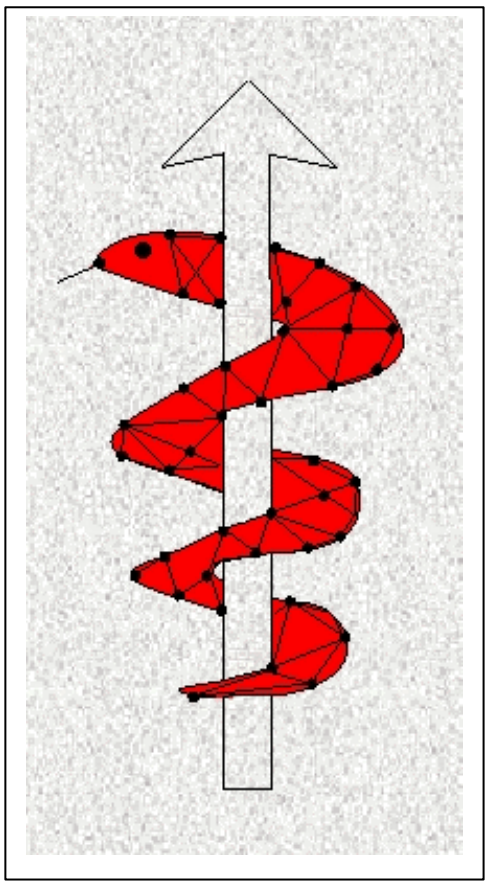
The IST Programme
Project No. 10378

SimBio

SimBio - A Generic Environment for Bio-numerical Simulation

http://www.simbio.de



Deliverable D1.2c
Final Mesh Generation Tool Release

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

Release History

Version	Date
0.1	04.10.02
1.0	10.11.02

The SimBio Consortium :
NEC Europe Ltd. – UK
A.N.T. Software – The Netherlands
CNRS-DR18 – France
Sheffield University – UK
MPI of Cognitive Neuroscience – Germany
Biomagnetisches Zentrum Jena – Germany
ESI Group – France
Smith & Nephew - UK
U. Maribor - Slovenia
Teaching Hospital Maribor - Slovenia

© 2002 by the **SimBio** Consortium

1. Introduction

FE models for medical problems are 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. Then, a finite element description of those objects has to be introduced by a mesh generation procedure.

The problem of 3D mesh generation is a current research topic within the field of computational geometry. Within the SimBio project two novel approaches have been realised, one has been evaluated:

- the VGrid Approach (see Section 2.1) and
- the Mesh Template Approach (see Section 2.2).
- The ZMD tool has been evaluated (see Section 2.3).

The concepts of the first two methods have been thoroughly described in D1.2a and D1.2b. In the cases where the real implementation now differs from the conceptual descriptions in D1.2a and D1.2b, the modifications are pointed out in this document. This is especially true for the marching tetrahedra algorithm (see D1.2a Section 3.4.2 and D1.2b Chapter 4) where –during the implementation- improved ideas for the fast and robust generation of smooth tetrahedra meshes were developed (see Section 2.1.3).

This document outlines the final status of the meshing software developed in the SimBio project. Besides it serves as documentation for all provided software tools (including helping routines and tools). A general user should be able to successfully use the tools and should have an idea of the algorithms behind after having read this document.

2. Final Status of Implementation

2.1 The VGrid Mesher

2.1.1 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≠0 correspond to different objects (i.e. soft tissue, hard tissue, etc.) and/or regions with specific material properties (i.e. electrical conductivity, stiffness). The VGrid meshing tool is fully integrated into the software chain which is: image acquisition=> image segmentation=>mesh generation=>mesh partitioning=> and so on.

2.1.2 The non-uniform meshing algorithm

Building the octree

The octree is built bottom-up. First the voxel image is subsampled at the highest wished resolution r_{min} leading to cells of r_{min} voxels in each space direction. After that, groups of 8 cells forming a potential coarser cell of $2r^3_{min}$ voxels are considered for joining. Depending on the involved material(s) and possibly interfaces, the cells are joined or not (for defining material and interface-dependend resolution see also the description of usage). If they are joined, the majority material (possibly weighted) is assigned to the new cell. This process is continued iteratively, until no more cells can be joined.

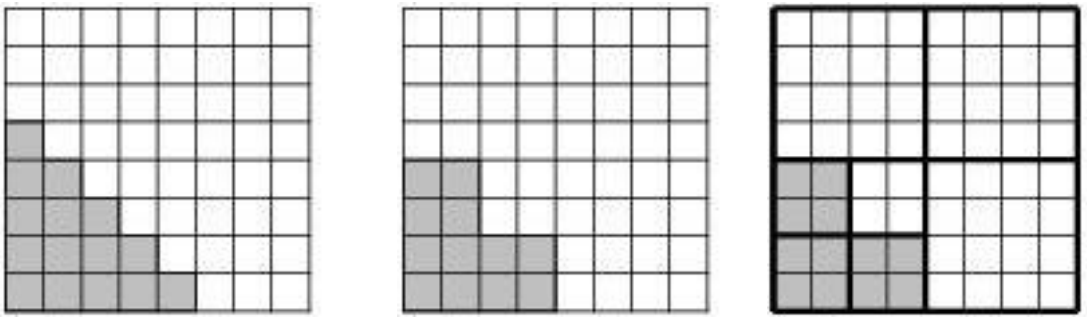


Figure 1: Octree generation in 2D

An octree can be considered as a special mesh, where each cell has the geometry of a cube, but potentially has more than 8 nodes (or more than 4 nodes in 2D). Regarded as a mesh, the octree on the right of Figure 1 has 7 cells, 20 facets (i.e.edges), namely 12 of length 2, and 8 of length 4, and 14 nodes.

Tesselating the octree cells

Given an octree, its cube-shaped cells have to be tesselated in a consistent way, that is, tesselations must match accross octree facets. This is simple in the case of a uniform octree, i.e. all cells have the same size. In this case, we can either output hexahedra, or adopting a simple scheme for subdividing the octcells into tetrahedra (see Figure 3).

The originally pursued strategy of applying a delaunay tessalation of the boundary nodes of an octree cell was skipped because of various reasons:

- It is difficult to assure that tesselations match across octree cells, at least if cells are to be processed in sequence, independently of each other
- Due to the highly regular nature of the input, and an inherent weakness of the delaunay triangulation in 3D, some kind of degenerate tetrahedra (so-called *slivers*) cannot easily be avoided.

Therefore, a different algorithm has been designed. It has the following useful properties:

- It runs completely local, that is, considers only the nodes of the current octree cell
- It guarantees consistency across octree boundaries
- It guarantees a minimum quality of the resulting tetrahedra
- It is insensitive to vertex coordinates, and therefore very stable
- It is very efficient (linear in the number of nodes)

The main idea of the tesselation algorithm is recursive: If we have a (simplicial) subdivision of the boundary facets of an octree cell, we get a subdivision of the octree cell itself by connecting its midpoint with the boundary simplices, see Figure 2 for the 2D case (upper left and lower right octcell). For the boundary facets themselves, we can apply the same idea recursively. This tesselation is evidently consistent across facets, because the subdivision of a facet depends only on the data of the facet, namely, its boundary nodes.

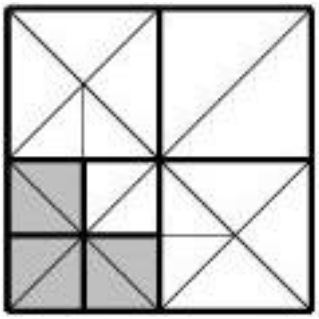


Figure 2: Octree tesselation in 2D. The additional nodes on the boundary of the left upper and right lower cell trigger the insertion of midpoints in these cells. For the cells in the lower left and upper right, template subdivisions are used.

Now, it would be a waste to insert midpoints in every case. Therefore, the rule is relaxed as follows:

- If there are exactly 8 nodes for an octree cell (4 nodes for a facet), then use a template subdivision without new midpoint node.
- Else use the midpoint rule.

The template subdivision of both octree cells and facets can be chosen in such a way that a consistent tesselation on both sides of an octree facet is guaranteed. It is a generalization of the subdivision of a Cartesian grid, where each cell is subdivided into 5 tetrahedra in an alternating, checkerboard way. This subdivision of the Cartesian cells also defines a subdivision of the Cartesian facets, in a way that depends only on the Cartesian coordinates of the facet. We can now view each level of the octree as a Cartesian grid, and apply the subdivision rule implied by this Cartesian grid to the octree facets of the appropriate level. Thus, each octree facet has a uniquely defined triangulation, which is consistent of the template subdivision of incident octcells (important in the case when no midpoint is inserted into the cell).

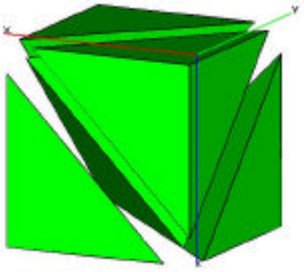


Figure 3: Template subdivision of a cube into 5 tetrahedra

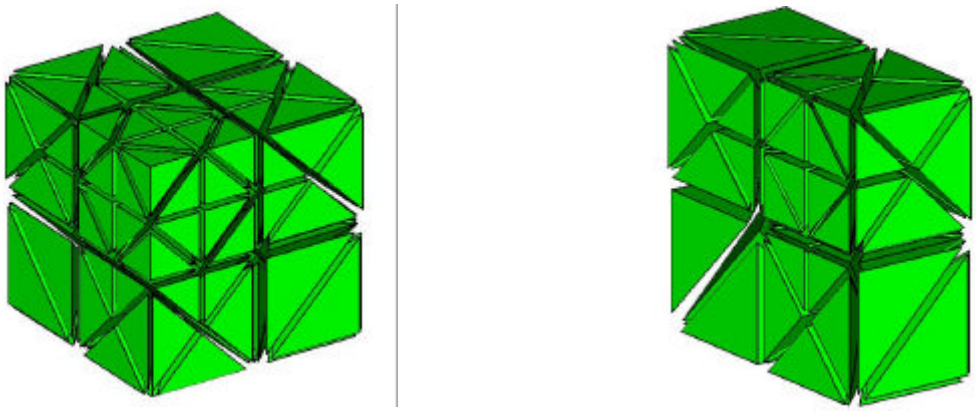


Figure 4: Non-uniform subdivisions in 3D. On the right, the rear half of the mesh is shown. The vertex on the middle of the upper middle octree edge triggers insertion of midpoints in the incident octree facets and cells (rear upper octree cell)

2.1.3 The generalized volumetric marching tetrahedra algorithm

The basic algorithm for non-uniform mesh generation does only allow grids with 'staircase' boundaries, parallel to coordinate planes. While -- strictly speaking -- the input voxel image does not provide more geometric information, the underlying 3D geometry typically has smooth boundaries. In order to achieve a better representation of such smooth boundaries, a variant of the volumetric marching tetrahedra algorithm has been designed and implemented.

The basic idea of this algorithm is to cut tetrahedra at the "smooth" interface between two materials. In contrast to most approaches known from literature which assume an underlying Cartesian grid with an implicit triangulation as indicated by Figure 3 the core algorithm can cope with arbitrary triangulations, in particular those arising from the non-uniform tesselation algorithm.

Here, we have restricted ourselves to the case that our geometry consists of exactly 2 material subsets M_1 and M_2 which have to be separated. In this case, the boundary between the two materials can be modeled as the zero-surface of a continuous function *f*.
For the multi-material case, the approach would have to be extended by assigning vectors of probabilities to vertices.

Several problems have to be solved, in order to implement the simple idea of cutting tetrahedra:

1. Starting from the binary cell classification, we have to define a smooth separating function f, with $f(x) < 0 \Leftrightarrow x \in M_1$, $f(x) > 0 \Leftrightarrow x \in M_2$, and $f(x) = 0 \Leftrightarrow x \in \overline{M}_1 \cap \overline{M}_2$
2. Given the function f, tetrahedra have to be cut where f = 0, and both halves of cut tetrahedra have to be triangulated consistent with their neighbors.
3. We have to ensure that the cut surface is in the interior of the mesh, that is, in the case of smoothing the outer mesh boundary, we have to create and additional layer of elements with background material.
4. It has to be ensured that the quality of the resulting tetrahedra cannot be arbitrarily bad.

Defining a separating function on vertices
For defining a suitable separating function f, we adopt an averaging of cell values on their incident vertices. Defining f provisionally on cell centers with $f(c) = -1$ if $c \in M_1$ and $f(c) = +1$ if $c \in M_2$, we can use the dual grid with nodes at the cell centers to interpolate these values on the vertices. On midpoints v of octcells c, we set $f(v) = f(c)$. Thus, f is defined on all vertices of the mesh, and can thus be linearly extended to all tetrahedra.

Cutting the tetrahedra
Given the function f defined on all vertices, we can classify each vertex as (a) positive (>0, above threshold), (b) negative or (c) zero, and each edge as either
(a) positive (all vertices ≥ 0),
(b) negative,
(c) cut (one vertex >0, one <0), or
(d) zero (both vertices =0).
Only cut edges (case (c)) are split by inserting a *cut vertex*.

Note that the case where vertices have a value exactly zero may seem to be a case with probality zero. This is only true if the underlying function f is arbitrary. In our case, however, f is derived from a binary labeling of underlying Cartesian cells, and these cases *will* occur (in fact, there will only be a small finite number of different vertex values at all). Thus, it is indispensable to consider this situation up-front. Handling these situation can also be used to avoid very small edges in the general case, by declaring a vertex 'zero' below a certain threshold.
Given the vertex classification, each tet can be classified by a tuple (n_+, n_0, n_-), where e.g. n_+ is the number of positive vertices. We can distinguish 5 main cases, counting symmetric ones only once (see also Figure 5).

1. $n_+ = 0$ or $n_- = 0$:
    a. $n_0 < 4$: This is the trivial case, the tet does not need splitting
    b. $n_0 = 4$: Still we assume no splitting is needed, but we cannot deduce from the vertex values which material the tet has. In our case, we use the material label of the underlying octree cell.
2. (3,0,1), (1,0,3): The isosurface cuts off one corner of the tet. Cut results in one tet (OK) and one prism (to be subdivided).
3. (2,0,2): The isosurface separates 2 opposite edges, one positive, one negative. Cut results in two prisms (to be subdivided).
4. (2,1,1), (1,1,2): The isosurface cuts off a tet through the two cut-vertices and the zero vertex. The cut results in one tet (OK) and one pyramid (to be subdivided).
5. (1,2,1): The isosurface passes through the two zero vertices and the cut vertex, dividing the tet into two new tets.

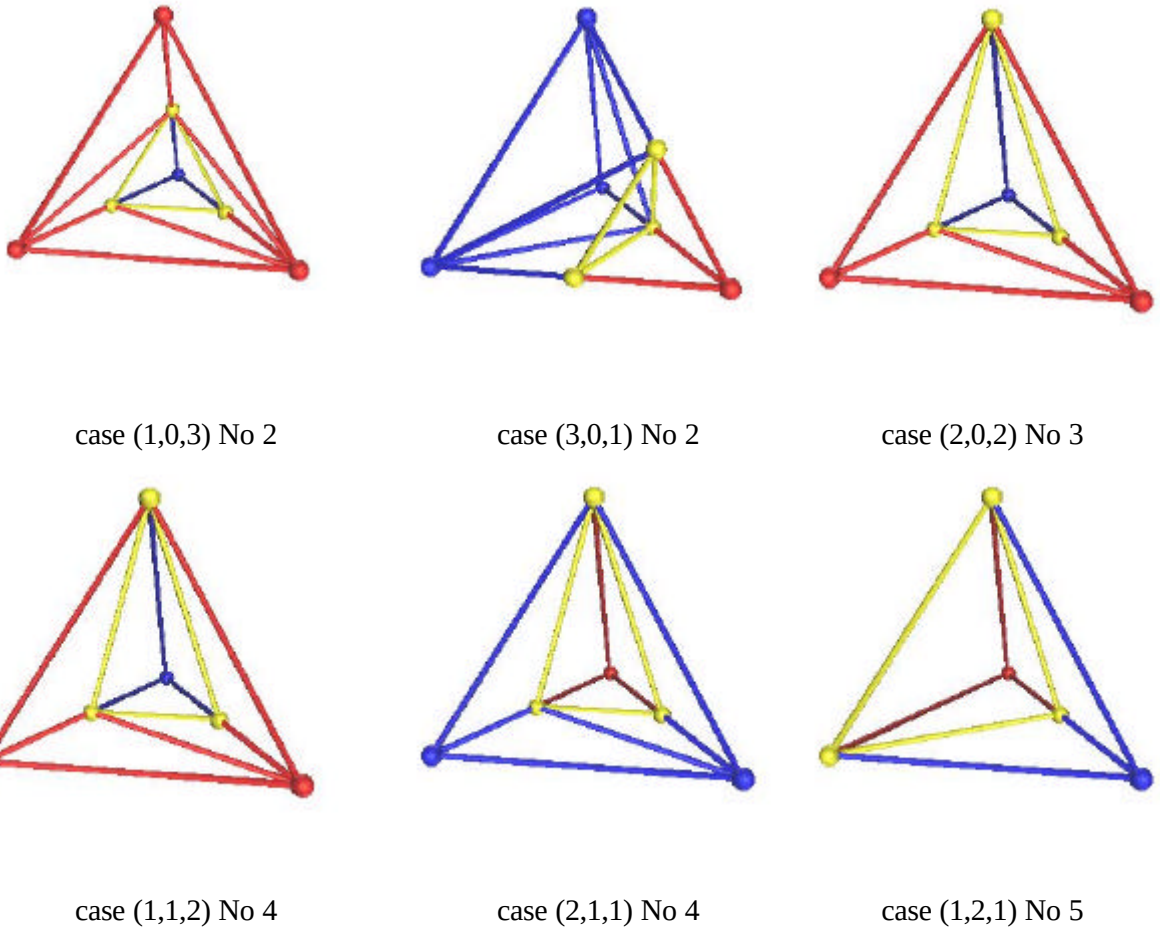


Figure 5: Configuration for vertex signs: Blue is positive, red is negative, magenta is zero

In the case of prisms or pyramids, which must be subdivided if we want a purely tetrahedral mesh, we have to ensure that the neighboring cell (which will also be a result of a subdivided tet) is subdivided in the same way. As there are subdivision of a prism's surface which do not allow a tetrahedrization of the prism without Steiner points, we must find a facet-based solution which also allows tetrahedrization. Fortunately, there is such a rule: If we impose an (arbitrary) order on the original vertices of the mesh, we choose in a quadrilateral face always that diagonal which is incident to the smallest vertex. The smallest vertex of the whole prism will thus have two diagonals, a configuration which always allows tetrahedrization. In the case of a pyramid, there is no problem, as either choice of diagonal in the quadrilateral face allows tetrahedrization.

Quality of the resulting mesh

The algorithm described before does guarantee a minimum quality of the resulting tetrahedra. This is evident from the following observation: First, the non-uniform meshing algorithm does generate only a finite number of different tetrahedra, the worst-case quality depending essentially on the balancing level of the octree, see Figure 6. Second, the separating function f can take on only value in the finite set {0, +/-1/4, +/- 1/2, +/- 3/4, +/- 1} Thus, also the marching tetrahedra algorithm does generate only a finite number of configurations. While an analytical investigation of the occuring case has not been performed, numerical studies show that the quality of resulting element is acceptable (using the mesh quality tool described below).

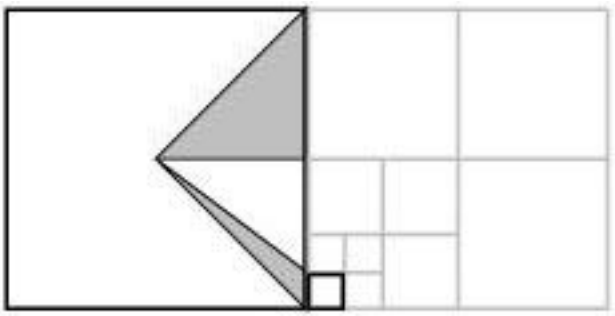


Figure 6: Worst cases in 2D for different octree balancing level L (gray triangles): L=1 (top) and L=3 (bottom)

2.1.4 Smoothing the resulting mesh at boundaries

The surface vertices may be shifted in order get a yet better/smoother approximation of the isosurface. The currently adopted heuristic is to move each surface towards the centroid of its adjacent surface vertices. This can be done in an iterated fashion, using a damping factor which gives the ratio between the calculated centroid and the actual motion.

2.1.5. Additional functionality of VGrid

In order to control some additional attributes of the resulting meshes, vgrid supports the following operations

- Setting fixed boundary conditions via the -constraint option
- Setting standard material ID's via the -simbio_mat_ids option, according to the table in D1.2b.

These options are explained in detail below (see Section 3.1.2 and 3.1.3 respectively).

2.2 The Mesh Templates Tool

Despite its ability to generate finite element meshes rapidly and automatically, one limitation of VGrid that was identified early in the Simbio Project lifetime was its inability to generate meshes that represented smooth surfaces adequately. For meshes of articulating structures, such as are found in the knee joint, this leads to serious problems between elements defining contact surfaces and their capacity to model joint motion sensibly.

As has already been discussed in Section 2.1.3, one approach to this problem was to develop the marching tetrahedra algorithm. To ensure a successful project outcome, the mesh template mesh generation method was developed in parallel. The approach relies on utilising the registration tools developed under Subtask 1.1 and described in D1.1c to morph a pre-existing mesh. Although considerable effort is expended in creating the template mesh initially, much of the effort in ensuring smooth surfaces and good element quality is rewarded by the subsequent automatic generation of patient-specific meshes that have inherently smooth surfaces.

The non-rigid registration algorithm that underpins the mesh template approach can be applied to images and meshes. Where it is applied to images it enables automatic segmentation as has been discussed in D1.1c. To aid clarity, the theory relating to the algorithm in its application to mesh generation, is repeated below.

Non-rigid registration algorithm

The underpinning theory of the non-rigid registration algorithm is based on the registration equation.

$$f(r) - m(r) = \frac{1}{2}\left[\Delta u(r)\frac{\partial f}{\partial r} - \Delta u(r)^{-1}\frac{\partial m}{\partial r}\right]$$

Where f is the fixed or target image, m is the moved or starting image, $\Delta u(r)$ is the mapping function which maps m to f and $\Delta u^{-1}(r)$ the inverse function which maps f to m. Making the assumption that $\Delta u(r) \approx -\Delta u(r)^{-1}$ and (gradient of f ~ gradient of m) it is possible to reduce this equation to either of the following:

$$f(r) - m(r) = \Delta u(r) \frac{\partial f}{\partial r} \quad \text{or} \quad f(r) - m(r) = -\Delta u(r) \frac{\partial m}{\partial r}$$

With respect to applying the resultant mapping functions to a finite element mesh it is a desirable quality of the mapping to have a one-to-one relationship. This has also been demonstrated by others to produce a more robust solution, and implemented by calculating the inverse mapping simultaneously and ensuring they are the true inverse of each other. The inverse calculation can be time consuming, so checks have been implemented to analyse the mapping based on the determinant of the Jacobian (detJ) matrix for each element. If detJ becomes negative then the mapping for that element is no longer one-to-one. In which case, the stiffness of the smoothness constraint is increased automatically to the point where the element no longer collapses. The adaptive stiffness approach means that the selection of an appropriate lamda is far less critical, as lamda will be modified locally to ensure the mapping function integrity.

Intensity offset is an additional degree of freedom that has been added to the system of equations, which permits the algorithm to modify the intensity of the images to match each other. The feature is a valuable addition especially when dealing with MRI images, which commonly have some degree of inhomogeneity in their intensities due to magnetic field deviations. The addition has been demonstrated to work and can produce good results even when the intensities in the image data set vary to a significant degree.

2.3 Evaluation of the ZMD Tool

2.3.1 Introduction

Sheffield University provided ESI with several segmented images of the knee. These images, in RAW format, contain 512*512*60 voxels which size is 0.35*0.35*2 mm, which is around 64 million voxels per cubic millimetre. By “segmented images” is meant that each voxel has an intensity equal to 0 (no tissue) or 1 (tissue).
The following images are provided:

- Femoral bone
- Femoral articular cartilage
- Medial meniscus
- Lateral meniscus
- Medial tibial articular cartilage
- Lateral tibial articular cartilage
- Tibial bone
- Patellar bone
- Patellar articular cartilage.

The goal is to evaluate how automatic it is to generate a mesh using Pam-Flow™ mesher (PamGen3D™) and how accurate this mesh will be for the kind of simulation that is intended to be performed. So, at first, a general strategy is described and discussed.

These images are first blurred using zmd bluring tool, with a kernel of 5. Once this was performed, an iso-surface of 0.1 is computed. It generates a 3D surface that will be the support of the mesh. Figure 7 shows different blurring levels of a structure and the associated generated surface.

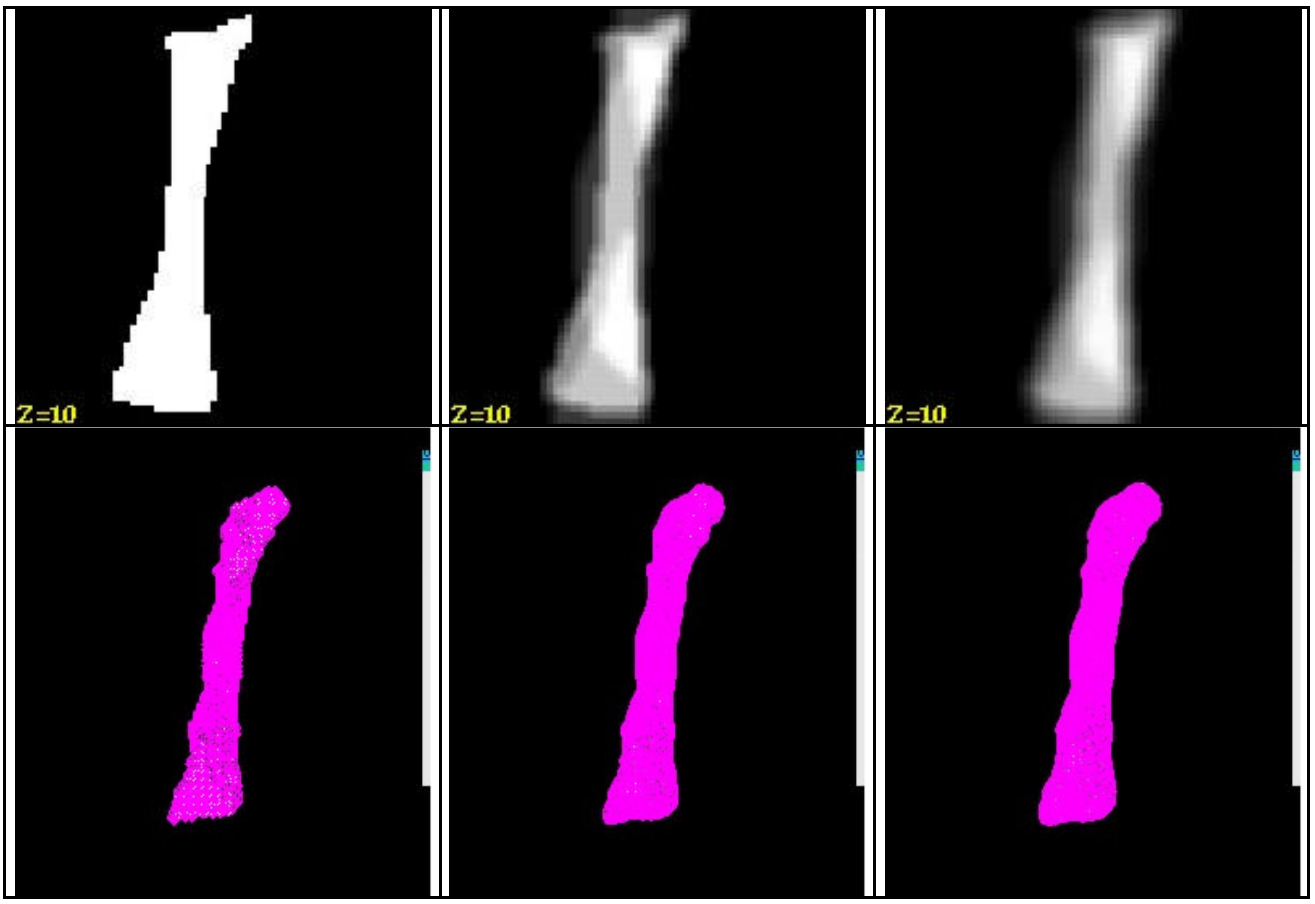


Figure 7 : Different levels of blurring of the same image and the resulting extracted surface

It is necessary at first to process the surface that has many sharp edges. It is smoothed in Zmd (50 iterations, Kph=-1, no-shrink flag on). For the given size of the elements, and the size of the mini-structures one wants to get rid of, a negative Kph factor is very efficient.

The processed surface is then exported into PreFlow™ for generation of the background grid, that will define the mesh properties. The meshing parameters are investigated by attempting to generate many meshes of the same initial surface and compare their properties. The meniscus being a critical part of this study, the lateral meniscus is chosen to be under focus.

2.3.2 Generating lateral meniscus meshes

As described in the later part, the image is blurred and an iso-surface of 0.1 is extracted. This 3D-surface is then smoothed 50 times with a K-factor of −1 (un-shrink flag on), and one time with an alpha factor of 1 (un-shrink flag off) to get a better triangulation. The 3D surface is fed into PreFlow™ in order to generate the background grid and specify element and source sizes. Then the volumic tetrahedral mesh is computed by PamGen3D™.

The following input parameters are of interest:
- maximal element size,
- Discrete surface sharp angle,
- Discrete Surface boundary angle,
- Number of points and background grid points necessary to perform the meshing,
- Source size.

The following output parameters are chosen to evaluate the quality of the mesh:

- Number of generated elements,
- Minimal, maximal and mean sizes of the elements,
- Computation time.

Two different computation machines are used: a 16-processor R10000 origin 2000 at 195 MHz, hereafter designed as “o2000”, and a 2-processor R10000 octane Duo at 175 MHz, hereafter designed as “psioct3”. All calculations are performed on one processor.
Thus four calculations are performed in order to evaluate the best set of parameters to be used for the rest of the structures. Results are displayed in Table 1.

		Comp. 1	Comp. 2	Comp. 3	Comp. 4
INPUT	Maximal element size (mm)	1.37	1.37	0.36	0.36
	Discrete surface sharp angle (°)	11	11	No angle defined	7
	Discrete surface boundary angle	11	11	No angle defined	7
	Number of Background grid points used	80,000	80,000	100,000	80,000
	Number of points used	320,000	320,000	400,000	320,000
	Source size (mm^2)	No source defined	1.5	0.5	No source defined
OUTPUT	Number of generated elements	1,101,000	1,638,561	2,105,832	1,107,000
	Calculation time (minutes)	54	81	64	33
	Machine used	Psioct3*	Psioct3*	O2000*	O2000*
	Minimal edge length (mm)	0.028	0.018	0.02	0.029
	Maximal edge length (mm)	1.13	0.86	0.73	1.0
	Average edge length (mm)	0.24	0.21	0.2	0.24
	Minimal volume (10^{-3} mm^3)	0.057	0.053	0.050	0.057
	Maximal volume (10^{-3} mm^3)	325	200	95	270
	Average volume (10^{-3} mm^3)	12	8.1	6.3	12

Table 1 : Mesh results

All these meshes were generated within around 1 hour calculation time (plus one hour engineering work on the image and surface). Nevertheless, being over one million elements, these meshes are unsuitable for computational structure dynamics using explicite finite element software. PamGen3D™, being specific to numerical flow applications, where number of element is not the primary issue, doesn’t focus on reducing the number of elements. A challenge in the future would be to try and reduce these nearly automatic generated meshed number of elements by some new algorithm. An other challenge is to check if the work methodology used with the meniscus can be applied on the other images.

2.3.3 Other meshes generated

The same methodology is applied to different structures. For lateral tibial articular cartilage and for medial meniscus, meshes are generated with characteristics described in Table 2.

	Lateral tibial articular cartilage	Medial meniscus
Number of elements	1,500,000	1,480,000
Computing time	45’	43’
Machine used	O2000	O2000
Minimal edge length (mm)	0.013	0.012
Maximal edge length (mm)	1.2	1.28
Average edge length (mm)	0.17	0.23
Minimal volume (10^{-3} mm^3)	0.013	0.0039
Maximal volume (10^{-3} mm^3)	400	610
Average volume (10^{-3} mm^3)	17	11.5

Table 2 : Other structures mesh characteristics

These two structures are displayed in Figure 25 and 26. For the other structures, it was impossible to generate a full 3D tetrahedral mesh. The quality of the surface probably is the limiting factor. The incriminated surfaces have to be reworked but that means no automatic generation is possible for those structures. Furthermore, the generated meshes would have a very high number of elements.

2.3.4 Conclusion

An evaluation of PamGen3D™ and its application to the SIMBIO project was performed.

First, the segmentation tool Zmd was described. This program being an intensity based image processing software, its main application is the automatic segmentation of arteries. For the knee, though the whole process is not automatic, it was possible to extract in a few steps the outer surface of the main structures such as bone (T2 sequence), ligament (T2 sequence) and cartilage (T1 sequence) with a good quality. Regarding other structures such as cruciate ligaments, separation between cartilages or menisci, the 256*256*60 images provided (resolution 0.7 mm*0.7 mm* 2 mm) do not contain enough information to allow as accurate a segmentation.

In the future, more precise scans of the knee will be performed at a 512*512 resolution. These scan may enable an easier segmentation of these small structures. The extension of SIMBIO to other structures than the knee could also be performed by that tool.

Second, an automatic sequence of operations was designed for meshing segmented images of the knee into 3D tetrahedral elements. This sequence was used to generate meshes of 3 structures, while proved inefficient for 7 others. The level of manual work and try-fail attempts necessary with these remaining structures will decrease interest in this software, as the goal is an automatic generation. The few generated meshes anyway can in no way be used in the domain of explicite finite element calculations. This is due to the fact that the mesh generator tool being designed for flow applications doesn't strive on reducing the number of elements but on following the surface geometry. Making these surfaces smoother requires a lot of manual work.

2.4 Auxiliary Tools

2.4.1 Mesh Quality Tool

Ensuring sufficient quality of mesh elements is crucial for obtaining accurate numerical results in an efficient way. The quality measure is based on the condition number of matrices formed by the directions of edges around a vertex of a cell. This measure is dimension independent and applies to all cell types which are *simple* polytopes, i.e.each vertex has three incident edges on the cell, like hexahedra and tetrahedra, but unlike pyramids. The method measures the deviation of a cell corner from the "ideal" corner of the unit cube. For tetrahedra, a weighting matrix has to be included, transforming the corner matrix of an equilateral tetrahedron to the corner matrix of the unit cube, because the "ideal corner" in this case is that of an equilateral tetrahedron.

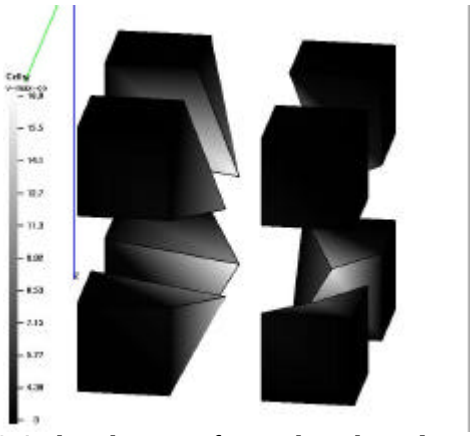


Figure 8:A shrunk view of a mesh with quality measure of vertices (light means bad quality)

More formally, let v be a vertex of a cell c, and v_1, v_2, v_3 be the 3 vertices of c adjacent to v. We set $e_i = x(v_i) - x(v)$, x being the vertex coordinates, and $A_c(v) = (e_1,e_2,e_3)$ (cf. Figure 8). Then we calculate the quantity

$$K_c(v) = \kappa(A_c(v))$$

where $\kappa(A) = \|A\| \|A\|^{-1}$ is the condition number, and $\|.\|$ is a freely choosable matrix norm. The quality of a vertex / cell is defined as

$$Q(v) = \max_{c \text{ incident to } v} K_c(v)$$

$$Q(c) = \max_{v \text{ incident to } c} K_c(v)$$

The mesh quality module has been used successfully to detect the degenerate tetrahedra produced in some cases by the original version of the non-uniform meshing algorithm.

2.4.2 Mesh visualization by shrunk view

It notoriously difficult to fully visualize and understand the structure of a 3D mesh. In order to visually assess the quality of meshes generated with Vgrid a specific tools has been designed which shrinks each cell towards its center, allowing to look inside a mesh to a certain extent (see Figure 8), which shows a shrunk view of a 2x2x2 mesh, overlaid with mesh quality information.

2.4.3 Additional Simulation-specific pre-processing

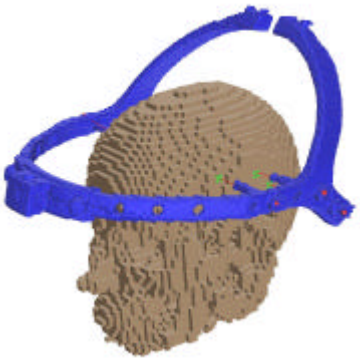
For many types of simulation, a mesh must fulfill additional properties, and problem-specific boundary conditions must be set.

Setting external forces boundary conditions (Halo tool)

Handling boundary conditions for FEM models can be a tedious task involving lots of manual labor. In the maxillo-facial surgery application we want to calculate the stress distribution in a human skull to which a so-called *halo device* is fixed with screws (see Figure 9) The halo geometry is given as voxel image, and meshed using Vgrid. The task consist of getting the forces exercised by the halo screws on the skull, which involves finding the locations where the screws would penetrate the skull, given a fixed position of the halo. We break down the problem into the following steps:

1. Determine the halo and screw data from the voxel-based halo mesh
2. Extract the skull surface from the volume mesh built from the voxel data
3. Find the intersections of halo screws with the skull surface
4. Find the vertices or cells in vicinity of the intersections which are intersected by the corresponding cylinder, and set corresponding boundary conditions.

A halo device for maxillo-facial surgery



Skull mesh with calculated halo intersections

Figure 9:Halo positioning problem in head mechanics

This approach resulted in a suite of tools, which allow to set the forces exerted on the skull with a minimum of user input, describing the relative positioning of the halo with respect to the head.

2.4.4 Filtering connected components

Disconnected components in a mesh which have no fixed nodes (zero displacement) result in an ill-posed problem for elasto-static or -dynamic problems, resulting in worsened convergence or no convergence at all. (Here, two cells are considered connected, if they share a common face.) Unfortunately, such components cannot be excluded, given the noisy image input data.Therefore, a tool filter-components has been designed which strips all but the largest mesh components.This tool is used to preprocess data for the HeadFEM simulation program, and it has proved invaluable especially when a reduced model consisting of only skull is employed.

2.4.5 Surface smoothing tools

In addition to the mesh template approach for the generation of patient specific finite element meshes, several tools have been developed to aid in the generation of the template mesh. The tools centre on producing smooth surfaces that do not contain 'terracing artefacts', which result from pixelation of the data in the medical images, as shown in Figure 10.

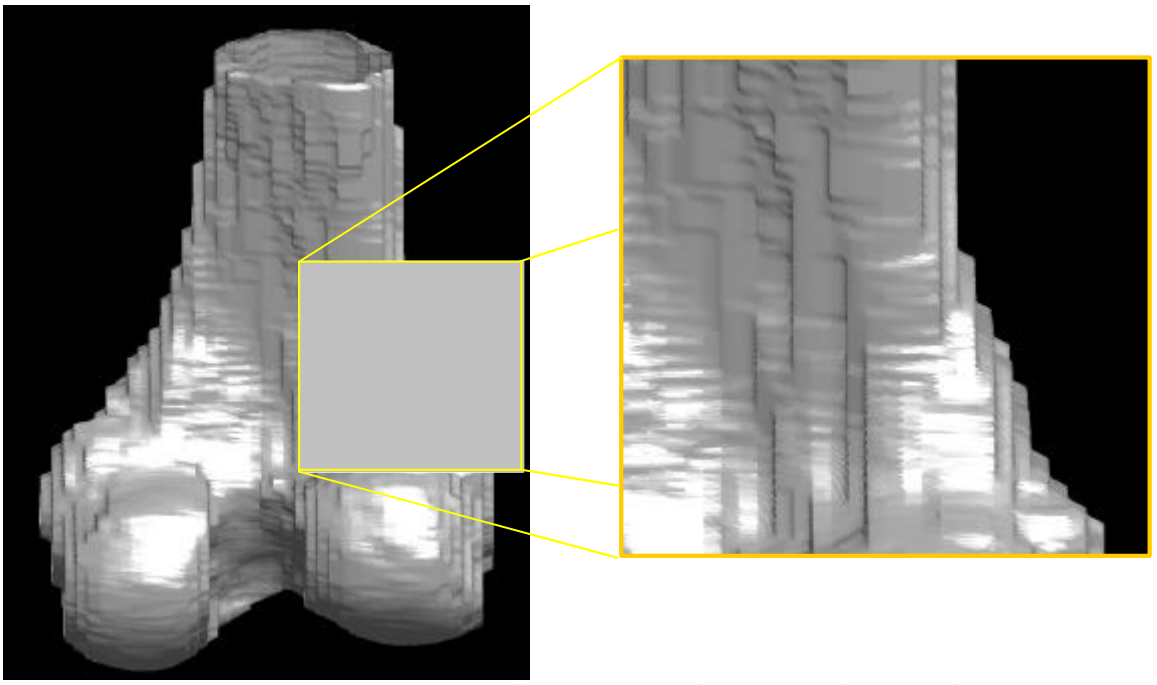


Figure 10: An example of terracing artefacts on the femur surface.

To remove such terracing problems, a non-linear registration algorithm has been implemented in 2D, similar to that used in vreglocal3d. The algorithm is used to register adjacent slices of the volume, which produce points of correspondence on each contour. Given this mapping function it is simple to specify a mesh density on the starting slice and allow the mesh to propagate across the surface of the object. A resultant sample smoothed surface of a femur is shown in Figure 11.

After the 2D mappings have been generated, it is possible to implement rules based on the quality of the meshes produced, which are desirable for surface finite element meshes. The foremost being that the size of an element may not change by more that 30% between adjacent elements. Such a mesh quality constraint helps reduce the potential for numerical conditioning problems. The surface mesh produced has been rendered as a finite element mesh in Figure 12, and appears to be of good quality.

The code developed will be available on request to any of the Simbio Project partners who wish to use it. However, because it is not a specified component of the SimBio environment, it will be supplied

on an ‘as is’ basis only without formal support. To run the code a copy of Matlab5 or greater is required, along with the ability to compile Mex files.

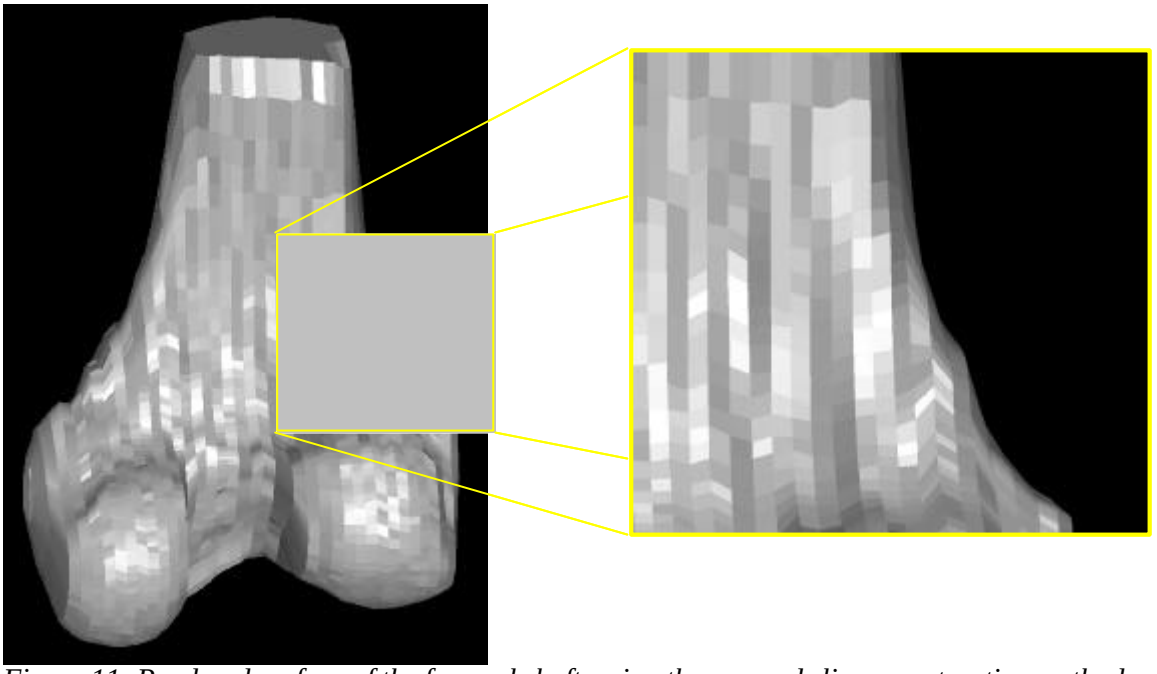


Figure 11: Rendered surface of the femoral shaft, using the mapped slices construction method.

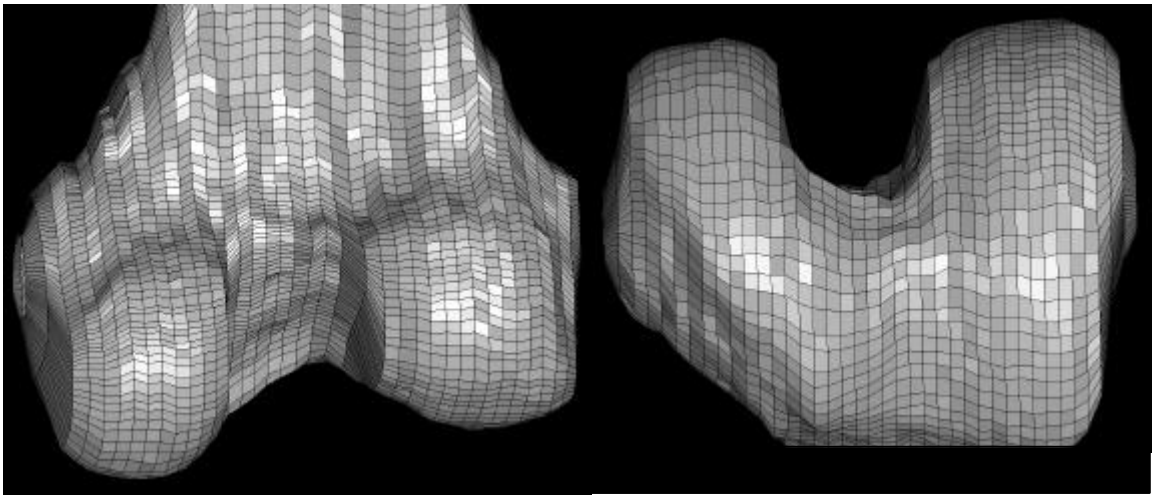


Figure 12: The surface mesh produced automatically by the algorithm, is of sufficient quality to be used in a finite element analysis.

3. How to use the Software

3.1. VGrid

3.1.1 Command Line Switches

The VGrid mesher is part of the Workpackage 1 software. The entire WP1 package is downloadable via the protected part of the SimBio webpage.

To get an idea of the options avalaible in the final release one can simlply type

```
./vgrid
```

and as a result one obtains a list of command line switches together with their possible parameters. The following lines show the output of the ./vgrid command:

-help
Prints usage information.

-in <string>
Input file. Default: (none)

-out <string>
Output file. Default: (none)

-elem cube | tetra5 | tetra6a | tetra6b
Primitive type. Default: cube

-min <integer>
Minimal grid resolution. Default: 4

-max <integer>
Maximal grid resolution. Default: 4

-smooth no | shift | marching
Smooth boundaries. Default: no

-shift <number>
Degree of node shifting (range 0-0.49). Default: 0.49

-surface [true | false]
Surface mesh output. Default: false

-np <integer>
# of partitions. Default: 1

The in Option

The -in option specifies the input file for the VGrid mesher. The input file can be any segmented dataset in the SimBio VISTA format. One can also feed MRI raw data into the programme but a sensible material property assignment is critical in this case.

The –out option

The -out option specifies the name of the VGrid VISTA output file. The format of VGrid output files has been discussed in detail in Section 2.1.4. VGrid output files can be visualised with the SimBio VM tool (see D5b).

The –elem option

The –elem option specifies the type of finite elements. Currently it can be chosen amongst the following types:

cube	hexahedral elements
tetra5	tetrahedral elements
tetra6a	tetrahedral elements
tetra6b	tetrahedral elements

The tetrahedral meshes differ from each other in the scheme used to to create tetrahedra by subdividing cubical elements (see D1.2a). In the next release an additional option will be the

hybrid	mixed mesh type

switch. Choosing this options yields mixed meshes consisting of tetrahedra and hexahedra. By default this switch is set to cube as element type.

The –min and –max option

By specifying the min/max options the VGrid user defines the spatial resolution of the mesh. Both options give the minimal and maximal edge length of the cell mesh (phase 2 of the algorithm). The parameters thus govern the number of elements produced. If small structures are present in the image the user is advised to choose a small value (2 or4) for min. Please note that the parameters for min/max can only be **integer numbers** as the edge length is measured in the number of voxels the cell is covering. By default min and max are set to the value 4.

The –smooth option

As explained in D1.2a the VGrid user can choose between different strategies to smooth mesh surfaces.

no	no smoothing
shift	nodeshifting is switched on
marching	the marching tetrahedra algorithm is used.

By default *no smoothing* is chosen.

The –shift option

If the –smooth option has been to *shift* (see above) the value of this option becomes relevant. The user gets the chance to influence the degree of nodeshifting. The maximal permitted value for shift is 0.49. A value of 0.5 would cause the creation of degenerate hexahedra. By default the value of –shift option is set to 0.49.

The surface option

Here the user decides whether he wants a surface mesh or a volumetric mesh as output. This is boolean parameter that can only take on the values *true* or *false*. The latter is set by default which means that volumetric meshes are generated if nothing is specified.

The –np option

The user has to specify already at this early stage on how many processors his application will run. This switch is important for creating an initial partitioning that is optionally improved by the DRAMA tool. The parameter for –np must be an integer number. By default it is set to 1.

3.1.2 Format of the material file

The material file allows to set meshing options specific to both materials and material interfaces, that is, pairs of materials. It also allows to map voxel values to materials. Materials are described by a material line, starting with the keyword `material`, and interfaces are described by an interface line, starting with the keyword `interface`. Both types of lines may be freely mixed.
The format of a material line is as follows:
`material` *name* max low-voxel high-voxel meshing-type weight here the parameters mean the following:

- [name] Type: string. The name of the materials. If the option `-simbio_mat_ids` is used, this name should coincide with a name defined in the `materialDB` class (see also D1.2b, page 6)
- [max] Type: Integer > 0. The maximal edge length of an octree cell for this material. Actual octree cells may be smaller e.g. near interfaces.
- [low-voxel, high-voxel] Type: integer in [0,255]. A range of voxel values representing the material in the image. For a single voxel value, set *low-voxel =high-voxel*

1. [meshing-type] The type of meshing for this material. Currently, two values are supported: meaning 'default meshing', i.e.use the global meshing type defined at the command line (by `-elem`, `-smooth` and values for *max*)
2. no meshing. This is the default for material corresponding to voxel value 0 (background).

- [weight] Type: floating point number. Default: 1.0. Weight is used to determine the dominating material in a super voxel, the number of occurences of the material is multiplied by weight before finding the maximum material. Can be used to (try to) avoid disconnected components or holes when meshing e.g.only skull.

The format of an interface line is as follows:
`interface` *material1 material2 max*

- [material1, material2] Type: string. Names of the two materials. Must match materials given in a material line
- [max] Type: integer > 0. The maximal edge length of an octree cell for this interface.

For missing interface lines, the *max* parameter will be set to the value specified by the global value specified by *-min*.
For missing material lines, the *max* parameter will be set to the value specified by the global value specified by *-max*.
For safety reasons, each material occurring in the image should be specified by a material line.

Example :

```
material  bg           16   0  0    2  1.0
material  skull        4    1  1    1  4.0
material  brain        16   10 44   1  1.0
material  scalp        8    50 50   2  1.0
interface skull  scalp 1
interface skull  brain 2
```

Comments:

- materials `bg`, `skull` and `scalp` are define by a single voxel value, whereas {`brain`} is defined by the range [10,44].
- Only `skull` and `brain` are meshed.
- Although s`calp`} is not meshed, the interface between `scalp` and `skull` get special treatment (finest resolution)
- The weight for `skull` is 4.0, which means that whenever 1/8 or more voxels of a supervoxel are labeled `skull`, the whole supervoxel will get assigned skull.

3.1.3 Format of the constraints file

FILE :box.con
Constraint ID
xmin ymin zmin (coordinates of the corner points of the box)
xdim ydim zdim
(dimensions of the box in x,y,z)

FILE:plane.con
Constraint ID
plane_dir (orientation of the plane, where 1 stands for xy, 2 for yz and 3 for zx)
plane_pos (z, x, y position of the plane)

If the specified position is not a coordinate of the mesh node list, then the algorithm is searching for the plane position closest to the specified coordinate.

FILE:snodes.con
Constraint ID

nocn (number of the nodes that follow)
x1 y1 z1
x2 y2 z2
x3 y3 z3

3.2 Mesh quality module

The mesh quality program is called by issuing ./quality <options>,where option can be:

-g input file grid input file (Vista or ASCII format)

-o output file output file (in GMV format)

-shrink f=1.0 shrink grid cells by f (default: no shrinking)

-hist h = 0 histogram size printed to terminal, for additional statistical information on quality distribution.

-maxcond m = -1 if m > 0, quality numbers larger than m are set to m and histogram shows only interval counts for values between the minimal number and m.This is useful if the mesh contains degenerate or near-degenerate element, where the quality measure tends to infinity.

3.3 Shrunk mesh view

Called by issueing <`options`
where options can be:

- `-g` input file: grid input file (Vista or ASCII format)
- `-o` output file: output file (in GMV format)
- `-s` s=1.0 shrink grid cells by s (default: no shrinking)

3.4 Setting boundary conditions and filtering connected components

For convenience, the tools involved are driven by a shell script boundary-*conditions.sh*, which accepts the following options:

-in input Input file (VISTA or HeadFEM ASCII)
-out output Output file (HeadFEM ASCII)
-forces f Strength of halo forces in Newton (N)

3.5 Mesh Templates: Practical use and limitations

Functions supplied

Registration Function	Purpose
Vreglocal3d	Performs an affine registration followed by several non-linear registrations using a multi-resolution approach (increasing mesh densities)
Transformation Function	**Purpose**
Vtransform	Applies the mapping function returned by Vreglocal3d to an image volume or mesh

Usage

Man page for Vtransform

Options :

-help **Help**
-in **Name of input file, can be an image or a mesh**
-out **Name of output file, will be the same format as input.**

-map The vista file containing the mapping function returned by Vreglocal3d.
-direction Direction in which to apply the mapping function. *forward|reverse*. Default *reverse*.

The introduction of the adaptive stiffness model has resulted in the mapping process being relatively insensitive to the value of lamda. Therefore, the release version of the algorithm runs with a lamda value of 1, which has been found to perform adequately.

The registration algorithm supplied is called `vreglocal3d`, and this code performs several registrations. It begins with a grid size of 5x5x5 nodes, and then doubles the grid density until it reaches the point where there are five pixels per element. This limit has been selected as the densest mesh that we believe would be stable numerically. The reason for this multi-resolution approach is that each element only overlaps with its adjacent neighbours. This means that for the system to behave sensibly the corresponding pixel in the target image must lie within two elements of the pixel in the source image. Therefore, if large deformations are necessary having a very dense mesh may not produce good results. By using this multi-resolution approach, the coarse detail in the images can be matched initially, and the finer features can be resolved as the mesh density increases.

Also provided is the `Vtransform` function to apply the mapping functions returned by the registration software to other images or meshes. It requires that a direction for the mapping is specified. In SimBio we have decided that the best strategy for registering the template and patient images is to register the patient image to the template. This has several advantages; firstly, a region of interest can be generated for the template image, as it does not change. The other advantage is that the mapping that takes the patient image to the template image is the one that takes the template mesh to the patient image, and will be slightly more accurate. The direction defined as 'forward' supplied to any of the mapping functions will perform the operation in the same direction as the registration. So, if the patient image is registered to the template image, it will be the 'reverse' mapping which takes the segment image or finite element mesh into the patient space.

Due to problems of interpolating binary volume data, which can result in distortion, as shown below, it was decided to map the segments across to the patient images as surfaces as shown in Figure 24.

3.6 The ZMD Tool

The ZMD software is divided in image processing, triangulated surface processing and curves processing. An image can be visualized in 3D or in 2D as seen on Figure 13.

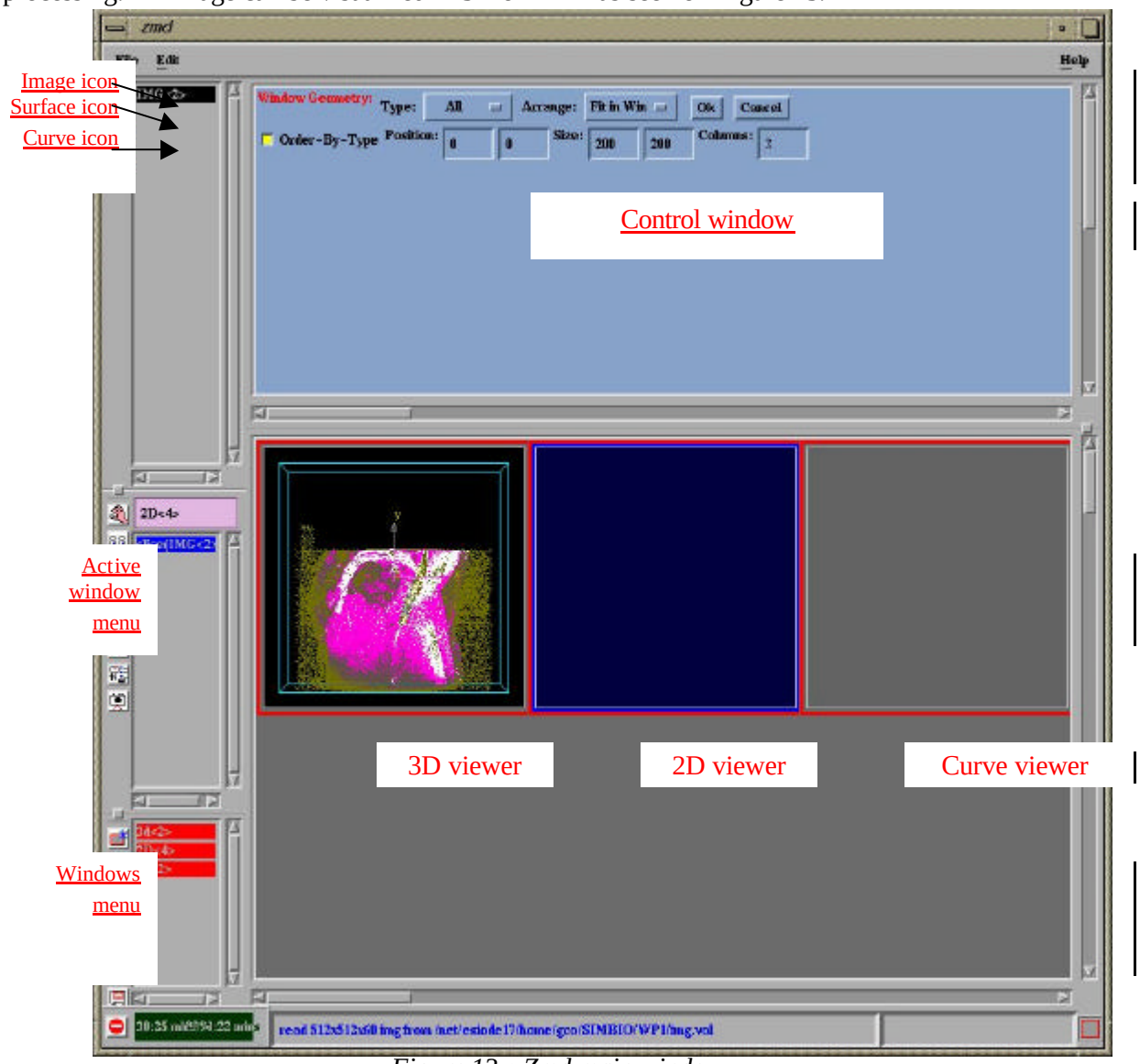


Figure 13 : Zmd main window

The image-processing menu enables to generate a triangulated iso-surface of a processed image. The surface processing tools enable to process the surface to be usable in a finite element application.

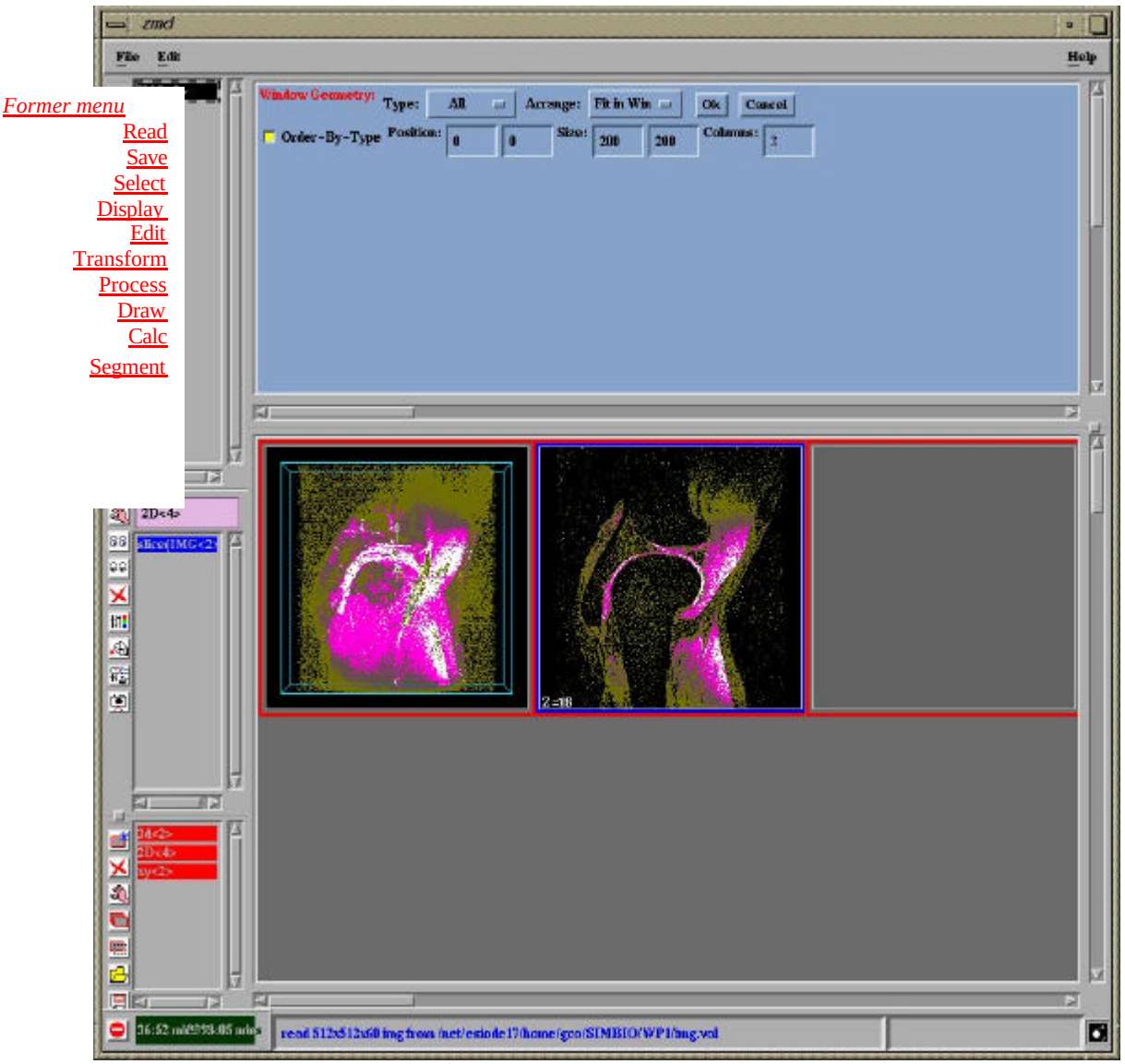


Figure 14 : Image menu

3.6.1 Image processing

The image-processing menu is seen in Figure 14.
Two types of image formats can be read: RAW and Analyse format which are two common formats found in the medical imaging world. Apart from the common **Read**, **Save**, **Select**, **Display** and **Edit** menu, one finds the following menus.

Transformation

Translating, Scaling and Flip-flopping the image are possible.

Process

The following options are available:

Negate
Negate enables to get the negative of the image.

Crop
Crop enables to pick up a pre-selected part of the image. This option is useful when wanting to separate structures.

Blur
Blur enables to smooth the image discrepancies. A kernel option permits to have different levels of blurring.

Contrast
Contrast enables to sharp or un-sharp an image.

Refine
Refine enables to split the voxels of a selected part of the image. Intensity of these voxels is then linearly interpolated from the present intensity to the interval [0;1].

Isotropize
Isotropize split rectangular image voxels in order to get cubic voxels.

Scale Min/Max
Scale Min/Max enables to scale the actual grey scale of the image from 0 to 1.

Gradients
Gradients computes two images from the current images. First the norm of the image gradients and second the norm of the second derivative of the image. The derivative are calculated according to the given kernel number.

Draw

Draw enables to paint voxels at a given intensity value.

Calc

A list of computational tools are found in this menu

Info
Info provides informations on the image.

Math
Math provides a range of operations like square, square root, sine, and cosine. They are useful for filtering the image information. Furthermore, operations between two images can be computed such as sum, difference, multiplication and division, min and max and logical operators.

ROI operations
ROI operations enables operations such as summing, averaging, taking the area, the min and max of a selected region.

Histogram
Histogram plots a curve as the number of points as the function of intensity.

Segment

The following algorithms are used for image segmentation.

Region growing
Region growing is a neighbouring technique that spreads image segmentation from seeds. A recent development enables to restrain the region growing to a given selected part. Region growing can be performed defining either a fixed range or a relative change.

Binarization
Binarization enables to set a range of intensity to a given value. This operation is useful for getting rid of some structures.

Iso suface
Iso-surface computes a surface of a given intensity. This surface can then be processed using the surface menus (see part 0).

Deformable model
Deformable model is a new algorithm that enables segmentation of tubular structures from their middle line.

3.6.2 Surface processing

The Image processing menu is seen in Figure 14.

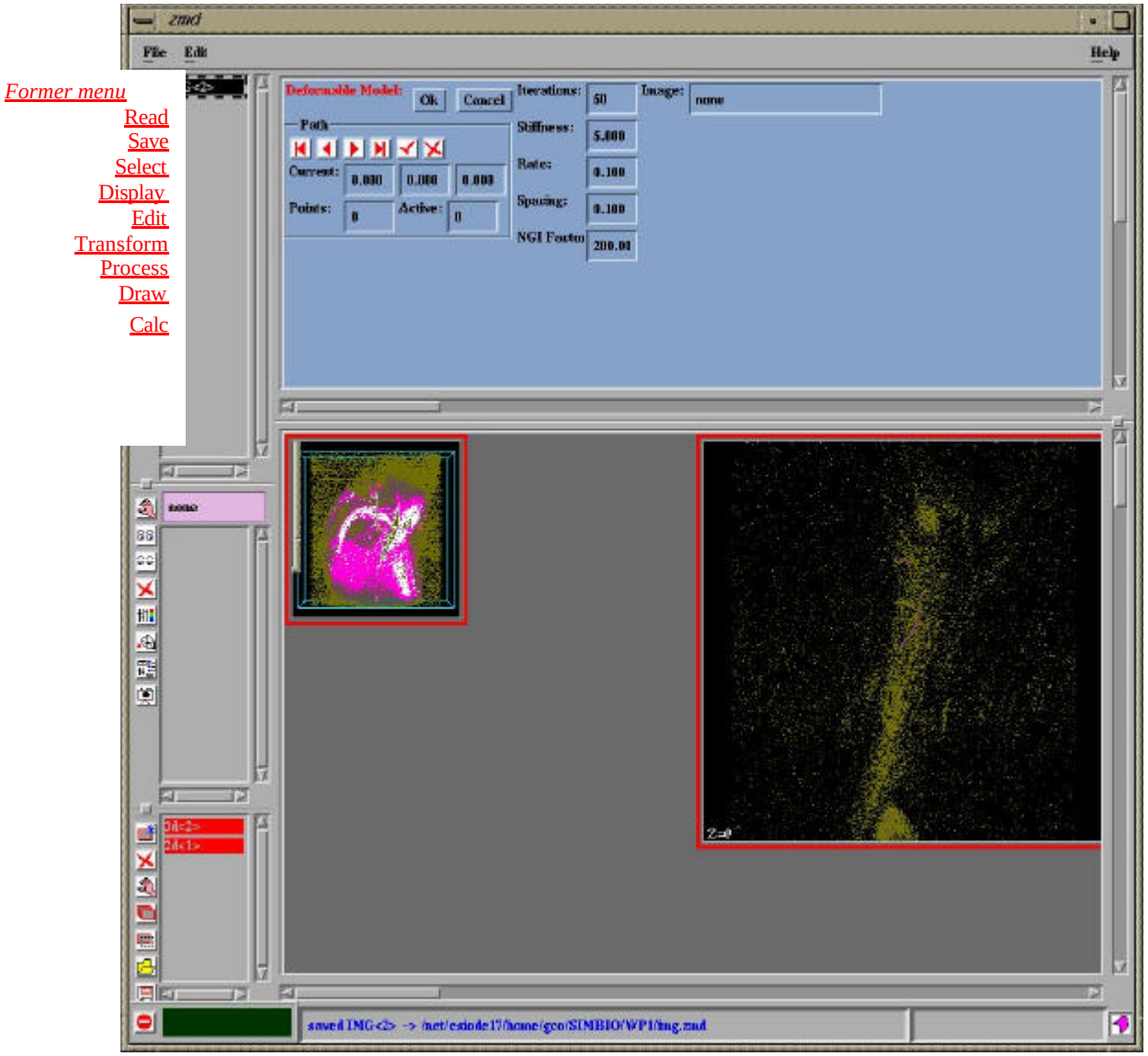


Figure 15 : Surface menu

Three types of surface formats can be read: Zfem, Gen3D, STL format which are three common formats found in the medical imaging world. The following treatment can then be applied on the given surface. Apart from the common **read**, **save**, **display** and **edit** menus, one finds the following menus.

Select

Select enables to select a given set of nodes or elements through their labels or geometrical properties, then their immediate neighbours, etc etc…

Transform

Transform enables translation of the surface or changing the orientation of their normals to get a consistent orientation.

Process

Smoothing

Smoothing applies a Laplace operator on the nodes coordinates to get a smoother structures. The coordinates of the j^{th} node x_j are thus modified to $x_j' = x_j + \text{alpha}* Dx_j$ where $Dx_j = \text{Sum}_{\text{Neighbours i}}\ w_{ij}\ (x_j - x_i) / \text{Sum}\ w_{ij}$.
The applied weights can be either the number of neighbours 1/N, the distance $1/[x_i - x_j]$ or the area $\frac{1}{2}(A_i + A_j)$ where A_i and A_j are the areas of the two triangle elements containing x_j and x_i.

It is common knowledge that such operations tend to shrink tubular structures. An inverse transformation $x_j'' = x_j' + \text{beta}* Dx_j'$ is applied to prevent this shrinking.

For very thin structures, this smoothing technique could give rise to perforation of the upper wall by the upper wall. An interface applies arbitrary force on the nodes to prevent this phenomenon.

Move points

Move points enables easy growing or thinning of the structure.

Refine

Refine, as for images, allows to refine a given set of elements.

Cut

Three options are of interest in this menu.

Plane+ROI

Plane+ROI enables to cut the given surface by a user-defined plane. It is enables further definition of boundary conditions.

Close holes

Close holes enables to close a given hole surrounded by triangle elements.

Connected components

Connected components allows to give a different label to all non-connected surfaces.

Join

Join is the contrary of connected components.

Draw

Draw enables many operations such as drawing points and elements as well as extruding a surface along a given vector.

Calc

Calc has two interesting menus

Info
Info gives some information on the selected surface.

Statistics
Statistics computes a curve of the number of elements with a given area.

3.6.3 Curves

Curves processing allows many regular operations on curves such as editing, scaling, splining, periodising or merging curves. Many mathematical operations are available too such as averaging, summing, integrating, deriving curves.

4.Results

4.1 VGrid

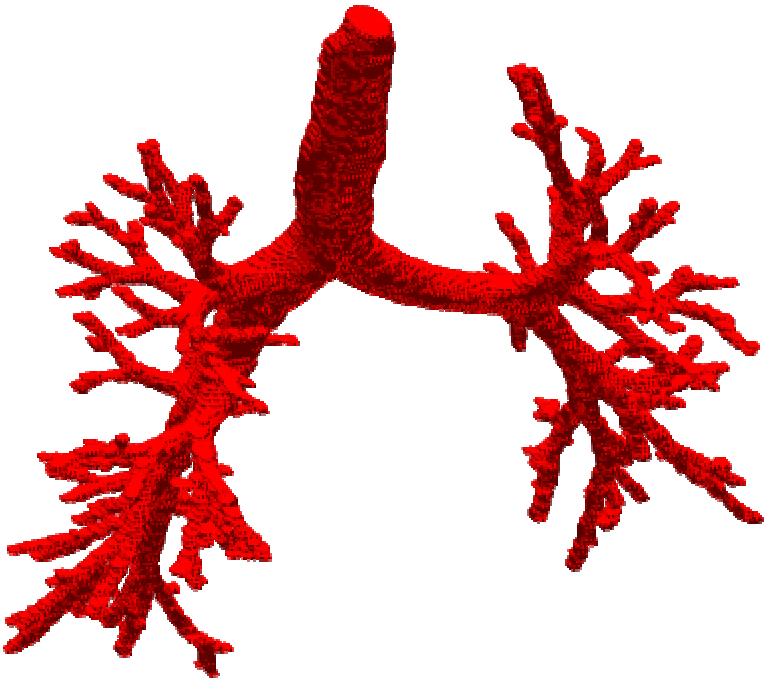


FIGURE 16: Uniform hex mesh of lower airways.

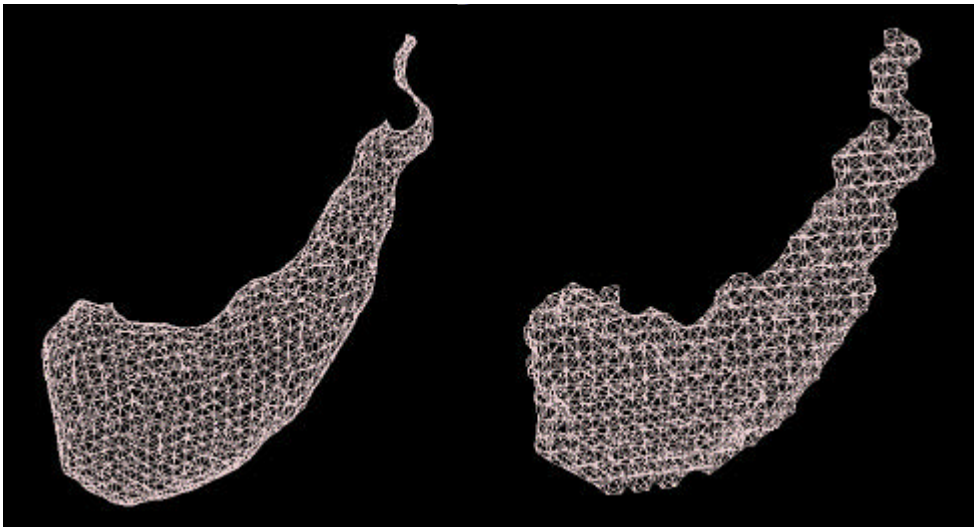


FIGURE 17: Smoothed mesh of neuroanatomical structure (caudatus nuclei)

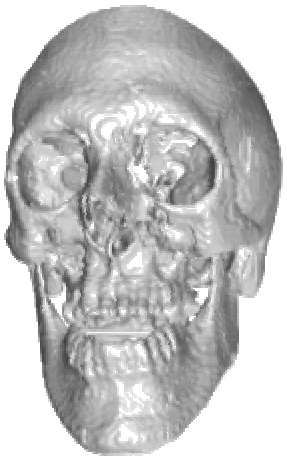


FIGURE 18: Smooth skull mesh

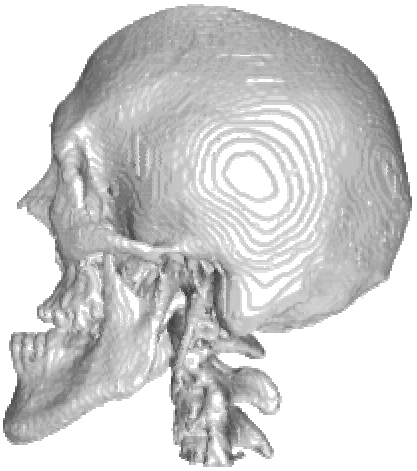


FIGURE 19: Smooth skull mesh

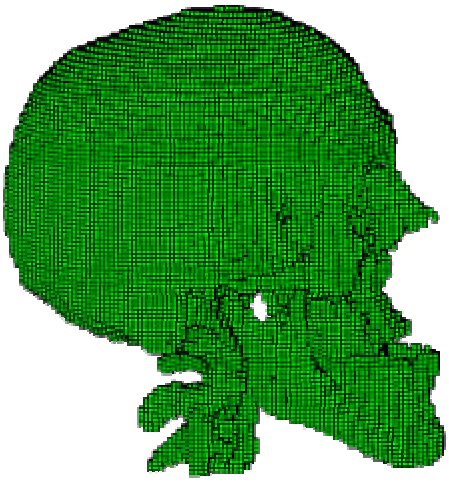


FIGURE 20: Uniform skull mesh

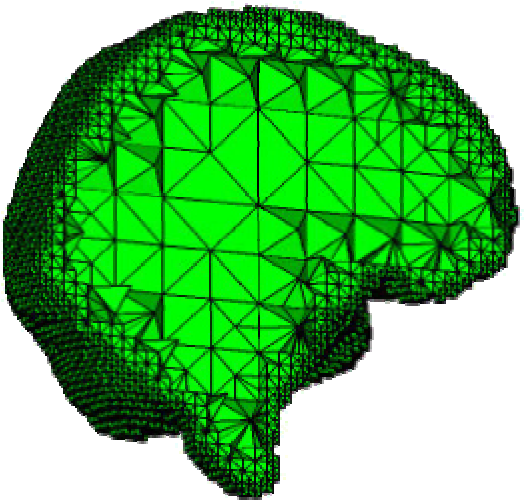


FIGURE 21: Non uniform tet mesh

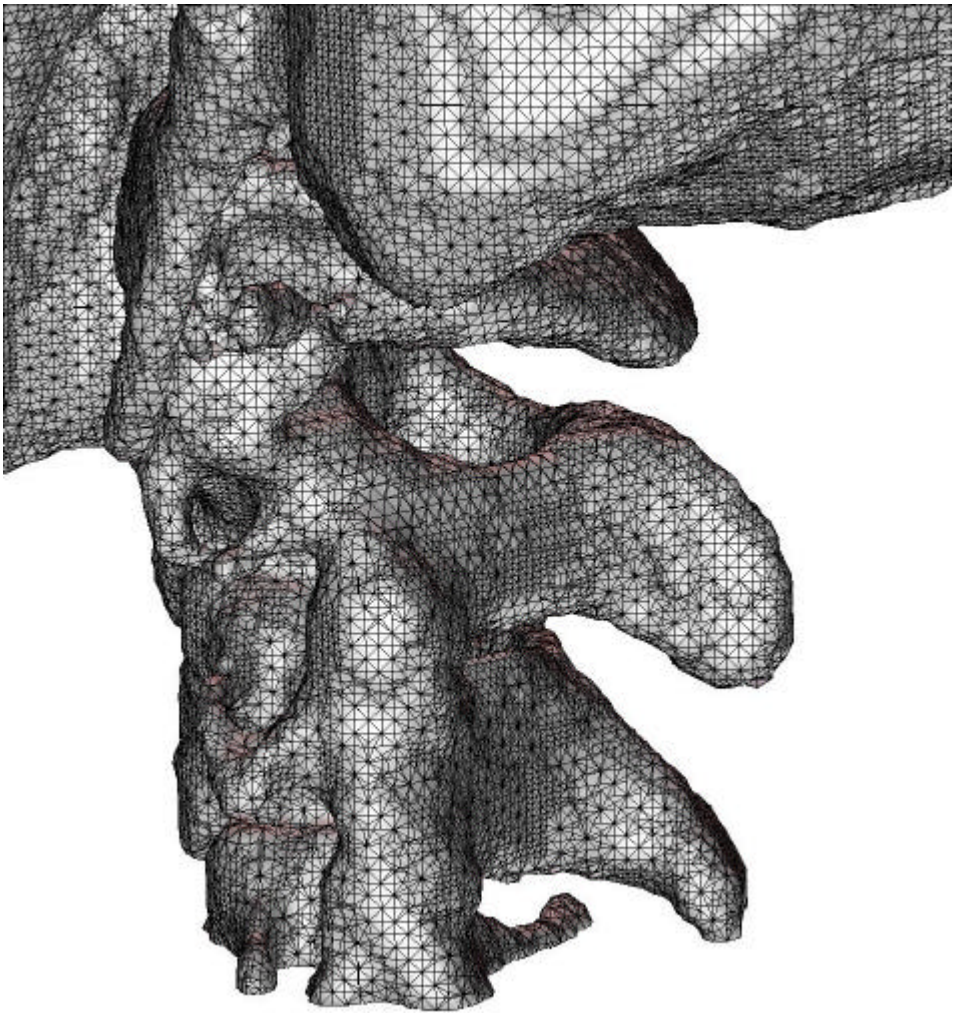
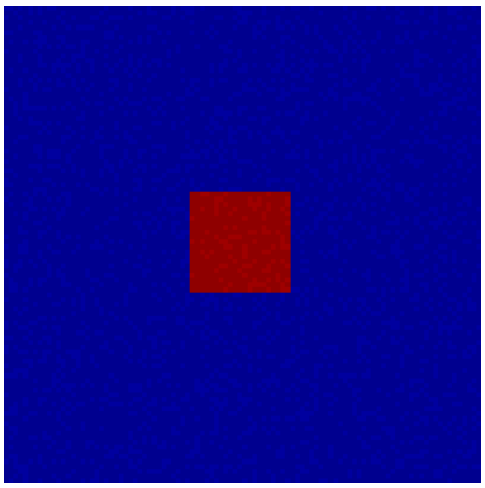


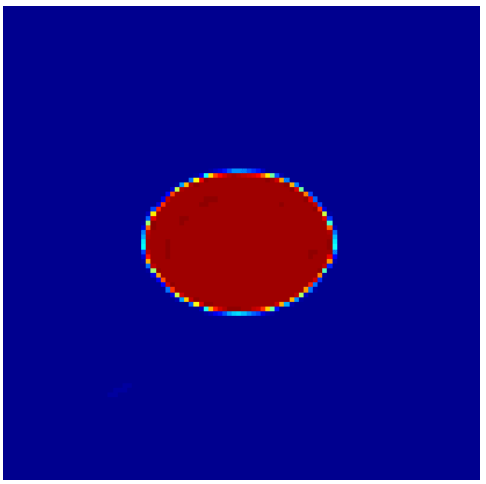
FIGURE 22: Zoom of mesh depicted in Figure 19.

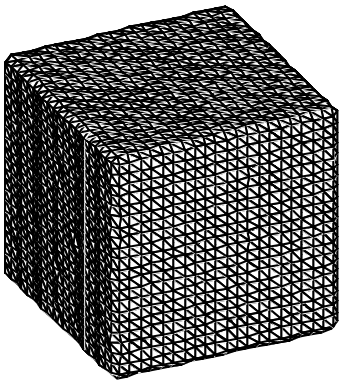
4.2 Mesh Template Tool

Original image

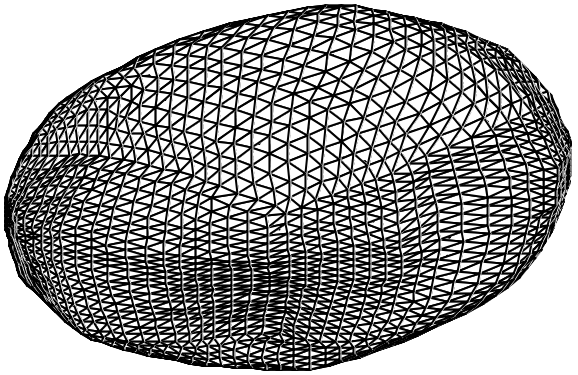


Transformed image

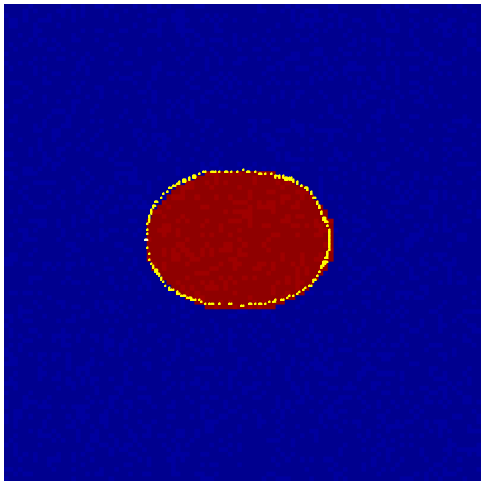




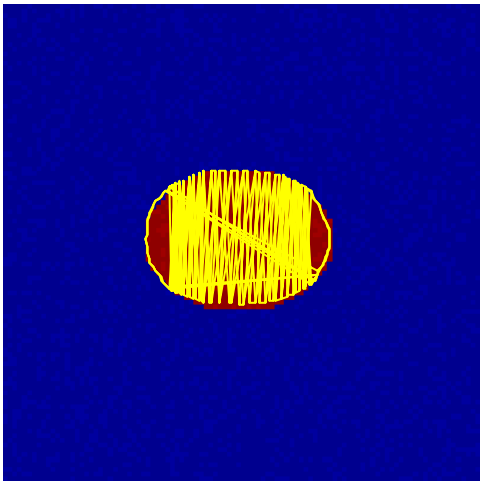
Original surface mesh



Transformed surface mesh



Target image with sliced surface segment as dots



Target image with sliced surface segment as joined line

FIGURE 23

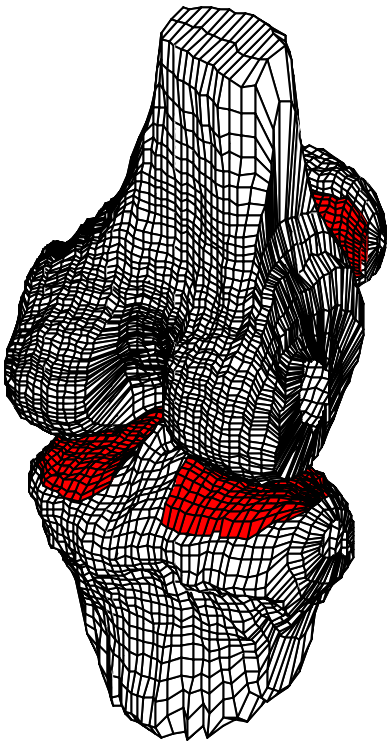
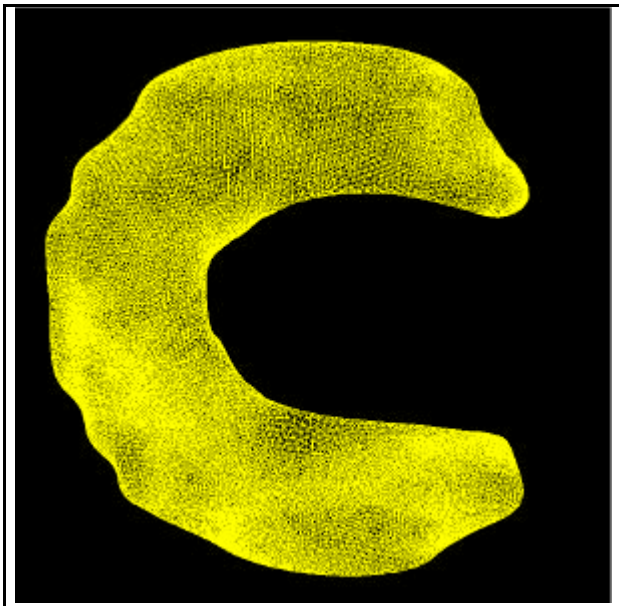
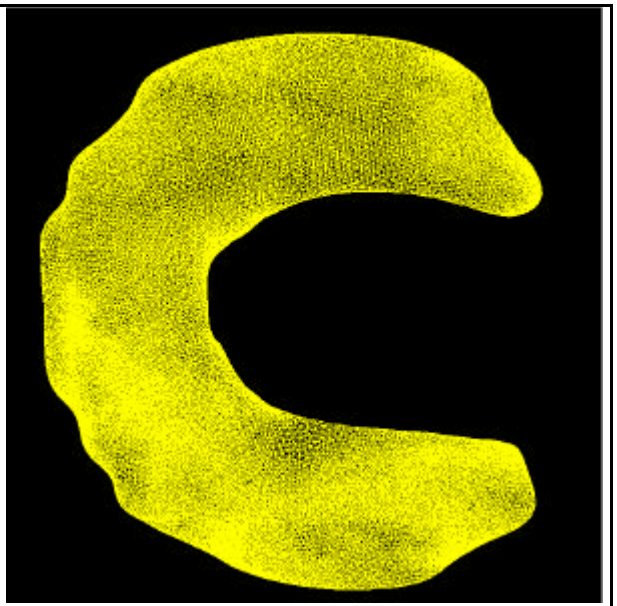


FIGURE 24: Smooth mesh of the human knee.

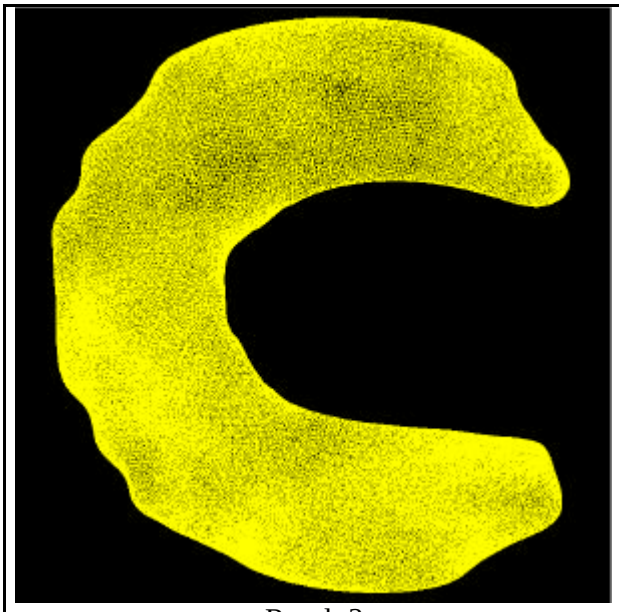
4.3 The ZMD Tool



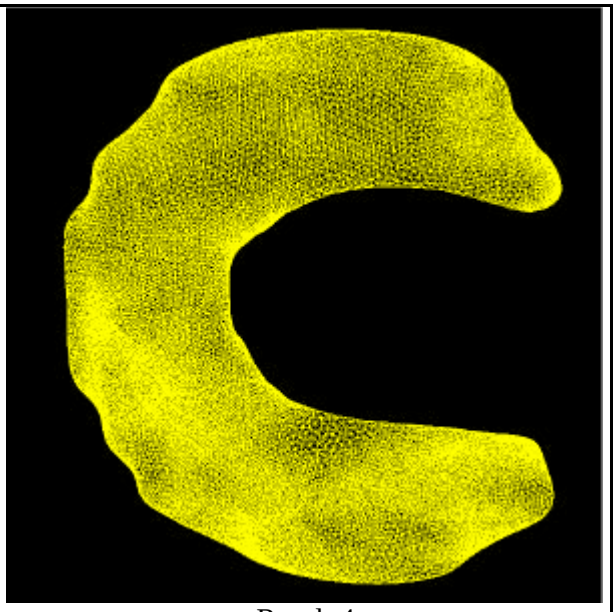
Result 1



Result 2

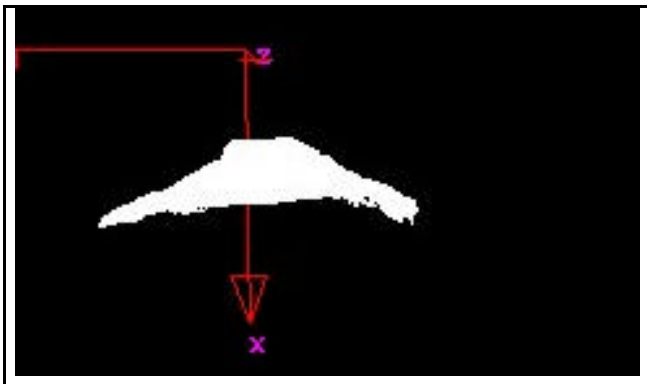


Result 3



Result 4

Figure 25 : Different meshes for different sets of parameters



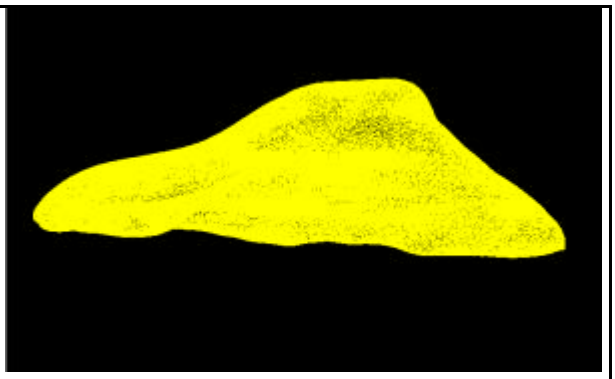
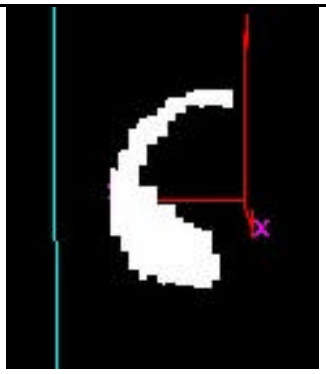


Figure 26: Lateral tibial articular cartilage



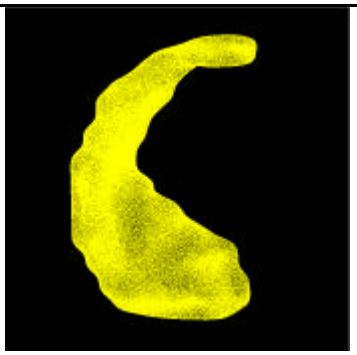


Figure 27 Medial meniscus