataset.

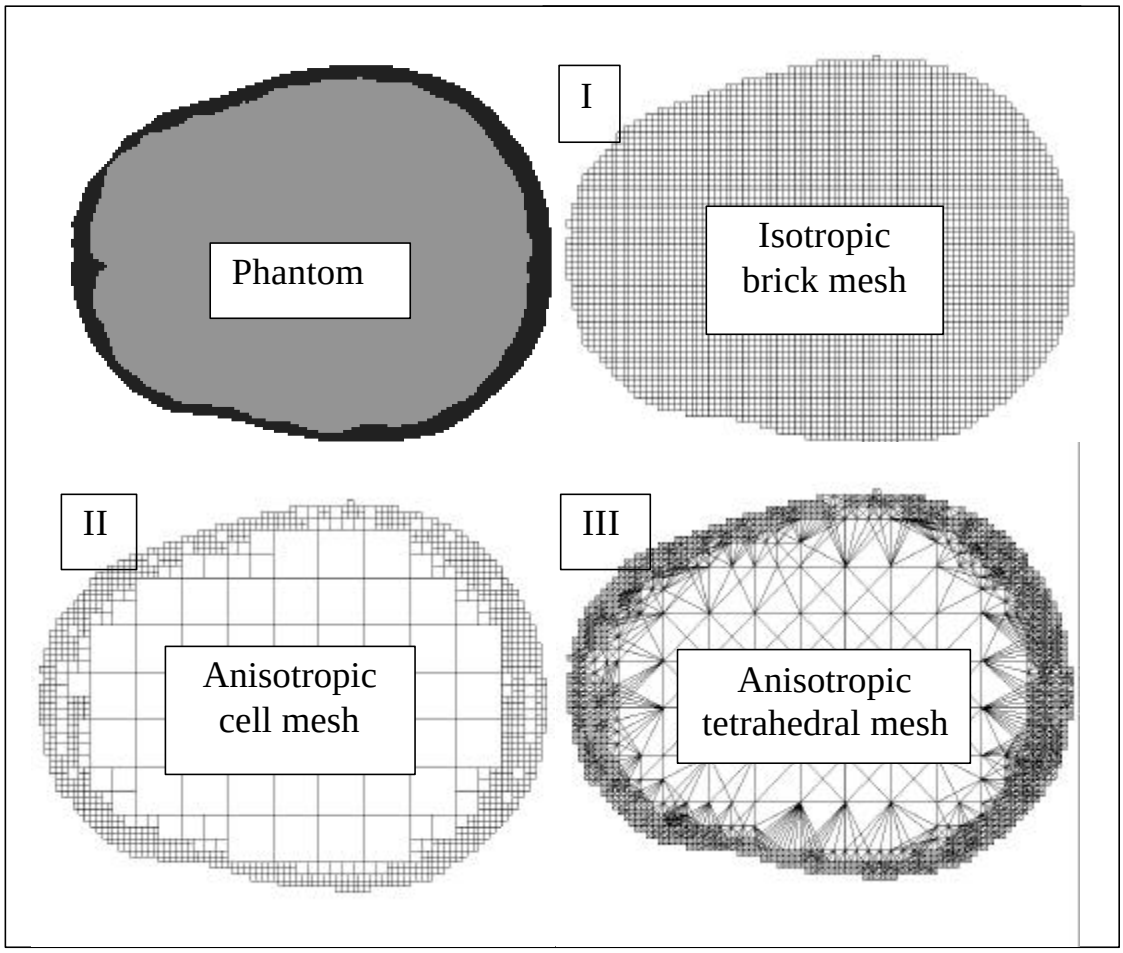


Figure 5: The three phases of the basic algorithm created for a skull-brain phantom.

1. The input image is first decomposed into bricks (for definition see preceding chapter). The edge length e_{min} of the bricks is defined by the user in terms of voxel edge lengths. To preserve the relatively fine skull structure e_{min} must not be set too high. Typically, e_{min} is set to 2 leading to a brick edge length of 2mm (if the voxel edge length of the input image is 1mm in all directions). A material label is assigned to a brick according to the result of the scanning procedure described in the previous chapter.

2. The isotropic subdivision is followed by a collection step. By recursively traversing the isotropic brick mesh (i.e. the result of phase I) bricks of identical material labels are collected to form cells with larger edge lengths. A cell inherits the material label of its building bricks. The maximal edge length e_{max} is a parameter to be specified by the user. The ratio e_{max} / e_{min} is a measure of anisotropy. In the example shown in Figure 5 the anisotropy ratio is eight. In image II of Figure 5 one can clearly see that cells of different edge length have been produced. Many of the cells depicted have nodes on their edges and faces. For carrying out FE analyses

based on such meshes a rather laborious FE formulation is required. For this reason such meshes will not be used within SimBio.

3. Starting from the anisotropic cell mesh an anisotropic tetrahedral mesh is generated. For this purpose each cell is processed separately. At first the algorithm checks how many nodes the cell contains. If a cell is made up of eight nodes (i.e. is a brick) then the cell is subdivided into tetrahedra, e.g. according to the scheme depicted in Figure 6. Our algorithm offers the subdivision of a brick into five as well as into six tetrahedra. If this step is omitted the algorithm will yield hybrid meshes. Cells containing more than eight nodes are tesselated by Delaunay algorithm. To produce tetrahedra with faces compatible with those of neighbouring cells, we selected an incremental tesselation algorithm. We have found the Clarkson [cla92] algorithm to be modifiable for our purpose. Since the typical cardinality of a set C is relatively small (9-150), the tesselation is stable and fast. A tetrahedron inherits its material label from its mother brick cell.

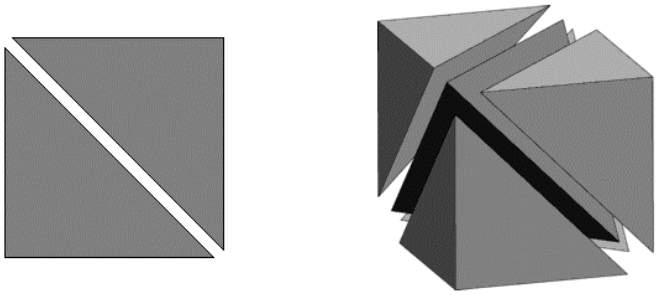


Figure 6: Subdivision of a square into two triangles in two dimensions corresponds to a subdivision of a brick into five tetrahedra in three dimensions.

3.3 Properties of the Basic Algorithm

Before we are going to discuss further developments the characteristics of the basic mesh generator are compiled:

1. It is fast: producing a mesh consisting of around 200000 tetrahedra takes 30s on a PC.
2. By specifying two parameters the user influences the number of elements to be created.
3. Tetrahedra produced have good quality with regard to FE analyses (see Section 6.2).
4. Hybrid meshes consisting of bricks and tetrahedra can be generated.
5. Node numbering is satisfactory (see Design Report D3a).

Item 5 reveals that another positive feature of our mesh generator is the way the mesh nodes are numbered. The nodal numbers determine the bandwidth of the FE matrices. As a large bandwidth of an FE matrix may slow down the (parallel) solver execution, meshes often have to be post-processed by renumbering algorithms [gibbs76, cuthill69]. These algorithms aim at minimising the difference D between the largest and the smallest node numbers in an element. The relation between D and the bandwidth BW of an FE matrix is given by the formula

$$BW = \max_e \left[D^{(e)} \right] + 1 . \tag{11}$$

As our mesh generator creates the elements during a row-by-row and slice-by-slice traversal through the MRI dataset, the maximum difference D_{max} is equal to the number of nodes that are produced for generating the FE elements of a single MR slice. However it should be stated that renumbering of nodes for meshes produced by our algorithm *does* have a positive effect on calculation time of the solver routines. The question whether or not renumbering yields a positive net effect has to be answered from case to case (see D3a).

Though the mesh generator is applicable in its current state, further improvements are planned within SimBio. Those improvements mainly refer to the creation of boundary representations of the human

body parts modelled within the SimBio project. The following sections describe extensions of the basic algorithm that enable better boundary modelling than that achieved with the basic version.

3.4 Extensions of the Basic Algorithm

This section describes two possibilities to create FE meshes whose external and internal boundaries between different materials are smoother (in a sense to be defined) than the meshes created by the basic algorithm. The first option called „nodeshifting“ will only be available as post-processing for isotropic brick meshes (that are often referred to as digital meshes in literature). The second option exploits the „marching tetrahedra“ algorithm and creates tetrahedral meshes with high resolution at material transition regions.

3.4.1 Node Shifting

According to a proposition of Camacho [cam97] the boundaries of digital or brick meshes can be smoothed by shifting nodes of the FE mesh. For each node of the mesh the material types (e.g. skull, brain, meniscus, bone) of all eight elements sharing that node are determined. If a material type is represented by one, two or three of the eight elements, the elements of this material type are designated as the *minority* elements. Relative to the central node, the spatial position of the centroid of the minority elements is calculated and assigned the coordinates (x, y, z). The node is displaced by the vector

$$\Delta x = k * x, \qquad \Delta y = k * y, \qquad \Delta z = k * z\,, \tag{12}$$

where k denotes a user-defined parameter that controls the degree of smoothing. Figure 8 shows a simple example for a 2D digital mesh.

The influence of the node-shifting procedure on the accuracy of simulation results is studied in detail in the cited paper. Preliminary SimBio results obtained with node-shifted meshes are presented in D3a.

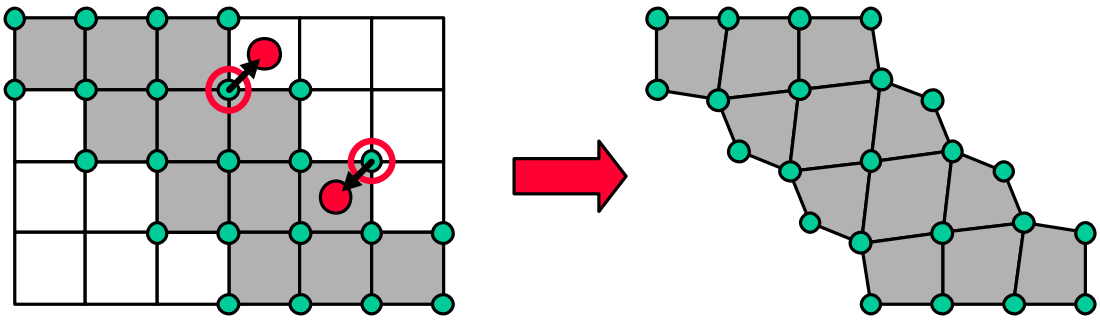


Figure 7: A simple 2D example of a node shifted mesh. The red circles symbolise the centroids.

3.4.2 The Marching Tetrahedra Algorithm

3.4.2.1 Introduction

The result of this version of the meshing algorithm is a tetrahedral mesh whose resolution is higher in the vicinity of boundaries than at other object regions. Input for the marching tetrahedra algorithm is an isotropic brick mesh. For this algorithm to produce good **external** boundary representations it is necessary to prescribe a criterion for the creation of elements that differs from that given in Chapter 2 (remember: the grey value of the majority of the voxels covered by a brick is assigned as material label).

Thus, input brick meshes for the marching tetrahedra version differ from those described in 3.2. The modification will be further explained in the next section.

3.4.2.2 Modified Brick Meshes

To simplify the understanding of the approach, the modified process of isotropic mesh generation is clarified in the following with help of a two-dimensional example. The observations made in 2D can again easily be extended to 3D where the 3D counterpart of a quadrilateral element is a brick and that of a triangle is a tetrahedron.

Figure 8 shows a 2D image lattice consisting of 9x9 image lattice points (pixels). Some of the pixels are labeled as object pixels (the object pixels are coloured in magenta). A 2D finite element mesh is generated by traversing the image row-by row and pixel-by-pixel. The nodes are coloured according to the scheme described in Section 2.2. A finite element is created if ONE of the four nodes that are the corner points of a potential finite element is labeled as object. In the next chapter we will see that this weak condition is favorable for precisely modelling an object with smooth boundaries. Mesh nodes (i.e. corners of the finite elements) are represented by circles where black circles are mesh nodes that make part of object and white circles are nodes that are labeled as background. The isotropic quad mesh adds a „halo“ of background labelled nodes around the original image object. Such a blown-up FE mesh does not necessarily preserve small structures as one can see in the right part of the image where the two „fingers“ of the object are fused to a single structure. In the next chapter we will see how we can regain these small structures.

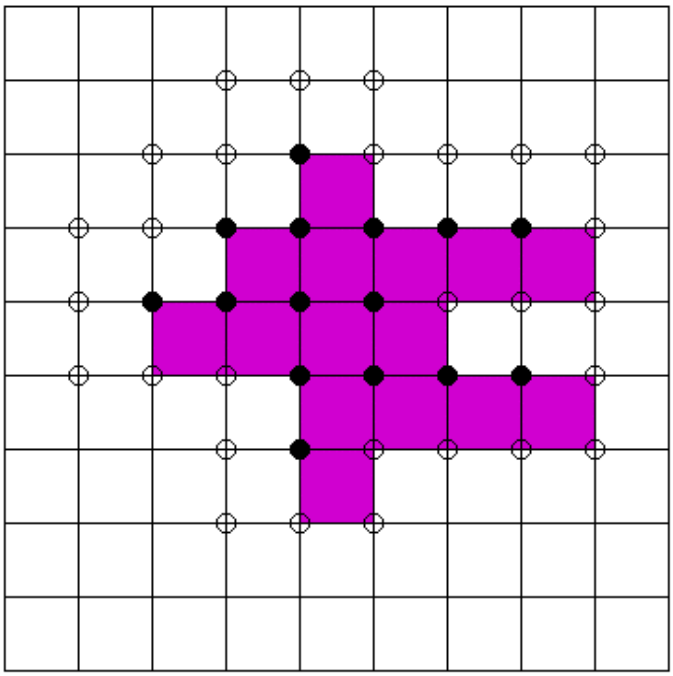


*Figure 8: A quadrilateral FE mesh with spatial resolution of **one** pixel edge lengths.*

3.4.2.3 The Marching Tetrahedra Algorithm

As mentioned earlier, in the final step the brick mesh is transformed into a tetrahedral FE mesh with smoothed boundaries. Each brick of the isotropic mesh is processed separately. The way in which a brick is tessellated depends on the result of a surface detection test (SDT). The spatial resolution of the resulting tetrahedral mesh (i.e. the number of mesh nodes) is governed by the number of objects present in the input data. The tetrahedrisation of bricks and its dependence on the result of the SDT is explained in detail in this section. First of all it is checked whether or not all eight nodes of a brick share the same material label. If the result of this test is positive (i.e. all nodes share a common label) then no surface has been detected and the brick is simply subdivided into five tetrahedra according to Figure 6.

If the result of this test is negative (i.e. the nodes have different labels) a material boundary is passing through the brick. In this case a special treatment becomes necessary. In the following we explain the

processing of such bricks in detail. Again, it is more comprehensible to first look at the problem in two dimensions.

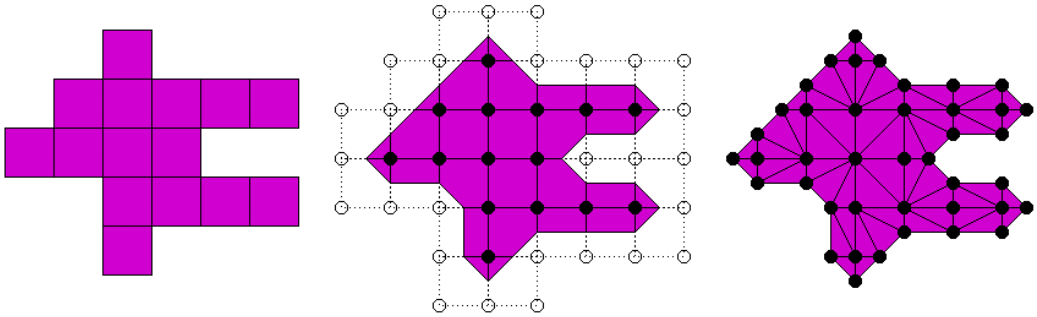


Figure 9: The transformation from the original pixel/voxel object via the quad/brick mesh to the final triangle/tetrahedra mesh is shown with a simple example.

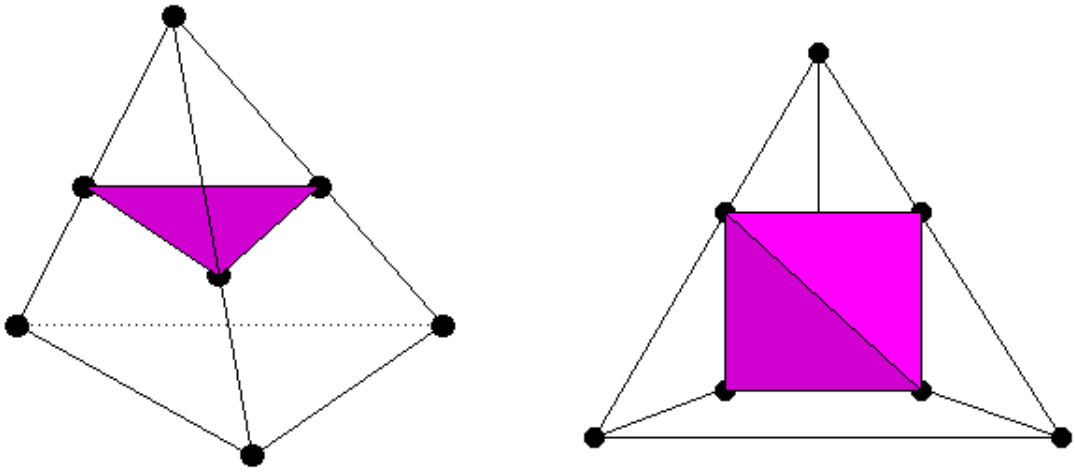


Figure 10: Insertion of triangles into tetrahedra to define material boundaries.

The test object of the preceding chapter is used again. The image in the middle of Figure 9 shows how the boundary is generated. Several cases have to be handled. A quadrilateral element might have three nodes with a label representing the outside and one labeled as being inside the object, the opposite case can be observed, or both labels occur twice in one element. In the first two cases a triangle or polygon is generated, in the latter case rectangles are created. The image on the right shows how one can arrive at a triangular mesh of the object by inserting additional nodes on the boundary and by subdividing the polygons and the quadrilaterals into two triangles. The resulting mesh is a very precise representation of the original object with a smoothed boundary. The degree of smoothness might be increased by applying a local smoothing operator to boundary nodes (see Section 3.4.2.4).

In 3D the concept for generating smooth boundaries is similar, but more cases must be discriminated. A brick that contains a surface is subdivided into five tetrahedra according to Figure 6. Each tetrahedron is checked in the same way as the brick for the presence of an object or material surface. The assignment of labels to the four nodes of a tetrahedron can appear in six different ways. However, the six cases can be reduced to two generic cases that are depicted in Figure 10.

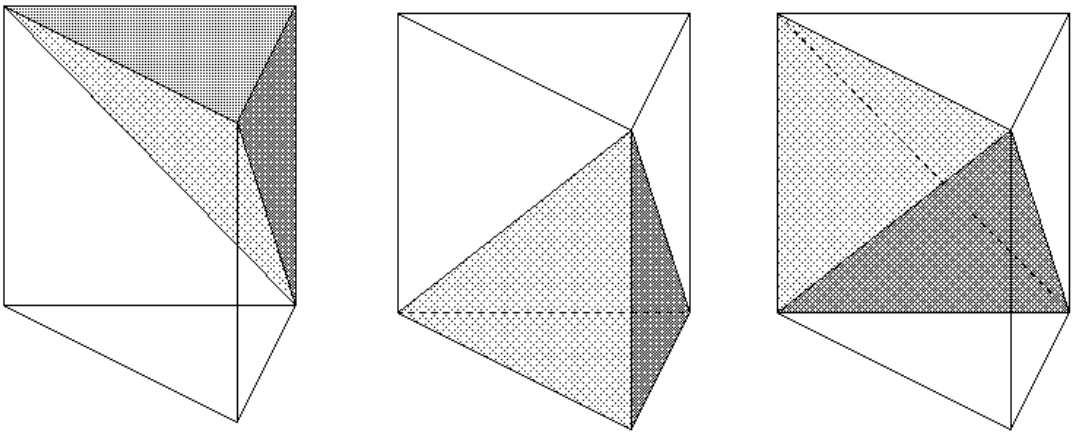


Figure 11: Tetrahedrisation of a prism.

In 3D we use triangles to divide a tetrahedron into two subspaces. The left image in Figure 10 represents the case where one node (in the example the upper node) of the tetrahedron has a different material label than the others. The right image in Figure 10 shows the case in which two nodes (the upper and the invisible node behind the triangle wall) of the tetrahedron have other labels than the rest. In the first case the tetrahedron is subdivided into a tetrahedron and a prism. Three additional nodes that span the triangle are introduced. In the second case two triangles are added causing four additional nodal points. By the construction of this wall two prisms are created. As it is favourable to have meshes consisting of a single element type (to avoid the setup of a prism finite element), these prisms can be subdivided into tetrahedra according to Figure 11.

3.4.2.4 Local Boundary Smoothing

At this phase of the project we cannot foresee which degree of mesh smoothness will be required by the SimBio evaluation applications. If the approaches described in this design report do not fulfill the requirements imposed by the applications, we will further explore the option of locally smoothing the internal and external mesh boundaries. Promising techniques have been developed to generate C1 smooth surfaces that we could implement as a postprocessing step for the meshes obtained with the algorithms described above.

3.5 The Approach used for the KneesUp EU Project

The approach used by USFD and ESI for progressing from MRI data to the creation of a 3D mesh of the knee in the KneesUp Project is summarised below:

- Segmentation of the components of the knee, for example, the bones and menisci using edge detection, region growing and smoothing algorithms written in-house (USFD) using Matlab™.
- Forming smooth contours around the segments.
- Sampling the volume (e.g. the femur) at 4mm slices in the Z plane and then sampling points at regular intervals per contour.
- Application of a Delaunay triangulation to the contour points to create a surface mesh using NUAGES (Bernhard Geiger, INRIA Sophia Antipolis, France) [gei93].
- Once the boundary representations of the segmented objects were achieved, the non-rigid surfaces were imported by ESI into the commercial FE package I-DEAS® to create solid meshes.
- Where anisotropy (such as in the cruciate ligaments and menisci) had to be modelled hexahedral elements were preferred because they have a more easily handled local coordinate system with which to define anisotropy. Thus, ESI remeshed the articular cartilages, cruciate ligaments and menisci manually to form hexahedral elements.

Although this approach is still feasible, it is not the preferred option for generating the finite element meshes for the knee for SimBio. The technique was robust but because it was not an automated process, it required considerable manual intervention and was thus labour intensive. However, it is still available as a backup method in the unexpected event of problems with the other methods.

3.6 The Mesh Templates Approach

Although segmentation and mesh generation are represented as separate tasks in D1.1a and D1.2a, we believe there is a good case for solving these tasks together for the knee. The principle approach is to segment the image and then register the segments to a set of pre-meshed standard segments or *mesh templates*. The mesh template can then be mapped directly onto the patient specific segments.

There are three steps in this process.

(a) Register the patient MR image to a standard, pre-segmented, pre-meshed image i.e. a *mesh template*

(b) Refine the segments for the actual patient mesh.

(c) Individually register the new segments to the corresponding template segment using the same registration method using a simple non-linear (local) registration technique (described in D1.1a)

The method proposed for creating meshes for the knee is based around templates and is summarised as follows:

- For each component of the knee joint a parameterised template of its boundary will be constructed.
- The templates used for the menisci, articular cartilages and (eventually) cruciate ligaments will then be discretised into hexahedral elements, preferred for the reasons given in 3.5.
- The key will be to register the segmented knee components to the mesh template.
- The mapping produced by the registration algorithm will permit the template mesh to be distorted to fit the segmented component, forming a new mesh while still maintaining the connectivity of the original template mesh.
- Constraints will be applied to ensure that the mesh quality provided by the template is maintained in the new mesh.

3.6.1 Creating the parameterised components: the medial meniscus as an example

A binary voxel dataset is created using Matlab™ to represent the parameterised meniscus. The meniscal volume is created using a quadratic equation with thirteen parameters, consisting of four variables with three coefficients each plus the angle (?) through which the meniscus is subtended. The cross-section of the meniscus is represented by a trapezoid where the four parameters are the inner and outer radii, and the inner and outer heights, each of which is a function of ?. The process is summarised as follows:

- An X,Y,Z array is defined and initialised
- The volume defined by the quadratic is calculated to form a meniscal boundary
- Voxels in the 3-D array that are contained by the meniscal volume are filled with ones
- The resultant binary file is saved

The resultant medial meniscus is shown in Figure 12. For illustration, the parameterised meniscus has been meshed using VGrid initially (see Figures A and B) in Section 4.2. Work is underway to generate meshes by an alternative method using an interpolation scheme in the parameterised space.

3.6.2 Registration

Figure 13 shows the cadaveric meniscus in red and the template meniscus in green prior to applying the registration algorithm. Areas of registration are shown in yellow. Preliminary registration of a pre-segmented cadaveric meniscus to the template has shown promising results, as seen in Figure 14.

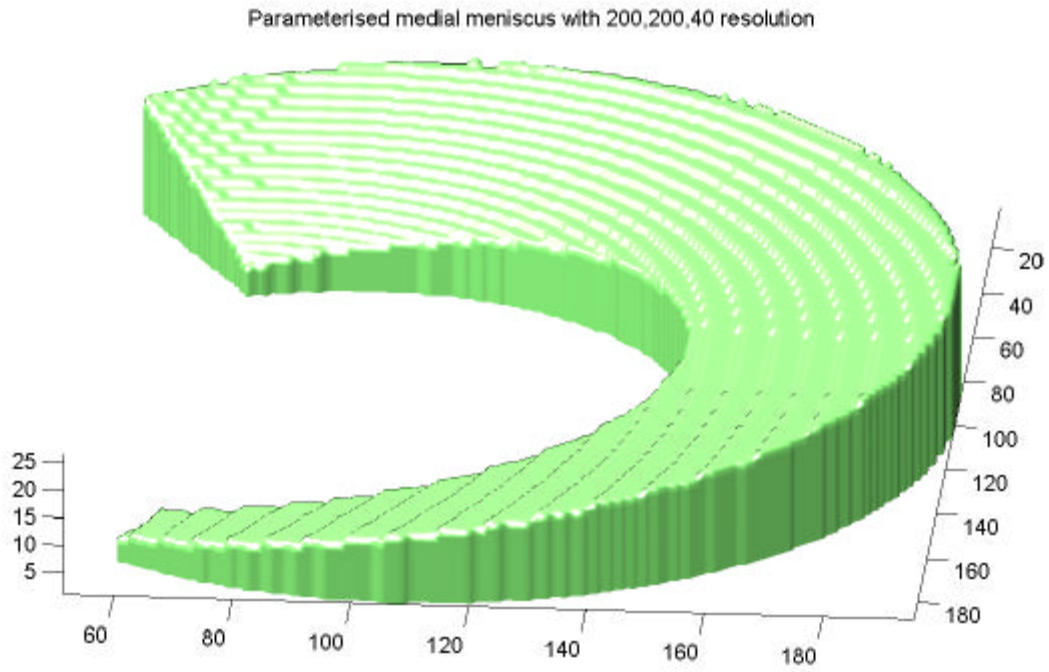


Figure 12: An example parameterised medial meniscus created using Matlab©.

It is expected that the registration will be better for live (i.e. non-cadaveric) segments because the cadaveric knee used for testing the algorithm had become degraded.

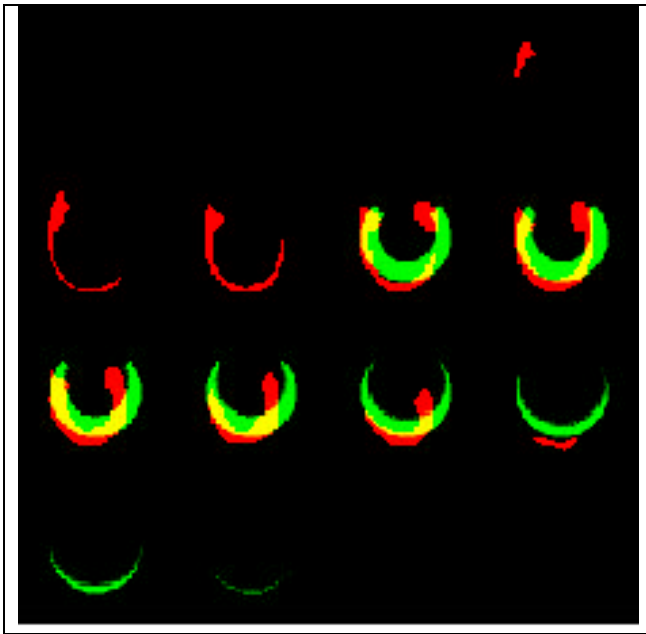


Figure 13: Pre-registration of the menisci

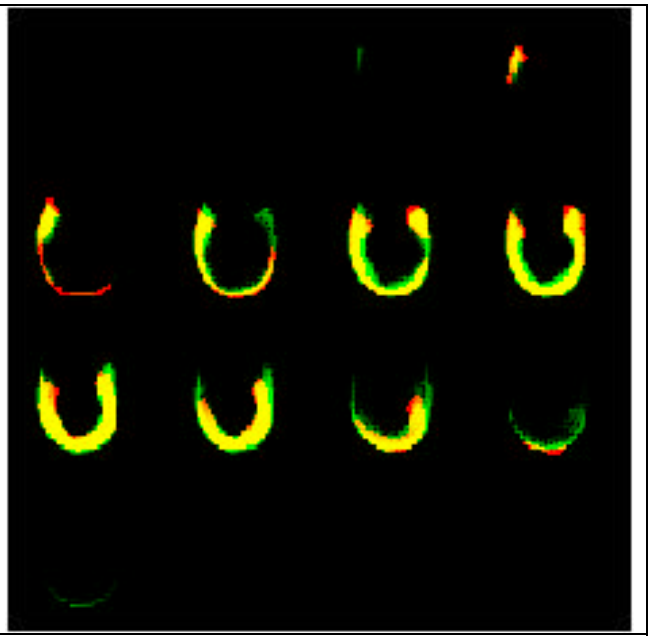


Figure 14: Post-registration of the menisci

4. Implementation

4.1 Current State of the Basic Algorithm

The basic version of the mesh generator enables the creation of hexahedral and tetrahedral meshes with user-defined spatial resolution. For boundary smoothing only the node shifting procedure is currently available. The algorithm has been implemented in the C++ language using the VISTA library for image data handling and the generation of graphical output. The basic algorithms described in Sections 3.3 and 3.4.1 have been tested with a number of MR datasets which were preprocessed in the following way:

- Acquisition of a T1-weighted MR brain dataset,
- Trilinear interpolation to an isotropic resolution of 1mm.
- (Optional) noise filtering with a Lee filter
- Classification of the tissue types using a method of ST 1.1.

4.2 Results Obtained with the Basic Algorithm

The following figures give some examples of meshes generated with the algorithm described in Sections 3.2 and 3.4.1.

Figure 15 depicts a high resolution mesh of the human head comprising five different tissue types. The mesh consists of about 150000 hexahedral elements with an isotropic edge length of 3mm. It was shown [hart98] that such meshes are appropriate for biomechanical simulations related to the head.

Figure 16 depicts a slice through a mathematical phantom representing the four layers of the human head (extracranial tissue, skull, cerebrospinal fluid, brain). This dataset is often used to test algorithms for the source localisation problem. A hexahedral mesh with elements of 2mm edge length is created using the nodeshifing option for smoothing internal and external material boundaries. The effect of this procedure is clearly visible in the slice through the FE mesh that is depicted in Figure 17. The influence of nodeshifting on the accuracy of the simulation results will be examined within the SimBio project (see Section 5.4).

A „real application“ mesh of the human knee is shown in Figure 18. The spatial mesh resolution is 4mm and the nodeshifting method has been applied. The quality of such meshes will be evaluated through simulations planned within Subtask 7.3.

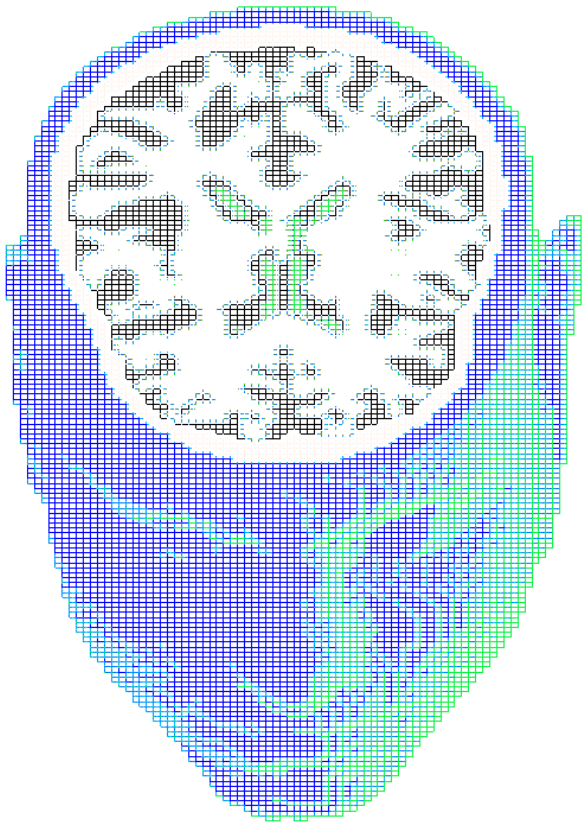


Figure 15: A hexahedral mesh of the human head.

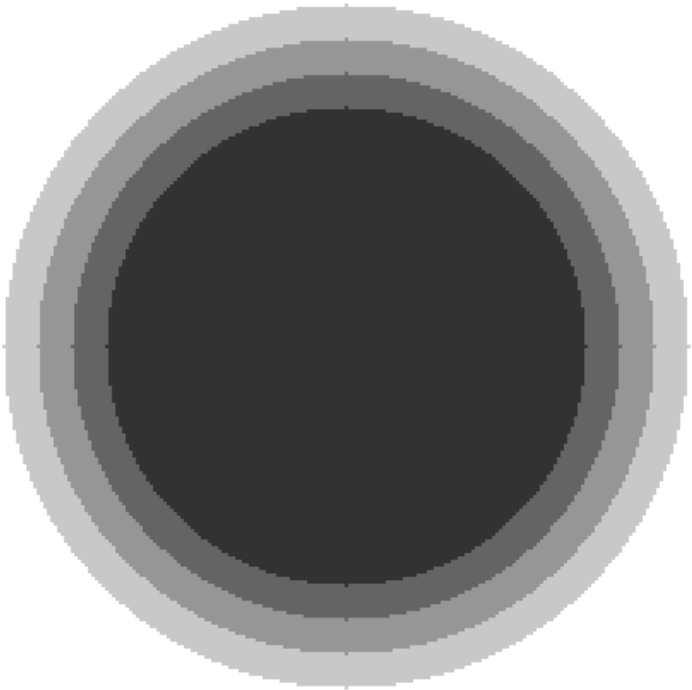


Figure 16: A slice through a 3D image dataset representing a 4-layer head model.

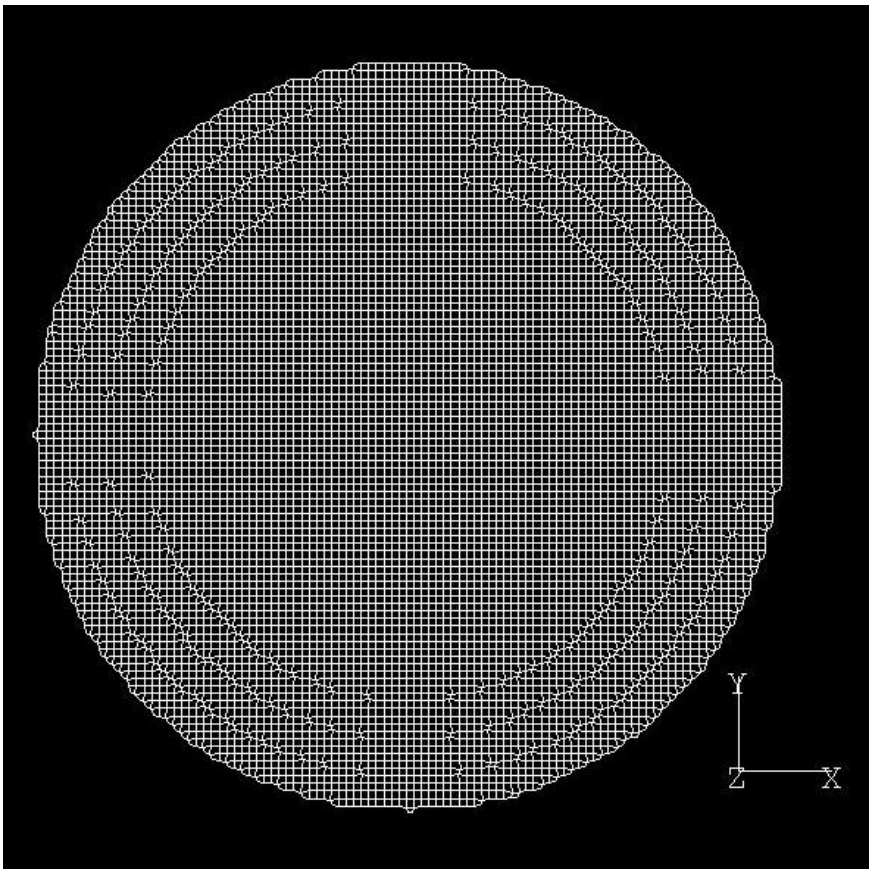


Figure 17: A slice through a 3D hexahedral mesh generated from the dataset depicted in Figure 16. The effect of nodeshifting is clearly visible in the material transition regions.

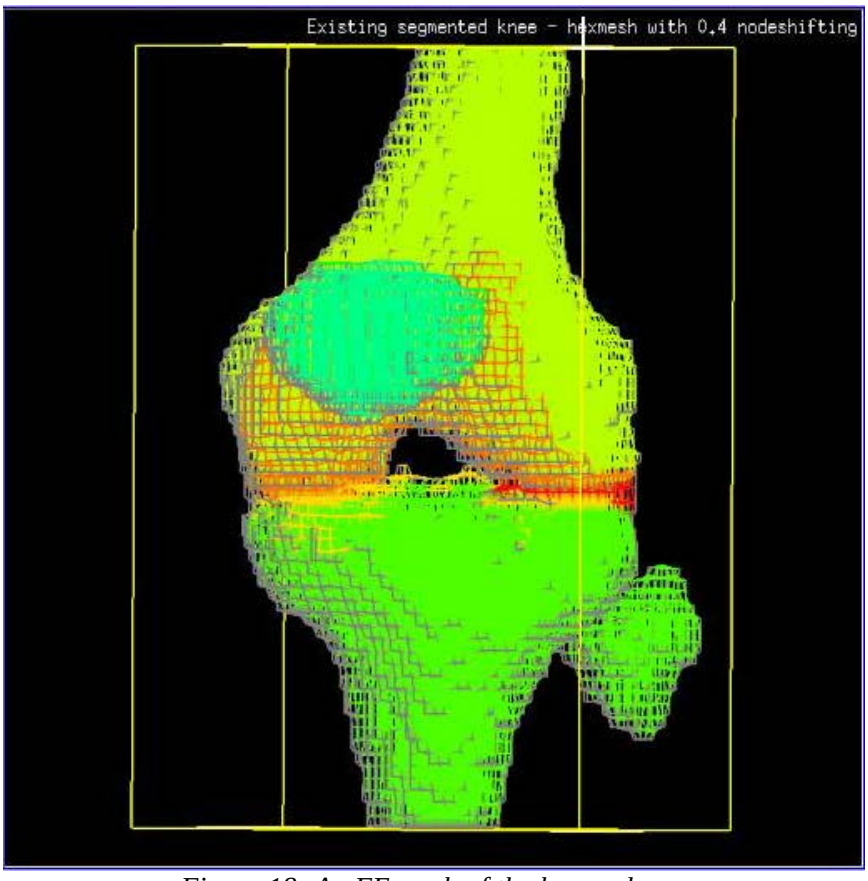


Figure 18: An FE mesh of the human knee.

5. Technical Development Plan

5.1 Meshes in Vista Format

The Vista format has been accepted as SimBio internal file format for medical images as well as for finite element meshes. For the latter the Vista graph type has been chosen. More information about the Vista format is to be found on the SimBio webpage. Depending on the application desired by the user two types of mesh formats have to be provided, one format for sequential applications and a second for applications running in parallel. Since in some cases the user will be interested in creating optimal partitions using the DRAMA tool (see D3a), the format conventions specified for distributed meshes hold also true for the DRAMA output meshes.

5.1.1 Sequential Mesh Format

An FE mesh in Vista format consists of two graph structures. A graph is a linked list of elements. The elements of the first graph representing a mesh are the nodal points of a mesh. This graph is referred to as the vertex graph. Vista offers six different types of nodes. In the simplest case only the Cartesian coordinates of the FE node are stored. All the other node types offer the possiblity to save additional node related information (e.g. normals, curvature, colour). Node related information not foreseen in this framework (e.g. a force applied to a node) can be stored in additional one-dimensional images that are included to the Vista mesh file.
The second graph called the primitives graph. Its nodes are the finite elements, i.e. tetrahedra or hexahedra. Finite elements are represented by a set of integer numbers according to formula (10). The links between nodes of the primitive graph can be used to express neighbourhood relationships.
Each Vista file starts with a list of attributes that is human readable. Besides compulsory attributes self-defined attributes can be added to the list. For our purpose it is useful to store the number of elements and nodes in this list.

5.1.2 Distributed Mesh Format

The distributed mesh format is rather similar to the sequential one, the only major difference being the number of graphs being stored in the Vista file. Whereas in the sequential case only two graphs are stored, for the distributed mesh 2*n graphs will be saved, where n denotes the number of partitions (i.e. the number of processors the code will use). That means that for each subdomain of the mesh a separate graph structure is created. For the element representation (see formula 10) double numbering is used which is a standard approach generally used in parallel FE applications. Double numbering means that each node is characterised by the processor number it is residing on and by a local ordering number.

5.2 Implementation of the Marching Tetrahedra Algorithm

The „marching tetrahedra“ algorithm that has been explained in detail in Section 3.4.2.3 will be implemented as an additional option for generating smooth internal and external surfaces. The resulting meshes will be tested in the SimBio evaluation applications with regard to their suitability for nonlinear FE analyses including contact/friction models.

5.3 Mesh Quality Module

For the sake of stability and accuracy of any FE simulation carried out with tetrahedral elements the tetrahedra have to satisfy certain quality criteria. As a rule of thumb, angles below 10 degrees are more likely to cause stability problems when solving the equation system. For the tetrahedra resulting from a brick decomposition (see Figure 7) this requirement is fulfilled in any case. We only find 3 different angles (all greater than 10 degrees) and 3 different edge lengths. For more complex versions of the algorithm (see Section 3.3.2) such theoretical considerations concerning element shape cannot easily be undertaken. Therefore it is very useful to implement a software module that checks the angles of tetrahedral elements. Such a module automatically invokes simple refinement strategies (e.g. placing an additional node into the centre of the tetrahedron) whenever an angle falls under a certain limit.

5.4 Analytical Testing of Mesh Quality

The influence of the nodeshifting procedure is to be validated. This is only possible for simple geometries, where analytical formulas for the partial differential equation exist. For SimBio's electromagnetic head-model application, the nodeshifting will be tested on anisotropically conducting multi-layer sphere models (see Figure 17). A more detailed description for this validation can be found in Section 5.3 of the design report for D3a.

References

[bec95] Beck R, Erdmann B, Roitzsch R (1995) Kaskade 3.0 - An object-oriented adaptive finite element code. Technical Report TR 95-4, Konrad-Zuse-Zentrum Informationstechnik Berlin.

[boe94] Boender E (1994) Reliable Delaunay-based mesh generation and mesh improvement. Communications in Numerical Methods in Engineering, Vol. 10, pp. 773-783, Wiley New York.

[buc97] Buchner, H., Knoll, G., Fuchs, M., Rienacker, A., Beckmann, R., Wagner, M., Silny, J., and Pesch, J. (1997) Inverse localization of electric dipole current sources in finite element models of the human head. *Electroencephalography and Clinical Neurophysiology*, **102**:267-278

[cig94] Cignoni P, Montani C, Scopigno R (1992) A merge-first divide and conquer algorithm for Delaunay triangulations. Technical Report 92-16, Instituo CNUCE-C.N.R, Pisa, Italy.

[cla92] Clarkson KL, Mehlhorn K, Seidel R (1993) Four results on randomized incremental constructions. Computational Geometry: Theory and Applications 3, pp. 185-212.

[cuthill69] Cuthill, E, McKee, J (1969) Reducing the bandwidth of sparse symmetric matrices. Proceedings of the 24th National Conference of the Association for Computing Machinery, pp. 157-172.

[dew95] Dewhirst D, Vangavolu S, Wattrick H (1995) The combination of hexahedral and tetrahedral meshing algorithms. Proceedings, 4th International Meshing Roundtable, Sandia National Laboratories, pp. 291-304.

[ede87] Edelsbrunner H (1987) Algorithms in combinatorial geometry. Springer Berlin-Heidelberg.

[fle91] Fletcher CAJ (1991) Computational techniques in fluid dynamics. Springer Berlin-Heidelberg.
[buch97]Buchner H, Rienaecker A, Beckmann RF, Wagner M, Pohlmeier R, Pesch J, Wolters CH, Knoll G, Silny, J (1997) EEG/MEG Source Reconstruction - The Functionality of CAUCHY. ISBET, Zuerich

[gibbs76] Gibbs, NE, Poole, WG, Stockmeyer, PK (1976) An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM Journal of Numerical Analysis 13, pp. 236-249.

[gei93] Geiger B (1993) Three dimensional modeling of human organs and its application to diagnosis and surgical planning, PhD thesis

[gob94] Gobat JI, Atkinson DC (1994) FElt: User's guide and reference manual. Computer Science Technical Report CS94-376, University of California, San Diego.

[gol94] Golias NA, Tsiboukis TD (1994) An approach to refining three-dimensional tetrahedral meshes based on Delaunay transformations. International Journal of Numerical Methods in Engineering 37, pp. 793-812.

[hart98] Hartmann, U, Kruggel, F (1998) Transient Analysis of the Biomechanics of the Human Head with a High-Resolution 3D Finite Element Model. Computer Methods in Biomechanics and Biomedical Engineering 2(1) , 49-64

[ho98] Holt GM, Penrose JM. (1998) Reconstruction of MRI images into 3D models: Problems, pitfalls and solutions. Journal of Bone and Joint Medicine 80-B(Supp III): 259.

[ho99] Holt GM, Penrose JM, Hose DR. Bickerstaff D.R. (1999) The development of a virtual knee joint for computerised testing of orthopaedic implants. Journal of Bone and Joint Surgery 81-B(Supp III): 277.

[joe91] Joe B (1991) Construction of three-dimensional Delaunay triangulations using local transformations. Comuter Aided Geometric Design 8, pp. 123-142.

[joe92] Joe B (1992) Three-dimensional boundary-constrained triangulations. Artificial Intelligence, Expert Systems, and Symbolic Computing, (Eds. Houstis EN, Rice JR) Elsevier Publishers, pp. 215-222.

[lew 91] Lewis PE, Ward JP (1991) The finite element method: principles and applications. Addison Wesley Reading.

[kee98] Keeve E, Girod S, Kikinis R, Girod B (1998) Deformable Modeling of Facial Tissue for Craniofacial Surgery Simulation, Computer Aided Surgery, Vol. 3, No. 5, 228-238

[kui95] Kuijpers AHWM, Claessens MHA, Sauren AAHJ (1995) The influence of different boundary conditions on the response of the head to impact: a two-dimensional finite element study. Journal of Neurotrauma 12, pp. 715-724.

[kru96] Kruggel F, Lohmann G (1996) BRIAN (brain image analysis) - a toolkit for the multimodal analysis of brain datasets. Proceedings, Computer Assisted Tomography '96, pp. 323-328, Elsevier Amsterdam.

[liu94] Liu A (1994) Quality local refinement of tetrahedral meshes. PhD thesis, Department of Computing Science, University of Alberta.

[lo91a] Lo SH (1991) Volume discretization into tetrahedra - I. verification and orientation of boundary surfaces. Computer and Structures 39, pp. 493-500.

[lo91b] Lo SH (1991) Volume discretization into tetrahedra - II. 3D triangulation by advancing front approach. Computer and Structures 39, pp. 501-511.

[mit92] Mitchell SA, Vavasis SA (1992) Quality mesh generation in three dimensions. Proceedings of the ACM Computational Geometry Conference, pp. 212-221.

[per89] Perucchio R, Saxena M, Kela A (1989) Automatic mesh generation from solid models based on recursive spatial decompositions. International Journal of Numerical Methods in Engineering 28, pp. 2469-2501.

[sch96] Schindler FR, Schneiders R (1996) Automatic geometry-adaptive generation of quadrilateral and hexahedral element meshes for FEM. 5th International Conference on Numerical Grid Generation in Computational Field Simulations. Missisippi State University, pp. 689-697.

[sch95] Schmelzer L (1995) Covariant geometry description, WIAS-Preprint No. 152, Weierstrass-Institute Berlin.

[sch90] Schroeder WJ, Shepard MS (1990) A combined octree/Delaunay method for fully automatic 3D mesh generation. International Journal of Numerical Methods in Engineering 20, pp. 37-55.

[she91] Shepard MS, Georges MK (1991) Automatic three-dimensional mesh generation by the finite octree technique. International Journal of Numerical Methods in Engineering 32, pp. 709-749.

[wea94] Weatherill NP, Hassan O (1994) Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. International Journal for Numerical Methods in Engineering 37, pp. 1841-1861.

[yer94] Yerry MA, Shephard MS (1984) Automatic three-dimensional mesh generation by the modified-octree technique. International Journal of Numerical Methods in Engineering 20, pp. 1965-1990.

[yue91] Yuen MMF, Tan ST, Hung KY (1991) A hierarchical approach to automatic finite element mesh generation. International Journal of Numerical Methods in Engineering 32, pp. 501-525.