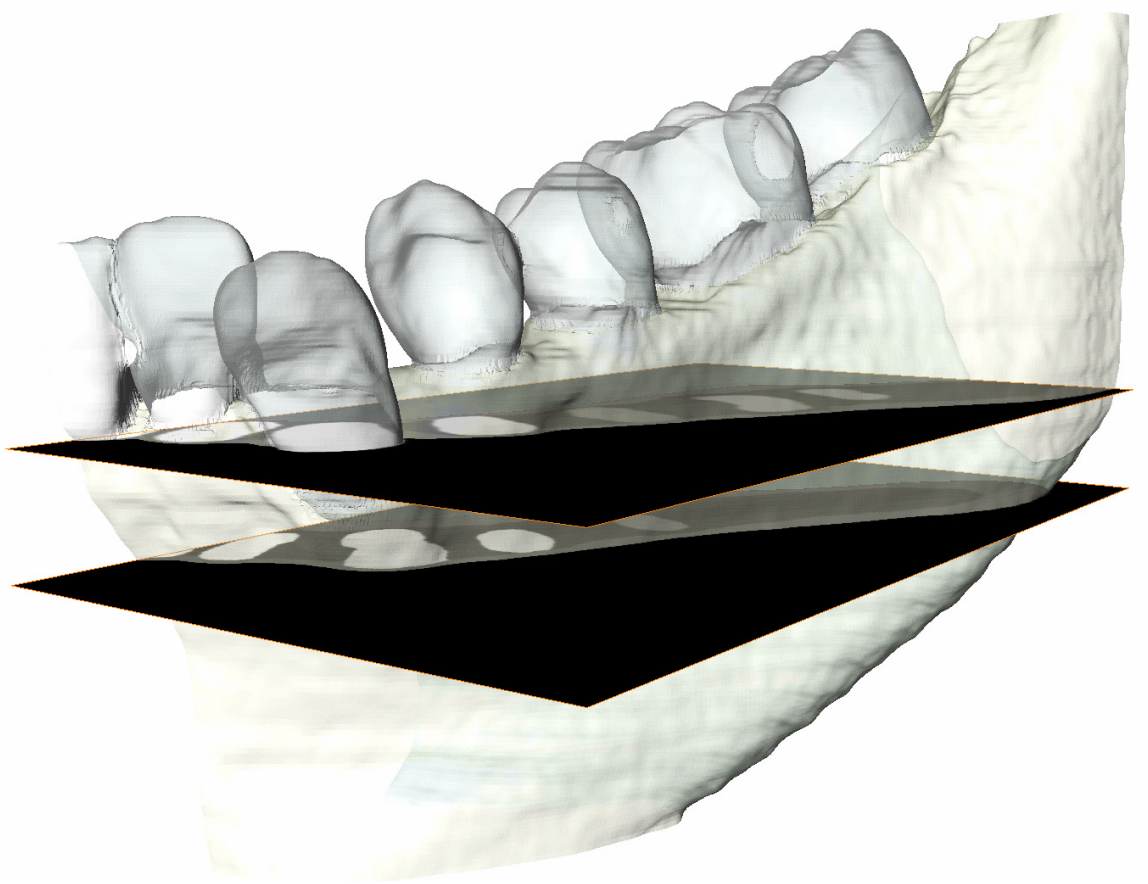
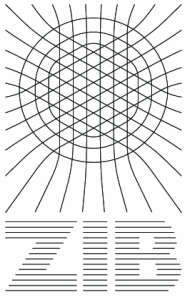
MASTER'S THESIS

Generation of smooth non-manifold surfaces from segmented image data



Jon Adolfsson and Johan Helgesson

23 May 2004



LUND INSTITUTE
OF TECHNOLOGY
Lund University

We would like to thank

Our supervisors

Dr. Detlev Stalling at the Zuse-institute
and
Lennart Ohlsson at the Department of Computer Science at the Lund Institute of Technology

Dr. Martin Seebaß at the Zuse-institute for all help and assistance with the thesis work here in Berlin

The Department of Orthodontics at the University of Ulm, Germany for the anonymized data sets we have used in our work

We would also like to thank the all the people who have helped us with the technical problems and the evaluation of our work

Abstract

This master's thesis treats the problem of correctly visualizing scientific data sets that contain many different materials. The studied data sets come from the medical applications where many different tissues within a patient have to be visualized simultaneously. These data sets are often acquired when using MR- or CT-scanner, devices more and more common as standard hospital equipment.

At ZIB in Berlin a visualization system called Amira has been developed to visualize these kinds of medical data. Amira visualize the tissues by generating three-dimensional surface meshes of triangles. Many different tissues can be visualized at the same time. Where three or more tissues meet some ambiguities arise of how to visualize the connections between the tissues correctly. In the current version of Amira all materials are treated in the same way which unfortunately leads to ridges and malformations on the surface when hard and smooth materials like bone are visualized.

The master's thesis deals with two different approaches to remove these ridges from the surface. The first approach modifies the way the surface construction algorithm in Amira works in order to directly generate one of the materials as smooth. The second approach deals with the output surface from the surface construction algorithm and removes the ridges by identifying their position and applying a low-pass filter.

The master's thesis was carried out at the Department for Scientific Visualization at the Zuse-Institute (ZIB) in Berlin, Germany. The smoothing techniques were implemented within the framework of the Amira software system, a modular and object-oriented 3d visualization system originally developed at ZIB and now also available in a commercial version (www.amiravis.com).

Keywords: *Visualization, Generalized Marching Cubes, Isosurface, Non-manifold surface, Surface smoothing*

1 Introduction

1.1 Background

1.1.1 Visualization systems

Visualization of scientific data is becoming more and more important. Not so long ago when computers weren't as powerful as today a simple image viewer or a plotting program might be sufficient for getting a good overview of the data. Computer simulations are today used in many areas of science such as engineering, physics, chemistry and medicine. These simulations produce very large data sets that often represent values in two or three dimensional space, sometimes also varying in time. Digital data acquisition technologies such as magnetic resonance imaging (MR), computed tomography (CT) and emission computed tomography (SPECT) also result in very large data sets that are similar in structure to those generated by computer simulations. To get an overview of such large data sets is hard, if not impossible. Without doubt these data sets are best understood visually as the human visual system is capable of processing large amounts of information and extract patterns and structures from it. Therefore visualization is very important in many areas. Today with the capacity of modern 3D graphics accelerators, 3D visualization can be performed on huge data sets making new insights possible wherever large data sets are produced. As our work will be mainly beneficial in the field of medicine, we have chosen to concentrate us with examples from that area.

1.1.2 3D reconstruction from medical images

There exist many applications where visualizing three dimensional data sets are required tasks. An important application today is the visualization of medical data. Results from computed tomography (CT), magnetic resonance (MR) and emission computed tomography (SPECT) are used as a base for acquiring stacks of pictures of the inner structure of patients. These images are later ordered and segmented, meaning that with the help of a segmentation tool the different materials in each image are recognized and their boundaries are determined and that the pictures are put in the order they were taken. From these segmented images, a three dimensional labelfield can be generated, this can be compared intuitively as ordering the pictures on top of each other in space with the distance separating them corresponding to the distance between the pictures at the time of acquisition. Now the data can be looked upon as a volumetric data set with all three dimensions. This can be thought of as a three dimensional grid, with all the grid points corresponding to one pixel in each picture. Using this three dimensional grid a lot of different visualizations techniques are possible such as volume rendering and surface rendering.

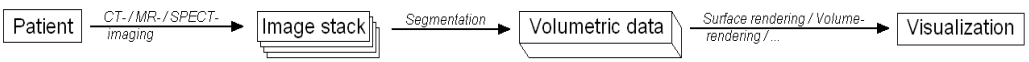


Figure 1: *The pipeline for 3D reconstruction of the inner structures of a patient*

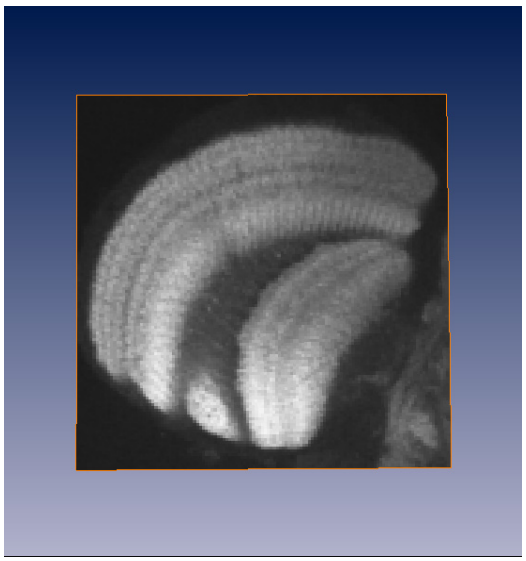


Figure 2a: *Image obtained from MR-scanner*

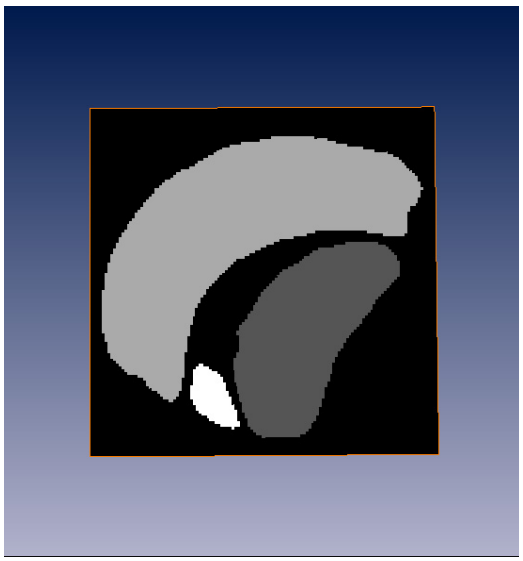


Figure 2b: *Segmented image*

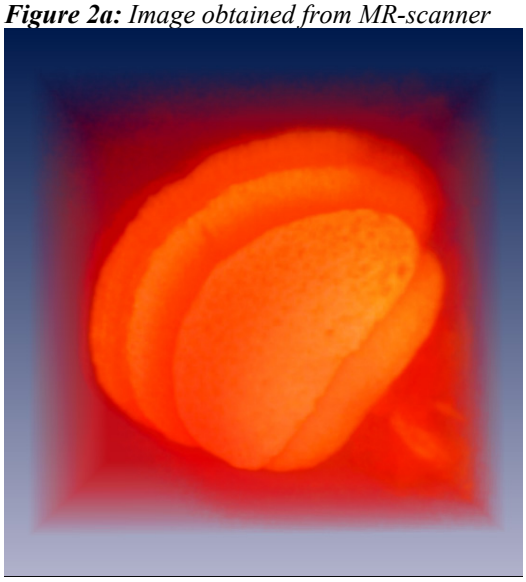


Figure 2c: *Volume rendering of original non-segmented image data.*

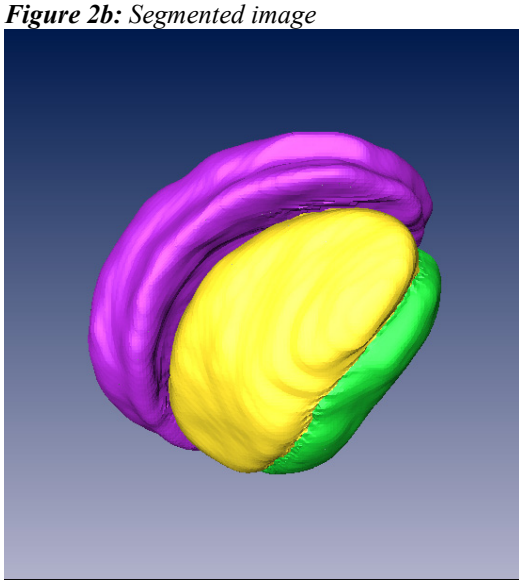


Figure 2d: *Surface rendering using segmented image data*

The visualization of medical data can give vital information to for example a surgeon when planning minimal invasive surgery or when fitting implants. The demand is therefore very high for high quality products in this area.

1.1.3 3D Reconstruction in Amira

In ZIB in Berlin a software package called Amira has been developed that can accurately visualize medical data with multiple materials or tissues. However in certain cases ridges can occur on the visualized surface, which is not desirable.

In this section we will present a brief description of the surface reconstruction algorithm which leads us to the formulation of the problem treaded by this thesis. The reader might find it valuable to consult the terminology section in Appendix A to get a deeper understanding of the description.

In the first step in the reconstruction pipeline is the medical data gathered from the patient. The medical data is normally a stack of two-dimensional images acquired by a CT-, MR-, SPECT-scanner or other acquisition devise. The images are taken with millimeter distance, allowing good detail for the reconstructed three-dimensional surface.

The second step in the reconstruction pipeline is to identify the different materials or tissues in the data. This is done during a segmentation step that labels the different pixels in the image slides as fat, bone or muscle and so on. This step is normally done semi-automatic where the user of Amira uses different tools to find the contours around the different regions of materials in the image slides.

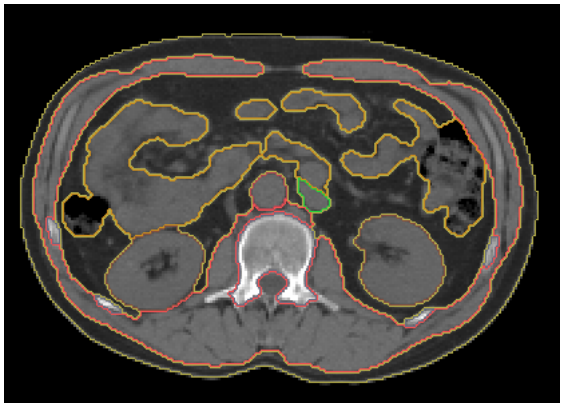


Figure 3: *One slice of a segmented 3d medical image data set. The contours separate the different regions*

The labeled image stack is then smoothed in order to obtain weights for how probable the labels of the pixels are. This improves the quality and the accuracy of the resulting surface model. The stack of labeled images forms a three-dimensional labelfield.

The last step in the reconstruction pipeline is to generate an isosurface that separates the different regions in the labelfield. This is done by dividing the labelfield into a uniform grid and calculating surface faces for every grid cell in the grid. The size of a grid cell is dependant on the resolution of the acquired images. This is called the triangulation of the grid cells. The grid cell has eight different corners that can be labeled with different materials. If all corners lie in the same region of the labelfield no isosurface passes through that grid cell, but if one or more corners lie in different regions an isosurface intersects the grid

cell, and it needs to be triangulated. The isosurface is complete when all grid cells have been triangulated.

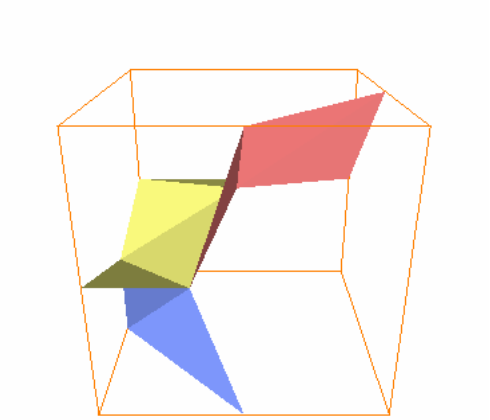


Figure 4: *Example of a triangulated grid cell with three different labels at the corners*

1.2 The problem

The surface generation algorithm in Amira can handle up till eight different vertex classes at the corners of the grid cell, thus allowing up till eight different materials meet in the same grid cell. During the triangulation the weights of the corners of the grid cell are used for determining where the isosurface should separate the different materials that meet in the grid cell. When just two materials meet in the same grid cell, the problem of reconstructing the surface is rather trivial, but problems may arise when more than two materials meet. The algorithm doesn't separate between materials that are known to be hard and smooth like for example bone or tooth and other soft materials like muscle or fat. This leads to visualizations that look incorrect, for example ridges occur where the bone meets two or more other materials. The problem is illustrated below.

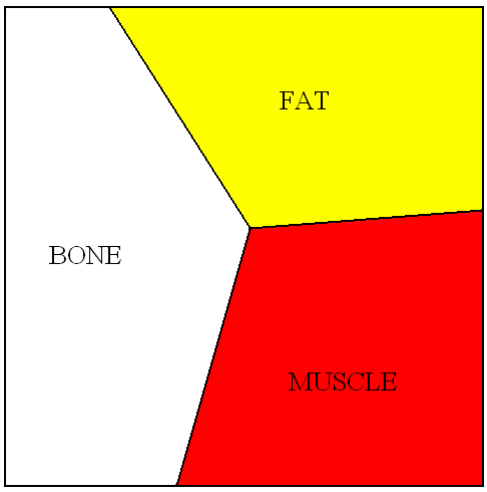


Figure 5a: *Cell face Configuration computed by the surface reconstruction algorithm leading to a ridge on the bone surface*

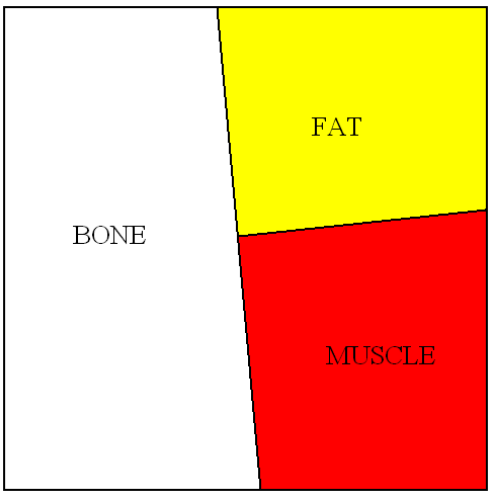


Figure 5b: *Cell face configuration not leading to a ridge on the bone surface*

1.3 An example of the ridge problem and consequences

The original request for improvements where made by Prof. Cornelia Kober, working at the Faculty of Engineering and Computer Science at the University of Applied Science, Osnabrück. In her studies of a biting simulation [8], teeth are pressed against some obstacle and bored into the bone. In this context, the unevenness of the teeth caused by the ridges causes incorrect stress and strain peaks on the bone. Removing the ridges would therefore be a benefit, and make the simulations more accurate. Cornelia has until now tried remove the ridges by surface simplification and overall smoothing, a strategy that will lead to loss of detail.

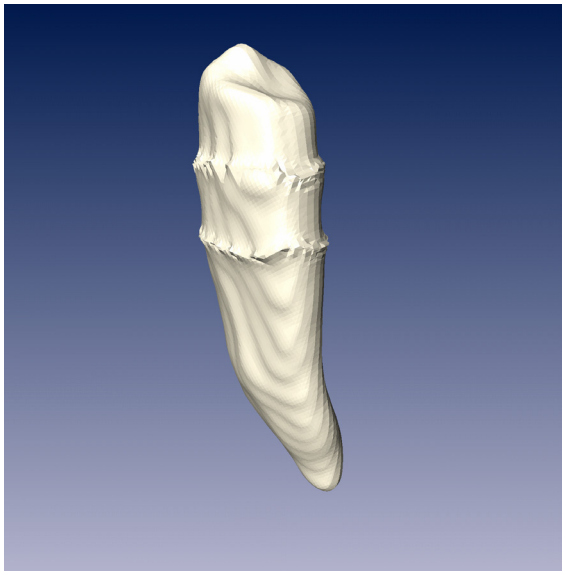


Figure 6: *Example of a tooth from the biting simulation. Note the ridges where the tooth meets other materials.*

Even when taking these measures it is not possible to totally remove the strain from the teeth caused on the bone, as shown in the figure 7b. If the ridges had not existed at the time of constructing the model it would be possible to respect more details of the dental anatomy.

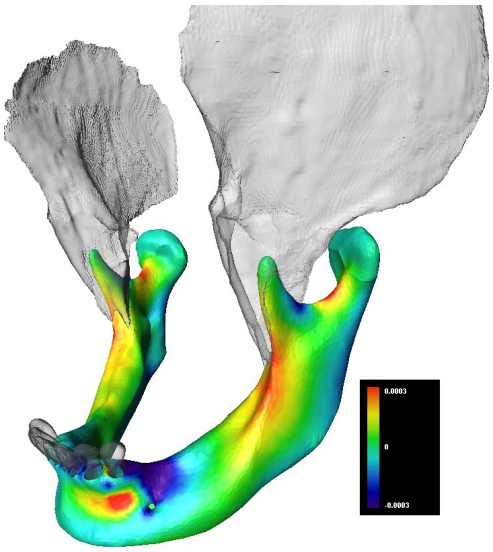


Figure 7a: *Figure showing the strains in the jaw bone during the biting simulation. Blue indicate compression and red indicate tension.*

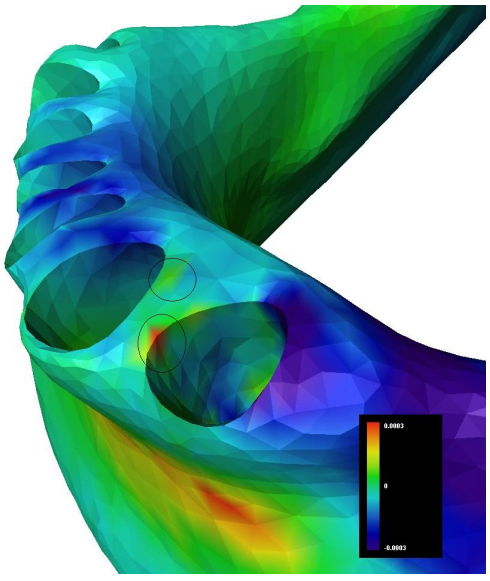


Figure 7b: *A close up on the jaw where the teeth have been removed, the indicated tension on the jar originates from the ridge on the tooth.*

1.4 Goal description

At the beginning of the project two different approaches for removing the ridges was suggested by the staff at ZIB; altering the GMC itself or to make some post-processing of the generated surfaces. The advantage of post-processing would be that information about the whole surface can be considered rather than just the small part inside a grid cell. Post processing is also easier to guide, one can choose exactly what to remove or not. As the surface is already generated, one can see the magnitude of the distortions and where they occur before taking action.

The second approach of altering the GMC algorithm so that it avoids creating ridges for a material selected to be solid and smooth is also a very tempting idea. The advantage of this approach is obvious as the surface will be generated correctly from the beginning, considering all the original information from the normal surface generation, such as weights and labelfields. Avoiding the construction of the ridges in the GMC also makes it easier to automate the process, as minimal user interaction is needed. Both approaches will be considered and compared in this master's thesis.

1.5 Contribution

Our contribution to the Amira visualization system was based on these two suggested approaches. We concentrated in doing research in the field of these approaches, trying to find the best solution possible for the time at hand. The implementation and research were made in a parallel manner. As the knowledge of system grew, more sophisticated implementations were possible. During our work some older versions were discarded and some were kept in order to extend functionality.

1.5.1 Changes to the GMC algorithm

The GMC algorithm works correctly for the cases where just two materials are present. Therefore it has been altered in order to simulate the behavior of reconstruction of just two materials. The one selected to be smooth (the special material) and the exterior. While ridges appear when three or more materials meet, none will occur using this approach. In order to do so the following changes have been made:

- The smoothing process has been altered in order to preserve the labeling and the weights for the special material just as if only this material and the outer were present
- The shifting policy when shifting the point on the edges in the grid cells has been changed to take in consideration the original weights of the special material.
- The shifting of point on the faces of the grid cells has been altered in such a way that the special material always has a straight boundary

towards the others. It has also been taken in consideration the way that the boundary between the two other materials is created.
- The shifting of the points inside the grid cell has been altered to just consider the resulting points from a triangulation similar to the one when only the special material and the exterior are present in the labelfield.

1.5.2 The smoothing module

We have created an Amira module for post process removal of the ridges from the surfaces generated by the original GMC algorithm. It lets the user select a material to be smooth and then smooth the surface in the regions where the ridges occur. Different by us invented smoothing algorithms as well as already existing ones have been implemented. The user can define a number of parameters in order to make the smoothing of the surface better and easier. There exist parameters for selecting:

- *Surrounding depth*, defining how far out from the surface boundaries the smoothing is about to take place.
- *Filter parameters* to allow the user to change the properties of the used filter.
- *Marking of boundaries* allowing the user to see how the boundary has been moved.

There are three algorithms in the module. One of the algorithms simulates a low-pass filter, removing the ridges and the general irregularities in this manner. Another modified version of this algorithm has been implemented for straightening out the boundary, making it less edgy. A third algorithm have been invented that gives virtually the same result as the first one for the special material, but without distorting the surrounding materials as much as it keeps the original form of the boundary.

These two approaches have been successfully implemented and proven to be useful in different situations in the use of the Amira visualization system.

1.6 Introduction to Amira

Amira, initially developed by the scientific visualization group at the Zuse Institute Berlin (ZIB), is a commercial product for visualization and analysis of volumetric data in many fields, including medicine, biology, physics and many more. Amira also contains several add-ons, amiraMol for molecular visualization and biochemical data analysis, amiraDeconv for deconvolution of microscopic images and amiraVR, which provides virtual reality interaction. Amira also contains support for other techniques as for example stereo viewing. In addition to the mentioned add-ons amiraDev is also available, making it possible to develop and extend the capabilities of Amira. A demo version of Amira is available at www.amiravis.com.

1.6.1 Overview

Amira is constructed in a modular and object-oriented way with computing modules and data objects being the basic components. When Amira is started three windows are invoked, the *main window* containing the object pool and the control area, the *console window* where user interaction is possible in means of typed commands, scripts and text output and a *viewer window* where the visualization results will appear. To import data in Amira the file browser or drag and drop can be used. All data sets and modules are represented as icons in the object pool. By clicking on them, additional information appears, and by right-clicking a pop-up menu appears, showing a list of modules that can be connected to the particular object selected. Connecting *compute modules* to data sets makes it possible to perform operations on this data that can be all from filtering, aligning of images, to generating surfaces from labelfields. The compute modules are also represented by icons in the object pool and the relations between the compute modules and the data sets are shown by lines connecting them, thus visualizing which modules get their input and output data from and to which data objects. Parameters of data objects and modules are displayed and can be modified in the *control area*. All these parameters can also be accessed by the *Tcl command interface* in the console window, allowing the user to query and modify them. Objects can not only be modified by the user in editors, but also by other components, mainly compute modules. The visual results from the main and the console window will be shown in the viewer window taking advantage of the latest release of the TGS Open Inventor graphics toolkit.

An example

The picture below shows a screen shot of Amira visualizing a surface reconstruction of a data set called lobus, a part of a flee brain.

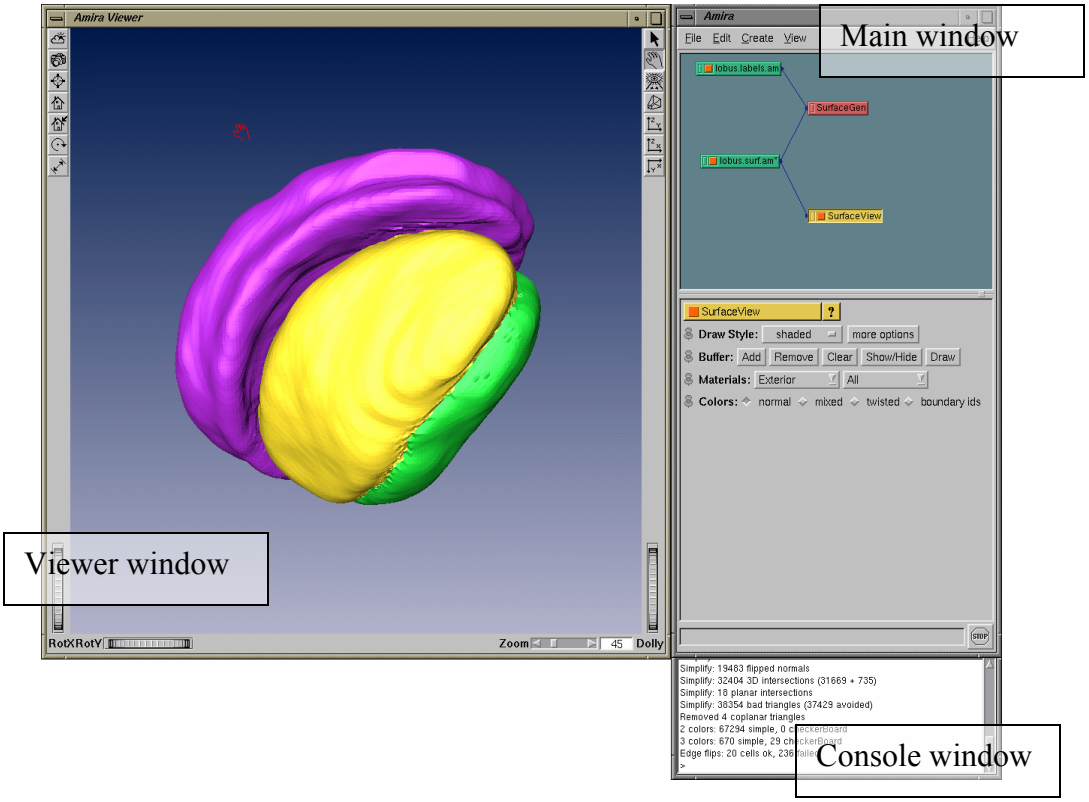


Figure 8: *Screenshot of the Amira viewer*

The visualization of the surfaces generated from the lobus labelfield shown in the viewer window was obtained in the following way;

- A label field data set was loaded (lobus.labels.am),
- Then a surface generator module (SurfaceGen) was connected to the data
- The "doit" button was pressed, generating the iso-surface data (lobus.surface.am)
- This data was finally connected with a surface viewer module (SurfaceView) which output is shown in the viewer window.

The four objects can be seen in the object pool with lines showing the dependencies. While the SurfaceView module is marked, its properties are being showed in the control area and can be altered. The console window contains some output that the modules generated during the process.

1.6.2 Features

Amira comes with various tools for treating and visualizing scientific data. A short description of them follows.

Direct volume rendering – meaning that light emission and light absorption parameters are assigned to each data point of the volume, then light transmission is simulated through the volume, thus making all the data visible. This can be done in real-time even with large data volumes with the help of modern graphics hardware. Multiple data sets can be rendered simultaneously and the result can also be combined with a polygonal model to give better understanding of the data.

Isosurfaces – are mostly used to visualize scalar grid fields. Applied to 3D images the method provides a quick and at most times sufficient reconstructed model using a polygonal mesh. Amira comes with an improved algorithm that generates isosurfaces with few triangles, making it possible to visualize large 3D data sets even on small desktop computers.

Segmentation – the component for segmentation comes with many features such as a brush tool for painting, lasso tool for contouring, magic wand for region growing, thresholding, intelligent scissors, contour fitting and much more. All for making the classification easy.

Surface reconstruction – for visualizing the data when segmented, Amira provides a fail-safe algorithm that creates topologically correct polygonal surface models using fractional weights and label fields.

Surface simplification – In order to make models processable on low-end machines and to reduce the number of triangles, Amira comes with a surface simplification tool.

Generation of tetrahedral grids – using a flexible advancing-front algorithm, tetrahedral grids can be generated, suitable for advanced 3D finite-element simulations.

1.6.3 Extending and altering Amira

1.6.3.1 Software structure

Amira is built to be easy to extend, this is ensured by the modular software architecture where related modules are organized into packages. Each package has the form of a shared library, which is loaded at run-time. This way only the needed code is loaded and this keeps the size of the executable small in the memory while keeping the flexibility to add or replace functionality. Extending Amira is quite easy as it is written in C++ and requires only a limited number of libraries. The 3D graphics is implemented in TGS Open Inventor which gives support for multiple platforms and many other advanced features. The user interface is built in Qt, giving multi-platform support, thus making it easier to port the code.

1.6.3.2 Modules and data

Modules are the core of Amira; they contain the actual algorithms for the visualization and the data processing. The whole process from treating input data to the final visualization of the result is handled by modules, linked to the intermediate data. There exist two types of modules, *compute modules* and *display modules*. Compute modules usually takes some data as input, modifies it, and produces an output while display modules normally just visualize their input data. Both module types are derived from the common C++ base class HxModule. When imported, both data and modules can be seen in the object pool. Data objects are persistent in memory and visually represented in the user interface. Data is accessed by the modules using the C++ interface of the data classes. Modules are loaded into a common process space at runtime, by using shared libraries. This way they can communicate like C++ objects in a normal C++ program.

1.6.3.3 About writing modules

By extending the C++ base class HxCompModule and including a few libraries a compute module can be created and integrated in Amira. Accessing input and output data is done via an interface, and there exists an API providing various functions to the programmer. The API is vast and since Amira itself is a huge project it can take quite some time to get going with the module writing, but when a basic knowledge of the system has been obtained the development goes smoothly. The same goes for creating visualization modules, but the base class HxModule can be used instead since a visualization module normally don't produce a data set. After the modules have been completed, they must be added to the Amira system. This is done by altering a configuration script that contains information about all modules included in the system.

1.7 Terminology

Medical imaging contains a wide range of technical terms and expressions that are specific for this field. Therefore for readers with little or no experience it might be valuable to consult the Appendix A before and during the reading of the presented material.

2 Altering the GMC

In order to generate smooth surfaces without ridges two approaches are obvious. To modify the algorithm so that the creation of such ridges is avoided or to try and remove these ridges after the surfaces have been created. This part will treat the first approach, to avoid creating the ridges at all.

As seen in the introduction of this report, when 3 or more materials meet in a cell the surfaces created by the GMC can contain ridges. The nature of the problem is such that to achieve satisfying results, only one of the three or more materials can be considered hard and smooth. Therefore, the goal of this approach was to modify the module containing the GMC to allow the user to select one material that should be considered to be hard and smooth and construct the surface of this material in such a way that this is achieved. The advantages of this approach are obvious. Correcting the problem as it occurs leads to less loss of data and precision, since a ridge removal after the construction of the surface may distort the surface in such a way that it does not longer correspond with the actual data in an appropriate way. Working directly with the GMC gives many advantages, direct access to the labelfield data, control of the smoothing of the labelfield data, control of the creation of the weights and much more. This makes it easier to create the surface in such a way that the deviation from the labelfield data is minimized.

Since the generation of a surface from data containing more than two materials could give ridges, the idea is to reconstruct the special material as if only this material and the material that defines the outer region existed. Amira allows a user to add and remove materials from a labelfield data set. Thus, removing all the other materials but the special material from the data set and then running the GMC algorithm on the data generate a ideal result which is the result to strive for. The drawbacks of this approach are that if a material is treated in a way as if it was the only material present in the labelfield, the other surfaces will be a bit distorted.

2.1 Theory

2.1.1 Marching cubes

In order to understand the Generalized Marching Cubes algorithm, the GMC, some understanding of the Marching Cubes algorithm must be obtained, which is the father of the GMC.

Marching cubes, as proposed by Lorensen and Cline in 1987 [4] is an algorithm that creates polygonal models from 3D data. The algorithm processes the 3D data in scan-line order and calculates triangle vertices using linear interpolation creating a surface that approximates the input data. This is done using *binary classification*, meaning that the data is supposed to contain only two materials, the inner and the outer material.

The original marching cubes algorithm consists of two primary steps, first, the construction of the surface according to a specified value c and then, to ensure image quality when the model is rendered the calculation of surface normals to each vertex of each triangle. The essence of the algorithm is contained in the first step, therefore the second step is intentionally left out in this short introduction.

In order to create the surface, the user has to choose a threshold value *c*. This value determines what shall and shall not be considered to be the outer material and the inner material in the slices. For example in the figure to the right, if the value of the threshold c is set to correspond to the grey scale value between a and b, then pixels with the grey scale of b or lower value (brighter) would be considered as not belonging to the inner material.

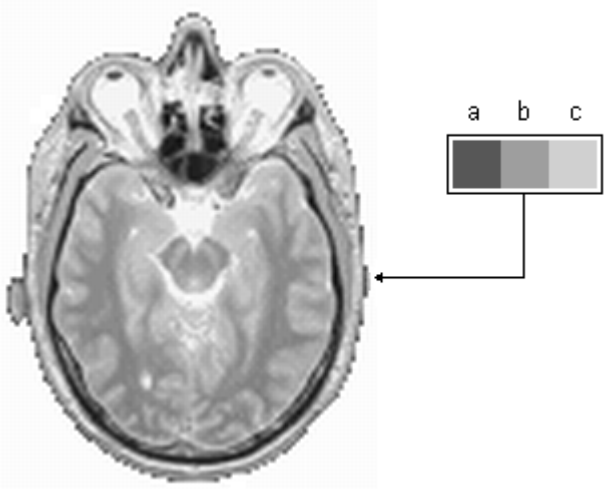


***Figure 9**: The figure shows a picture slice of a head, with amplification of a region on the right.*

Setting c to this value means that the surface is located between a and b as the surface is located at the intersection of the outer and inner material. The marching cubes algorithm attempts to locate the surface in a so called *grid cell* created from eight pixels in two adjacent slices, having one pixel corresponding to each vertex in the corners of the cube. The cube in the figure below is drawn in cell space, where (i, j, k) corresponds to the pixel of the coordinate (i, j) in the k^{th} slice.

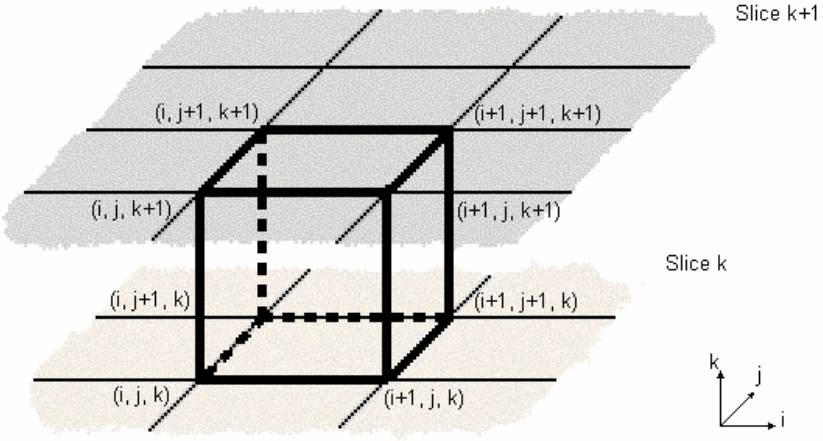


***Figure 10:** A grid cell created from 8 pixels in two adjacent slices. The pixels are located at the corners of the cell.*

The algorithm then test how the surface intersect the cube and continues (marches) to the next grid cell.

To find the surface intersection a 1 is assigned to the cubes vertex if the value at the vertex (normally the grey scale value obtained from the corresponding pixel in the slice) equals or exceeds the threshold value c. If a 1 is assigned, this vertex is considered to be on or inside the surface, the other vertices that are located outside the surface are assigned a 0. Then, all cubes that contain at least one vertex inside or on the surface (assigned to 1) and at least one vertex outside the surface (assigned to 0) are intersected by the surface.

The number of possible configurations of the vertices, inside or outside, is 2^8 (256), but considering the fact that the topology remains the same if the configuration is mirrored reduces the number of cases to 128. If rotational symmetry also is considered the number of cases are diminished to a manageable 14 cases, plus the case when no triangulation is necessary, when all the cubes vertices are above or below the threshold (case 0). Putting these cases in a look up table instead of triangulating each cube individually speeds up performance considerably. To the right the different cases that can occur are shown.

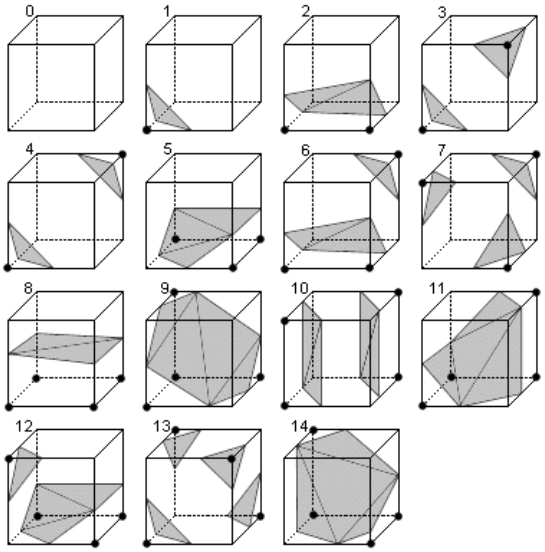


Figure 11: *The possible topological cases in marching cubes.*

When each cube has been assigned a triangulation, the surface intersection can be interpolated along the edges of the cube and the surface will be constructed of the triangulations made in all the cubes. For example, consider that the surface constant is chosen to c=86, and two of the cubes vertices corresponds to pixels giving them the value of 68 and 104. Then using linear interpolation the intersection of the edge would be exactly in the middle of the two vertices since the distance to the intersection point from the vertex with the pixel value 68 is $(86 - 68)/(104 - 68) = 0.5$

In general, the marching cubes algorithm works as follows:

- Scan two slices and create a grid cell from four pixels in one slice and four pixels in the next slice.

- Determine the index of the cube in the lookup table comparing the threshold constant c with the values assigned at each vertex of the cube.
- Using the calculated index, look up the triangulation of the cube
- Determine the intersection point p (where p=c) for all vertices that lies on the cubes edges interpolating the values assigned at each vertex.
- Continue with the following pixels in the slices or when ready, change slices

2.1.2 The GMC-algorithm

As previously seen in the section describing the marching cubes algorithm, surfaces can be generated from binary classifications, that is, data where just two different classifications are possible, interior and exterior. However, the creation of surfaces from non-binary classifications is an important task in many areas, for example in medical imaging, it's important to be able to create geometric representations from MR images. In such images normally more than 2 different data classes exists. With the normal marching cubes approach just one tissue could be visualized, leaving out the others. The advantage of being able to see the surrounding tissues as well is obvious. An algorithm that is able to handle such cases is needed.

The traditional approach in medical imaging to create these geometric representations has been to connect the contours of the different classes in neighboring slices. This method often leads to ambiguities and requires user interaction to produce a satisfying result. Methods to create such geometric representations already exist, for example Bloomenthal and Ferguson [10] have developed a method to generate surfaces by decomposing a grid cell in tetrahedras. A marching tetrahedra algorithm is also proposed by Nielson and Franke [11]. The problem of these algorithms is that, even though they solve the problem of generating surfaces from non-binary data, they generate an excessive amount of triangles whereof many are badly shaped.

The GMC algorithm (Generalized Marching Cubes), developed at Konrad-Zuse-Zentrum in Berlin [3] is an extension to the previously explained marching cubes algorithm. The motivation for the development of the algorithm was to create a fast and robust solution that would be able to create surface representations from data containing multiple vertex classes. While the GMC allows an arbitrary number of vertex classes to be defined this leads to that new concepts has to be introduced. The generated surfaces does not longer belong to a single solid material as in the marching cubes, but is constructed of surface patches, separating volumes of two different classes.

As in marching cubes, the grid cells or grid cells, are traversed and classified according to their configuration of vertex classes. The triangulation is computed interpolating weights that are set for each of the vertices, which may already be given with the data set, or be assigned by a special smooth-method. The method to find the triangulation that approximates the boundary surfaces is completely

automatic in comparison with other existing approaches that may require user interaction to resolve ambiguities. This, with the help of a look-up table that gives the algorithm the speed comparable with of marching cubes, assures fast generation and a topological correct solution to the surface generation problem.

The resulting surfaces generated by the GMC are also referred to as so-called *non-manifold surfaces*, meaning that one edge may be part of three or more triangles.

2.1.2.1 The Algorithm

In order to generate an iso-surface using the marching cubes algorithm the data has to be constructed in such a way that only two different data classes exits. With this assumption 2^8 different configurations can occur in a cubic grid cell, allowing the use of a look-up table for each triangulation case of the cells. In this way a very fast algorithm is obtained, but it yields a lot of problems when working with data sets that contains more than one data class. Instead, the GMC works in a way that it generates iso-surfaces that separate the different data classes of the volumetric data, allowing multiple data classes in the same data set. Thus, surfaces generated by the GMC may contain contours and points where three or more regions belonging to different classes join, producing non-manifold surfaces. In most cases the cells triangulated by the algorithm contain no more than 2 or 3 different vertex classes. Using look-up tables for these configurations, which after elimination of the geometric equivalents reduces the number from 6561 (3^8) to 58 different triangulations, a performance similar to the one in marching cubes is achieved. Still, this leaves out the more general cases that have to be triangulated by a special triangulation method. Since they are not very frequent in the normal application the speed isn't affected too much in the normal case.

2.1.2.1.1 A schematic overview of the GMC algorithm

The schematics of the GMC algorithm do not differ very much from the marching cubes algorithm. The biggest difference lies in that the data is represented as a labelfield lattice that could be thought of as a uniform grid, where every grid cell has a label in each corner indicating which material it belongs to. Every label also contains a set of weights, one for each of the existing materials.

The algorithm could roughly be described as follows:

- Examine the labelfield to find out which materials exists
- Smooth the labels in the labelfield to obtain weights
- Scan two slices at a time and create a grid cell from four points in one slice and four points in the next slice of the labelfield
- If the cube contains at most 3 different labels, look in the look-up table for the triangulation. If the cube contains more labels, create the triangulation with the triangulation algorithm.

- Shift the points of the triangulation according to the weights
- Continue with the following slices

2.1.2.1.2 Labeling, the separation of the regions

In medical imaging a labeling is normally obtained using a segmentation algorithm on a stack of MR images, while in other applications it may be obtained by evaluating an analytical expression. Other techniques also exist to mark the regions, such as contouring.

In order to reach the goal of creating non-manifold surfaces that separate regions of different type, space must be represented in some way. To make the explanation of the algorithm easier, let's assume that space is divided in a uniform discrete grid, with every grid cell corresponding to a label.

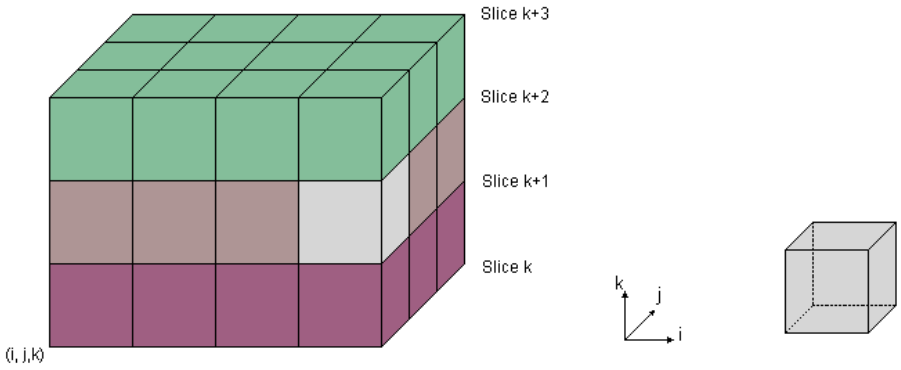


Figure 12: *A uniform discrete grid, made up of 4 image slices that are segmented into regions, labeled and later put together. The uniform grid is a simplification due to the fact that the distance between the image slices and the distance between the pixels are not always the same in the real case, for example when image data are obtained from MR scanners.*

Let σ denote the total number of region types that exists, yielding σ different labels. We can now define a cubic grid cell with eight vertices from the discrete grid. Let there exist a set of σ weights $\{p_0..p_\sigma\}$ at every vertex, where p_0 denotes the weight that the vertex belongs to the label 0 and so on. If a vertex is assigned to the material k, then every weight variable p_i is set to 0 except for p_k, which is set to 1. To classify an inner point of the cell the region of maximal weight is chosen, the function

$$\text{Region}(x) = i \mid p_i(x) \ \ maximal$$

can be evaluated using for example tri-linear interpolation, where $p_i(x)$ denotes the interpolated weight in the point x. Classification using the maximal weight does not require the weights to be set to 0 or 1, but restricting to integer values in this way makes it easy to distinguish between the different topological configurations. That way using a simple indexing scheme a look-up table can be

used, increasing speed remarkably. To obtain smoother results however a non-binary classification is preferred.

2.1.2.1.3 Obtaining weights, the smoothing of the labelfield

In order to make the triangulation as correct as possible and to obtain smooth surfaces as result, weights are used together with the labels. Normally a label lattice does not contain any weights, but weights could have been assigned when creating the labelfield. A simple and very effective way to obtain weights that will generate a good result is to use a smoothing filter on the labels.

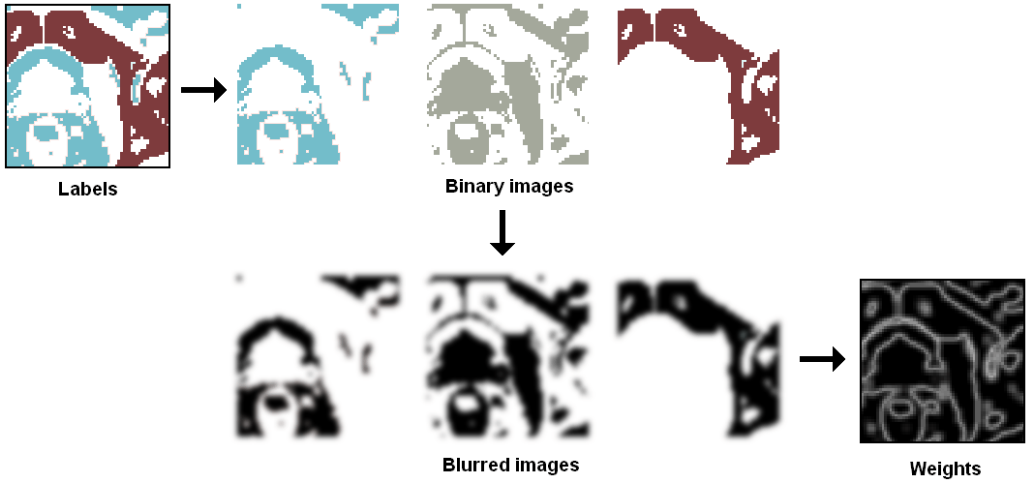


Figure 13: *Starting from a labelfield, binary images are extracted and blurred. These are later used to obtain the weights.*

First, to illustrate the method let us begin in the 2D case. This could be compared to treating just one slice at a time from the stack of slices that make out the labelfield. Let us hold on to the definition of σ as the number of different materials in a labelfield. Then in the first step, the 2D labelfield is separated into σ intermediate images, where every pixel is set to the maximum intensity where the current label occurs and zero otherwise. Then all the images are blurred using a Gaussian filter, resulting in σ images containing pixels somewhere between the maximum and minimum intensity. Now, at every position (i, j) in the grid, the weights of the different materials are set to the corresponding intensities found in the σ smoothed images.

In some cases, this smoothing can result in that a position (i, j) in the labelfield contains a label *a*, but the weight for the material a is less than the weight of some other material b at this position. In order to tackle this problem, two different strategies are available, *constrained-* and *unconstrained smoothing*. In the case of unconstrained smoothing, if such situations arise, the position (i, j) will be re-labeled to contain the label of the material that has the biggest weight. If instead constrained smoothing is used, the original weights are always preserved resulting in that no such re-labeling will occur.

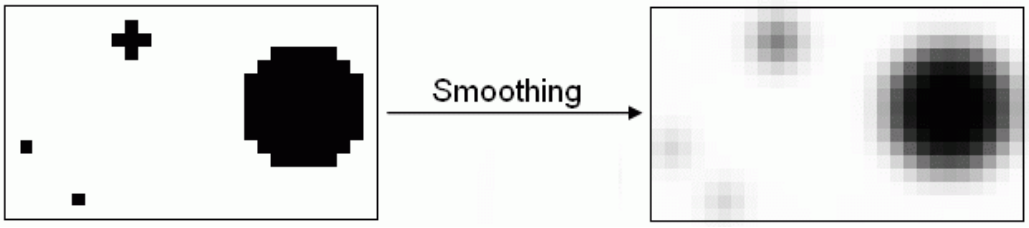


Figure 14: *The picture on the left represent a 2D labelfield with two labels, black and white. The right picture is the result after the smoothing using a gauss-kernel of width 1. If unconstrained smoothing is set, the two small dots would have too low weights in comparison to the surrounding material and would be re-labeled to belong to the white region. If constrained smoothing is set, these points would still be labeled to belong to the black region. As seen in the picture, a smoother, less "noisy" result is obtained with unconstrained smoothing on the cost of loss of detail*

The advantages of the unconstrained approach are that isolated labels will be removed producing a smoother, less noisy result with smoother boundaries. But this may also lead to loss of data or precision, speaking for the constrained approach that preserves the original labelfield.

The smoothing strategy described is pretty easy to generalize to the 3D case. Nevertheless, some precautions have to be taken, while separating the labelfield into σ labelfields and smoothing each field separately increases the memory usage, which makes large models impossible to handle in computers. The approach used to reduce the needed memory is to process the labelfield slice-wise. Combining bounding boxes that contain the weights of each material and the use of only $3*\sigma$ buffers, lots of space can be saved. This while memory just has to be allocated to hold the size of the bounding boxes of each material and not always the size of a whole slice.

A schematic overview of the smoothing process

- Examine each slice and set a flag for each material that occurs in the slice
- Set a bounding box for each material that occurs in each slice, big enough to hold the material and the size of the smoothing kernel.
- Copy and smooth 3 adjacent slices and put them in 3 temporal slice variables
- For all slices
    - Compute the new slice from the 3 temporal ones
    - Copy and smooth the next adjacent slice and shift the temporal slices
- Output the resulting weights

Holding the weight for each material in every grid point consumes a lot of memory, therefore the optimization has been made to just save the biggest

weight corresponding to the label in the grid point and letting the other weights be 1 minus the weight at that point, thus making the space needed σ times smaller.

2.1.2.1.4 Triangulation of the grid cells

When a labelfield with corresponding weights has been obtained, the triangulation of the grid cells can be performed in order to extract the surface. The labelfield together with its weights define a unique set of surfaces separating the different labels inside the cell. The surfaces are defined so that the two largest weights are equal on them when interpolated from the cube corners. Since tri-linear interpolation is a non-linear transformation, these surfaces will not be planar, therefore to represent these surfaces in a good and in computer graphics convenient way, a triangular approximation is constructed using bi-linear interpolation. This triangulation has to be consistent between the adjacent grid cells in order to avoid holes in the resulting surface mesh and it has also to preserve the topology inside the cells.

As seen previously, in the case of 2 different region types, 256 configurations are possible that can be represented of 14 topological cases, and in the case of 3 different regions an additional 44 is needed. To be able to handle any case, that is up to 8 different labels, a method is needed that can generate the triangulation of any case automatically.

Re-labeling

Before a triangulation can take place, the labels in the cells corners has to be re-labeled. In a labelfield an arbitrary number of materials can be defined, but in order to make the triangulation algorithm easier to implement, and to make the look-up table easier to use, the materials are given labels from 0 to 7, as the maximum number of different labels that can exist in a grid cell is 8. This gives the advantage that the triangulator just has to deal with 8 different materials since the re-labeling is done before the triangulation. When the triangulation of the cube is done, an inverse re-labeling is done to obtain the original labels.

Subdivision of the grid cells

The method used in the GMC is based on a simple sub-division approach. To obtain a representation of the inner structure of a grid cell, the cell is subdivided into a number of smaller cells. For each sub-cell the maximum weight is interpolated at a point placed as in figure.

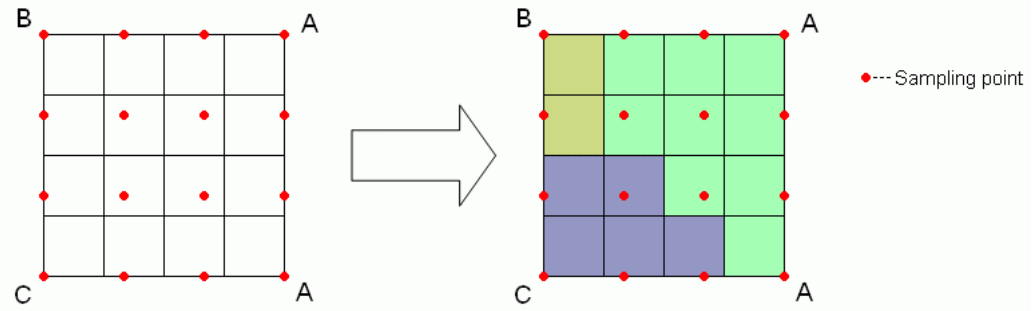


Figure 15: *The picture shows an example of subdivision by 4 in the 2d case and the outcome of the evaluation of the sub cells. The sub cells memberships are determined using linear interpolation of the weights in the four corners in the sampling points in the picture, giving the membership of the sub cell to the label that has the maximum interpolated weight. The letters in the corners represent the different labels.*

Note that the placement of the sampling points is translated from the center of the sub cells to lie on the cells faces and boundaries. In this way consistency can be guaranteed between adjacent grid cells since the points on the faces and the boundaries are shared with the adjacent grid cells. The sub cells are classified to the maximum weight found in their corresponding point. Whenever two sub-cells are of different classes, their common face is added to an intermediate triangulation as shown in the picture, which later will be simplified.

Finding patches

When the intermediate triangulation is finished, the next task is to find and classify the patches contained in the cell. All connecting triangles separating the same vertex classes are grouped into patches. Then the boundary curve of each patch is extracted.

For each vertex in the triangulation the number of boundary curves it belongs to is determined. For a point inside a patch this number is then zero and they are not longer needed to create the final triangulation. Points belonging to more than one boundary curve are named branching points.

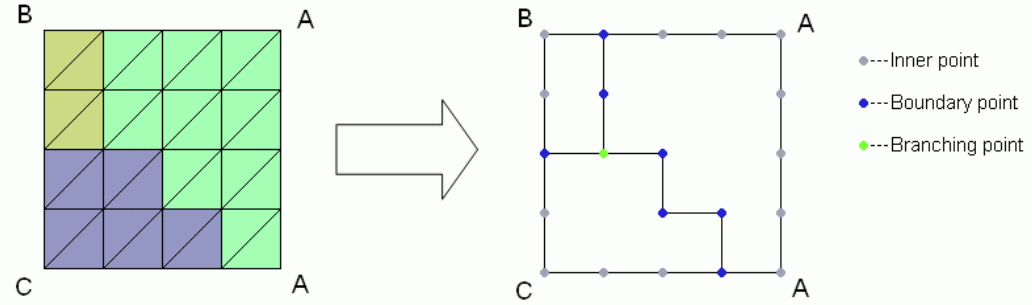


Figure 16: *The picture shows an intermediate triangulation on the left and on the right the boundary curve of each of the three patches together with the classification of the points.*

Simplifying the boundary curves

In order to produce the final triangulation, the intermediate triangulation has to be simplified. To make this, only a few of the produced vertices are of real

importance namely the ones that are branching points and those who lay on the cells edges. This because the branching points reflects the cells inner topology and the vertices on the edges are referenced by multiple cells. All the other vertices will only increase the complexity on a sub-cell level and are not needed to produce a topological correct result. After this is done the boundary curves of the patches consists of only those vertices, normally 3 or 4, but cases with more vertices also exists.

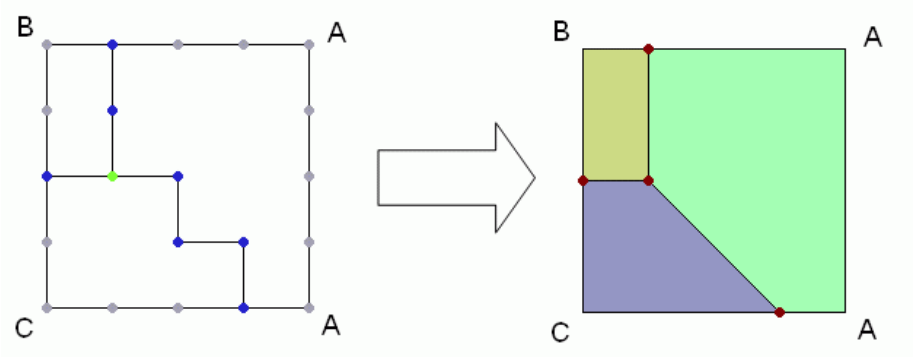


Figure 17: *The resulting triangulation after the simplifying step.*

Retiling patches

After the simplification of the boundary curves is completed, the patches are once again retiled with triangles. If a patch is planar, this can be done using a simple *anchor point* strategy. When this is not the case, special care has to be taken while unfavorable triangulation in combination with the simplification of the boundary curve can introduce penetrating triangles. This is avoided by introducing a center vertex into patches with 5 or more boundary points and using a *fan-type triangulation* strategy.

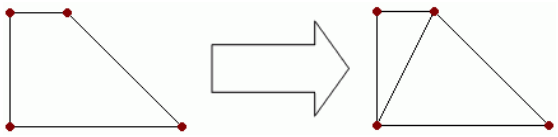


Figure 18: *Anchor point strategy, using one point as anchor, edges can be connected from this point forming triangles.*

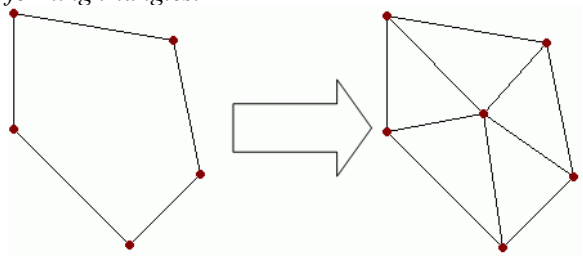


Figure 19: *Fan-type strategy, placing one point in the middle of the polygon edges can be defined between this point and the points on the polygon to create triangles*

After the retiling of the patches the basic triangulation is done and the different regions in the labelfield are separated by surface patches. Triangulating cells is a quite time consuming task, and many triangulations are topologically equivalent. This calls for the use of a look-up table in order to reduce the number of triangulations to a minimum.

The look-up table
In order to save memory the look-up table is created on the fly. That means, when a cell needs to be triangulated and it consists of 2 or 3 materials, the look-up table is queried to check if any topological equivalent already has been triangulated. If so, then the triangulation is returned from the look-up table. If no triangulation is previously calculated for this topological case the triangulation is made using the algorithm previously described and added to the look-up table.

The table doesn't store the exact position of the points on the edges and the faces, it merely stores whether a face or an edge contains a point or not, just as after the retiling step. Therefore the points must later be shifted on the edges, faces and sometimes even the inner points that may exist.

For a complete overview of the different triangulation cases that exists, a table is set up in the appendix B.

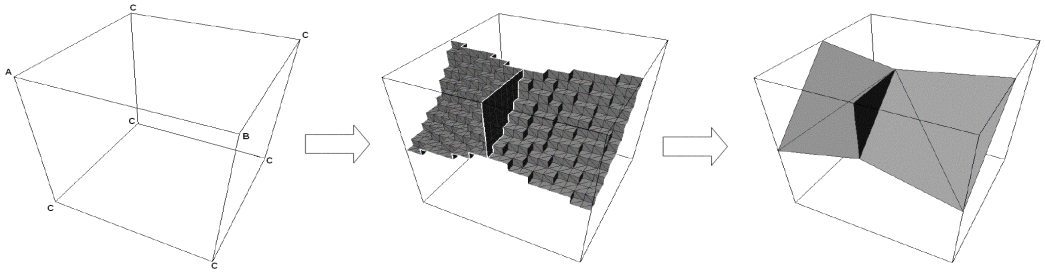


Figure 20: *Example of triangulation in the 3D case. Where the first cube shows the labels, the second the intermediate triangulation and the third the final triangulation*

A schematic overview of the triangulation process

- For all grid cells
  - If the cell contains three different materials or less and a topological corresponding case already has been triangulated
    - Choose the triangulation from the lookup table
  - Else
    - Triangulate the grid cell.
      - Do a sub-division of the grid cell
      - Do an intermediate triangulation of the grid cell
      - Group the triangles in patches
      - Simplify the boundary curves of the patches, saving only boundary points on edges and branching points
      - Re-tile the patches using fan- or anchor point strategy
      - If the cell contains three different materials or less, save the triangulation in the lookup table

2.1.2.1.5 Shifting the points

The result obtained from the previous steps is triangulated grid cells with a set of points defined on and in them. The surfaces that these cells define is a topological correct one, and valid since the separation of all regions is correct. However, the location of the points is just determined to be on the center on the edges, faces and the interior by the triangulator. A quick solution to this would be determining the location of the points in the sub-division step, but it would give a non exact result in respect of the weights as the sub division just could give an approximate location of the points. Instead, using the weights that are defined for each point in the corner of the grid cells, the points could be shifted to a more exact position, giving a better result for every grid cell and resulting in more exact and smoother surfaces.

The algorithm goes through all points and first shift all points that are located on edges, then the points located on the faces (at most one point per face) and last it relocates the inner points. The reason of doing the shifting in this order is that the position of the inner points depends on the position of the face and edge points, and the position of the face points depends on the position of the edge points. Thus, shifting the points in this order saves calculations.

Bilinear interpolation

To determine where on an edge a point should be located, the weights in the cubes vertices belonging to this edge are used. If weights for more than two materials exist at the vertices, determining where the point should be located turns in to a quite complicated matter. To remedy this, the simplification is made that all but the maximum weights at each vertex are set to be equal. Then only two different weights occur at a vertex, the primary that is the strongest weight and the secondary representing all the others. Making this simplification also gives the advantage that the amount of memory needed for storing the results from the segmentation is reduced drastically.

Now the location of a point on an edge can be determined by bilinear interpolation.

Setting up the equation system:

$$y = (w_{b2} - w_{a1})x + w_{a1}$$
$$y = (w_{b1} - w_{a2})x + w_{a2}$$

Gives the intersection point

$$x = U = \frac{w_{a1} - w_{a2}}{w_{b1} - w_{a2} - w_{b2} + w_{a1}}$$

which is the new location of the point.

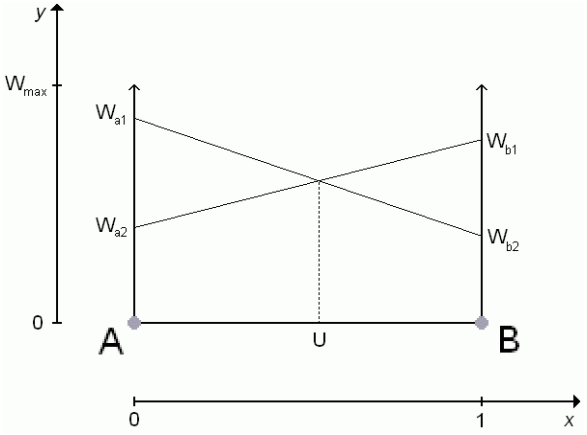


Figure 21: *Bilinear interpolation done geometrically between two vertices.*

In the implementation of the GMC the weights are chosen so that $w_2 = 1 - w_1$ to save space. This is guaranteed to be correct since the primary weight w_1 is always bigger than the secondary weight w_2. Only one case exists where the interpolation fails, that is when the denominator is zero. In this case U is set to 0.5 which is the logical due to the fact that the denominator is zero when the weights are the same.

Edge shift

When a point lies on an edge, the new position of the point is calculated using bilinear interpolation of its weights as explained in the previous section. Shifting the points on the edges gives a smoother result than just letting them be on the middle of the edge, which would give an 'edgy' surface as result.

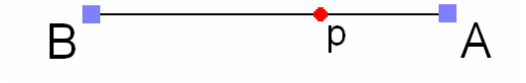


Figure 22: *The figure shows an edge from a grid cell with the point to be shifted in red.*

Face shift

The shifting of the points on a face is somewhat more complicated than the shifting of the points on the edges, there exist two different topological cases when points on faces occur. One case is when three edges of the face have points on them and one is when all four edges of the face have points on them.

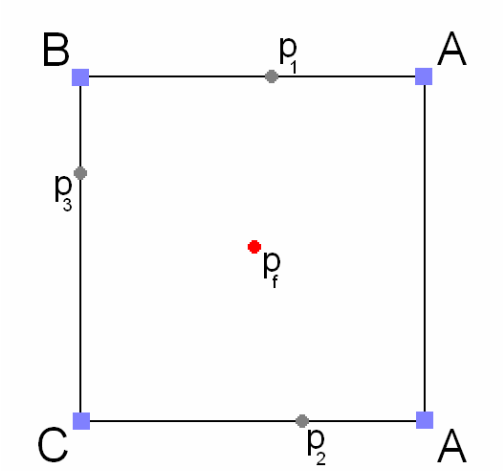


Figure 23a: *The figure shows the face of a cube with three different materials present (A, B, C) and the points on the edges that separate these materials. The red point is the point to be shifted on the face.*

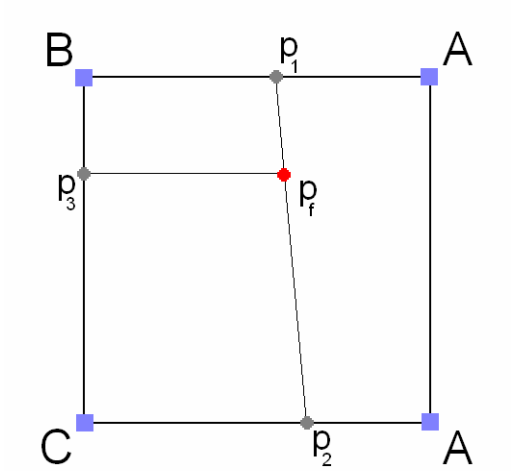


Figure 23b: *The new location of the point after the shifting has been made.*

When three edges has points on them 4 different configurations are possible, but all are equal if the face is rotated. At first an interpolation is made of the weights at the vertices of the face to get the corresponding weights for the points p_1 and p_2. Then the fractional value U is calculated with bilinear interpolation of these weights. The point on the face p_f is placed on the straight line that can be drawn between p_1 and p_2:

$$p_f = p_1 + U * (p_2 - p_1)$$

Note that the position of the point p_3 is not considered at all when calculating the new position of the face-point.

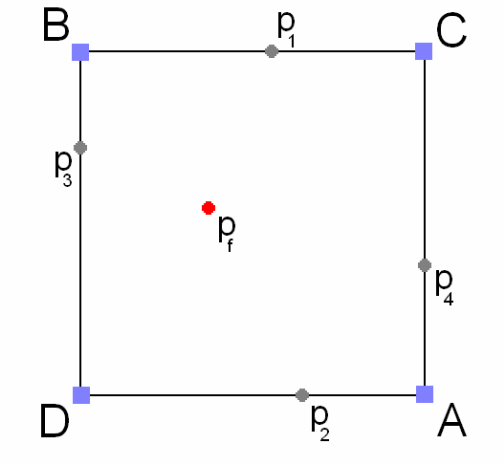


Figure 24a: *The figure shows a cell face where four different materials meet.*

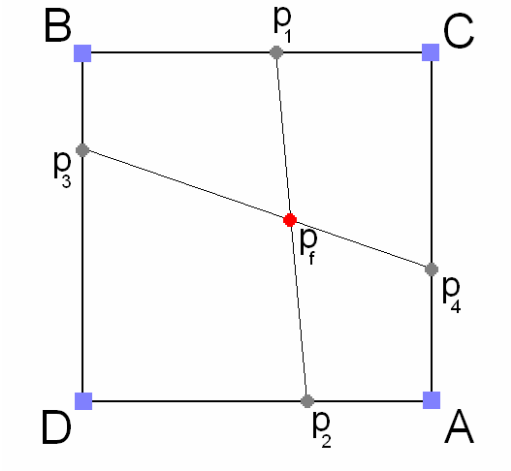


Figure 24b: *The placing of the face point is in the intersection of the lines drawn from the points lying on opposite edges.*

The case when all four edges have points on them is solved by computing the intersection of the lines drawn between the points that lie on opposite edges. The face-point is then placed where these lines intersect.

Inner shift

The inner points are placed in the position given by the mean value of all the neighboring points. The neighboring points to an inner point p_i are all the points that are connected to p_i with edges. Thus, the placing of the points during inner shift is not directly dependent on the weights, but indirectly because of the placing of the neighboring points, the points that lie on the faces and the edges, are determined by the weights.

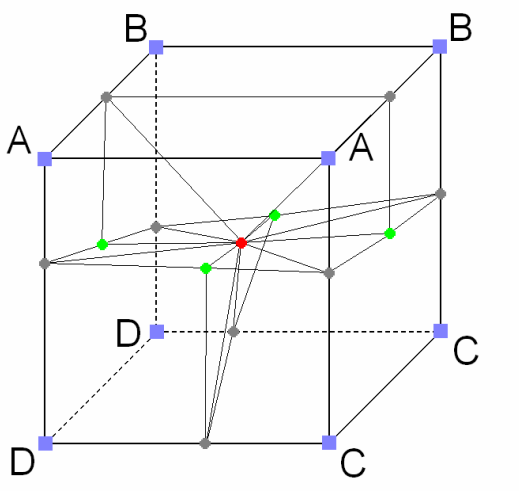


Figure 25: *In the picture the entire grid cell can be seen with the inner point marked with red and the face points with green. The triangulation is also shown with the thinner lines representing the edges between the points. Notice how the points separate the different materials in the cube.*

All the shifting of the points in the surface model can be done whenever desired, assuming that the weights still are defined for all the points in the labelfield. However, to save memory, in the implementation this is performed on each cell directly after its triangulation.

2.2 Our modifications

Motivation

The motivation for modifying the GMC algorithm was to correct the problem of the ridges that were created when 3 materials meet in one cube cell. The idea was to give the user the ability to pick a *special smooth material* that should be generated to be smooth, and have no ridges on it. A natural example is the example of when bone, muscle and fat meet each other in a model. Bone is considered to be smooth due to its natural properties and therefore it is more natural if the muscle and fat follows the bone in such a way that no ridges occur, than that all materials are interpolated to follow each other.

The idea

The ridge problem occurs when three or more materials meet in a cube cell. Eliminating all materials except for the special material and the material that

defines the outer region would yield an optimal calculation of the surface of the special material, since just two materials would be present.
The idea was to change the algorithm so that it generates the surface triangles that belong to the special material as if no other materials were present in the model. Using this strategy leads to distortions in the other materials present, since the shifting of the points will have to be modified, but it was agreed upon that the benefits of the change would be greater than the drawbacks of these distortions in the model.

Changes in the smoothing process
To be able to generate the surface of the special material without distortions from other materials, the original weights have to be saved. This is due to the simplification step previously mentioned, where only the dominating weight is saved and a secondary weight is generated to represent all the other materials. In the implementation of the GMC the weights are represented by a byte, giving them values in the range of 0 to 255 where 255 is the maximum weight.

The smoothing process was modified in such a way that:

1. A vertex is always re-labeled to belong to the special material when the weight of the special material, w_{sm} is such that $w_{sm} \geq 128$.
2. A vertex can't be labeled to belong to the special material if $w_{sm} < 128$.
3. An extra table is created containing just the weights for the special material

When forcing the algorithm to re-label a vertex to belong to the special material if w_{sm} is 128 or bigger, and not allowing vertices labeled to belong to the special material when w_{sm} is smaller than 128, the algorithm behaves as if *only two* materials existed. This is due to the fact that the maximum allowed weight is 255, and if a labelfield contains only two different materials and one of them has the weight 128 in a vertex, the value of the other must be 127. In our case this would be the same as running the smoothing process on a labelfield containing just the special material and the outer material. Using this approach the outer material can be considered to have the weight 255 minus the weight of the special material. The weights are also saved for the special material, making it possible to perform the shifting of the points in a manner that the surface of the special material is not distorted by the other materials. This makes the changes of the smoothing process complete.

Changes in edge shift
The weights saved in the smoothing process are used if one of the vertices that lie on the treated edge is labeled to belong to the special material. This way the shifting of the points on the edges will be done in the same manner as if the special material was the only material existing. When none of the vertices are labeled to belong to the special material the shifting is done as normal, using the normal table of weights.

Changes in face shift

There exist two basic configurations to consider when the special material is present in a face shift; when two corner vertices belong to the special material and when one corner vertex belongs to the special material.

1. Configurations with two vertices belonging to the special material

When two vertices belong to the special material, two different configurations are available.

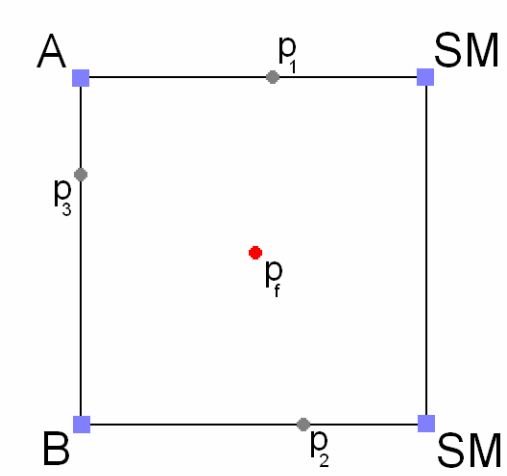


Figure 26a: *A cell face where two vertices belong to the special material*

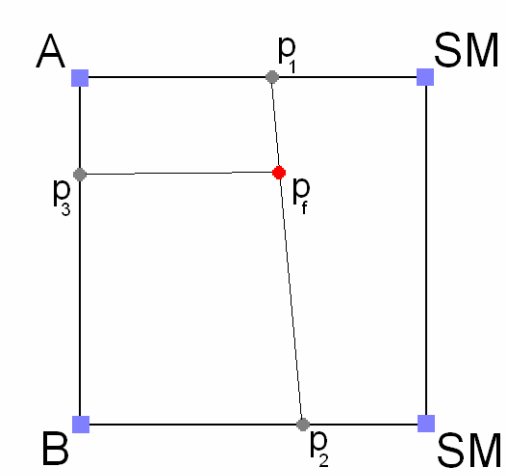


Figure 26b: *The final placing of the face point giving a straight line as border for the special material*

The case when three edges have points on them, the vertices of the special material are placed so that they are on the same side, sharing one edge. In this case, no changes have to be done, since the implementation of face shift is such that the face point p_f will be placed on a straight line between the two points separating the special material from the other materials. Thus, creating a straight border towards the other materials as also would be the case when just two materials were present. In the case when all four edges have points on them, the placing of the vertices labeled with the special material are located on opposite corners, creating a so called checkerboard case. This case is handled by a special method that resolve the ambiguities and it is not handled in faceshift.

2. Configurations with one vertex belonging to the special material

If just one vertex is labeled to belong to the special material and three edges have points on them, two basic configurations are available with their rotational equivalents. The two cases are shown in the pictures below. In order to make the implementation easier, the points p_3 and p_2 are re-named to take each others name in the second case, giving just one case and its rotational equivalents to consider.

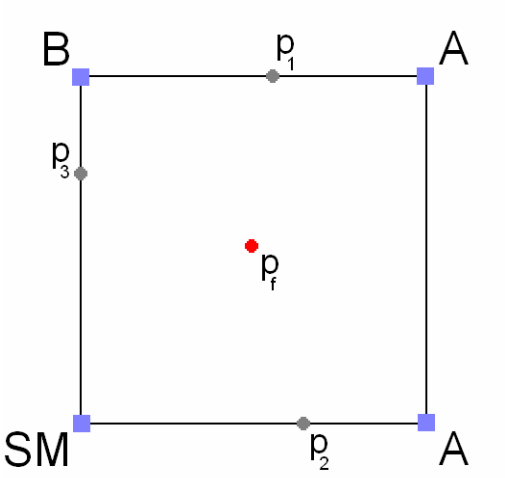


Figure 27a: *Configuration case 1*

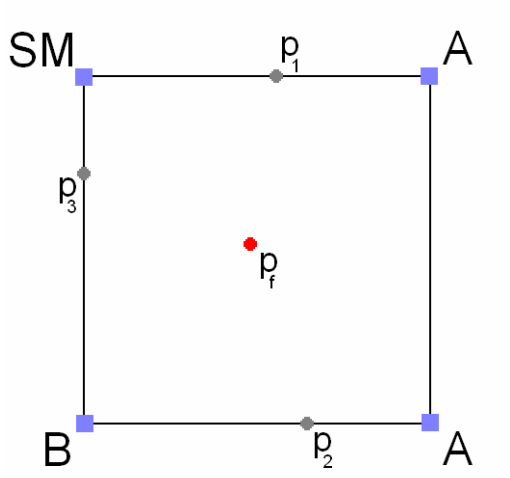


Figure 27b: *Configuration case 2. If p_1 and p_2 are switched we get the same configuration as in case 1, considering the special material. The same points are used for separating the special material from the others.*

The objective of the changes is to simulate that only the special material and one material representing the outside are present in the model. If this was the case, a straight line would have been drawn between the points p_3 and p_2, separating the special material from the outside. In order to recreate this, p_f has to be placed somewhere on this line separating the special material from the others, namely the one created by p_3 and p_2. The ideal placement of p_f is achieved when $(p_f - p_1)$ is orthogonal to $(p_3 - p_2)$.

To make a solution that works for all the 4 rotational equivalents of the face and that is independent of the coordinate system, vectors are used.

Let

$$\bar{u} = p_2 - p_3$$

and

$$\bar{v} = p_1 - p_3$$

now, the orthogonal projection of $\bar{v}$ on $\bar{u}$ is

$$\bar{v}_p = \frac{\bar{v} \cdot \bar{u}}{|\bar{u}|^2} \bar{u}$$

then the ideal placing of the face point is

$$p_f = p_3 + \bar{v}_p$$

The idea is illustrated in the pictures on the right.

However, the idea only works when the angle α between $\bar{u}$ and $\bar{v}$ is such that $45° \geq \alpha \geq 90°$.

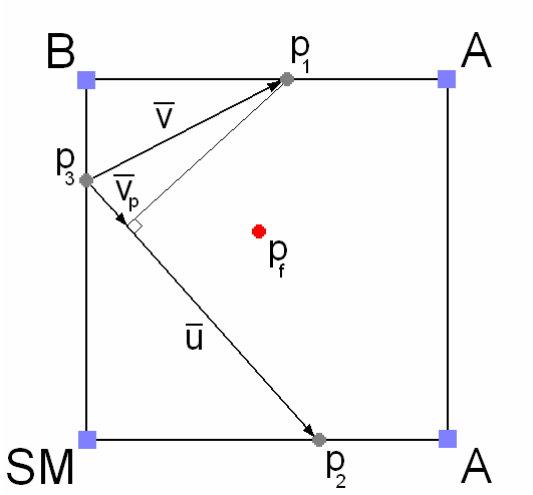


Figure 28a: *To relocate the face point the orthogonal projection of $\bar{v}$ is made on $\bar{u}$.*

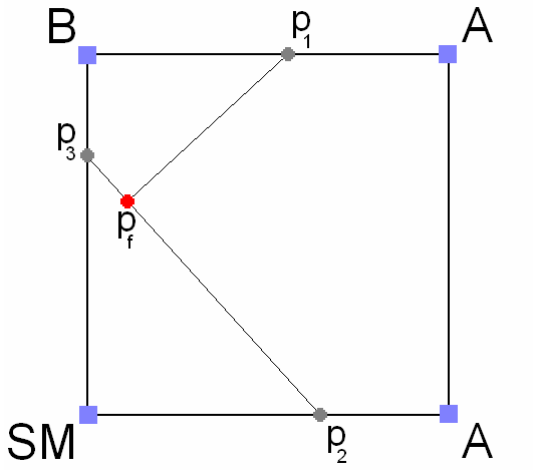


Figure 28b: *Placing the face point in the position $p_3 + \bar{v}_p$ gives an angle of 90 degrees between the lines.*

In the other cases when α is outside this interval, the face point has to be placed in either one of the extremes of the line between p_3 and p_2. This can be detected by looking at $\bar{v}_p$. When $\bar{v}_p$ is pointing in the opposite direction of $\bar{u}$, $(\alpha < 45°)$ the face point is placed in p_3. When the length of $\bar{v}_p$ is greater than the length of $\bar{u}$, $(\alpha > 90°)$ then the face point is placed in p_2. However, the placement of the face point in p_2 or p_3 gives degenerate triangles. In the case of moving the face point to p_3, the triangle $[p_1, p_f, p_3]$ will be degenerate, and in the case of moving the face point to p_2, the triangle $[p_1, p_2, p_f]$ will be degenerate. Normally such a situation should be avoided by removing these

triangles right away since we know that they are degenerate. However, due to the structure in which the triangles are indexed and stored in the implementation of the GMC, this is a quite time consuming and not so easy task. Instead of doing this at every single instance when a degenerate triangle is generated, the problem is remedied when the triangulation of all cube cells are finished. In this step the surface is simplified and all such triangles are removed.

The last case occurring in face shift is when four edges have points on them, or in other words, when four materials meet in one cell face. This is a quite unusual case and does not occur too often even though more than three materials might exist in the labelfield. The solution to this case when no special material is present consists of just calculating the intersection point p_i given by the intersection of the two lines constructed of the points on opposite sides of the face. Then the face point is placed in this intersection point. However, when the special material is present in one corner, using this solution would distort it. In order to simulate that the special material is the only one present the face point must be placed so that the border between the special material and the other materials is a straight line.

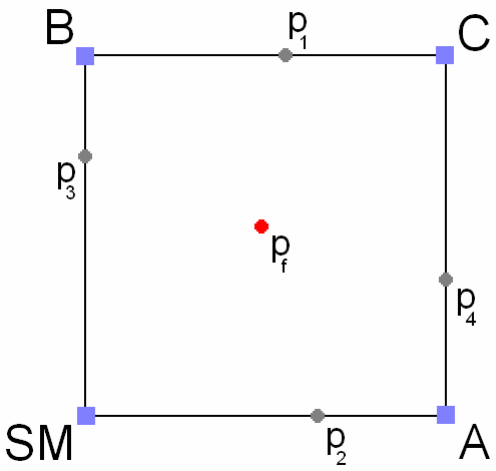


Figure 29a: *Four materials meeting on a face giving an intersection at each edge.*

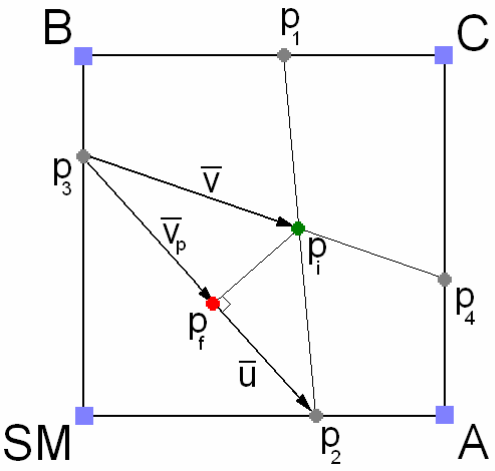


Figure 29b: *The vector $\bar{v}$ is projected on $\bar{u}$ giving the vector $\bar{v}_p$ that determines the new location of the face point p_f. p_i, the intersection point is used to construct $\bar{v}$.*

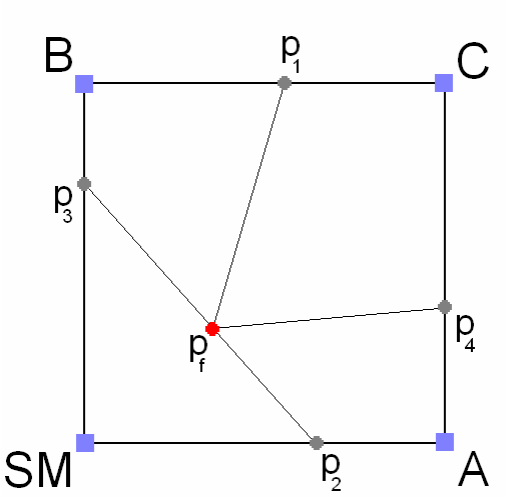


Figure 29c: *The final placement of the face point* p_f. *Note the straight border between the special material and the others.*

In order to not disturb the balance between the other materials, the face point is placed on the line between p_3 and p_2 a distance $|\bar{v}_p|$ from the point p_3 where $\bar{v}_p$ is the orthogonal projection of $\bar{v}$ on $\bar{u}$. This gives the exact properties of the solution to the configuration just explained above. The same restrictions apply as well when the projection gives a resulting vector $\bar{v}_p$ so that the placement of the point p_f would be outside the face.

Changes in inner shift

Inner shift is originally implemented to work so that the inner point is moved to the average of all the other points it is connected to. In the case when only the special material and the outer region exist, there would just be one surface patch separating the special material from the outer region.

In order to simulate this situation, the inner shift is separated in two cases

1. When the cube cell does not contain the special material
2. When the cube cell contains the special material.

In the first case, the inner shift is done as normal, the points that are neighbors to the inner point are collected and the mean value is computed. The inner point is then placed in this position.

In the second case, all the points that together with the inner point define triangles that separate the special material from the others are collected and the mean value of them is computed. Meaning all the points that are neighbors to the inner point, and that are located on patches that separate the special material from the other ones are extracted. In this way all the materials apart from the

special one are treated as if they were part of the outer region, giving the desired result.

A schematic overview of the GMC after the modifications[1]

- Examine the labelfield to find out which materials exists
- Smooth the labels in the labelfield to obtain weights
  - Save the weights of the special material in separate table
- Scan two slices at a time and create a grid cell from four points in one slice and four points in the next slice of the labelfield
- If the cube contains at most 3 different labels, look in the look-up table for the triangulation. If the cube contains more labels, create the triangulation with the triangulation algorithm
  - Shift the points in the cube. If the special material occurs in the cube, use the weights saved in the table and use the strategies developed to simulate the presence of just the special material and the outer region
- Continue with the following slices

2.3 Result

The purpose of the changes to the GMC was to let the user select one material that should be treated to be smooth (the special material) without ridges. The ideal case was chosen to be the surface generated with the original GMC with just this special material and the exterior present, due to the fact that no other materials then could distort the surface. The results have been chosen to be evaluated on a single labelfield.

2.3.1 The evaluation model

The model chosen for the evaluation is the *zahn mit kortikalis* model. This is a good example because of the easy structure of the model and the visual aspects, making it easy to distinguish the ridges and other anomalies.

The model is generated from a labelfield of the dimensions 143x239x24 and the ready model contains of 57486 points making up 115350 faces distributed on 5 patches. Below the whole model is shown with its different parts. It consists of four different materials, the dentin, enamel, kortikalis and the exterior. As seen in the pictures dentin is in connection with three different materials at two patch boundaries; where it connects to enamel and where it connects to kortikalis. Therefore dentin is chosen to be the special material in the test.

[1] The modifications are marked with a non-filled circle

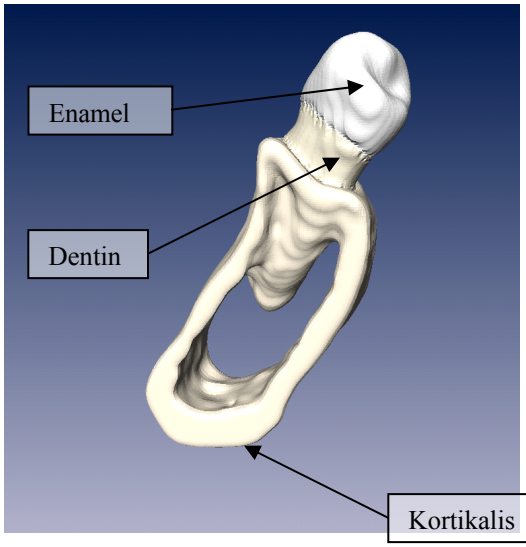


Figure 30a: *The whole model generated by the original GMC*

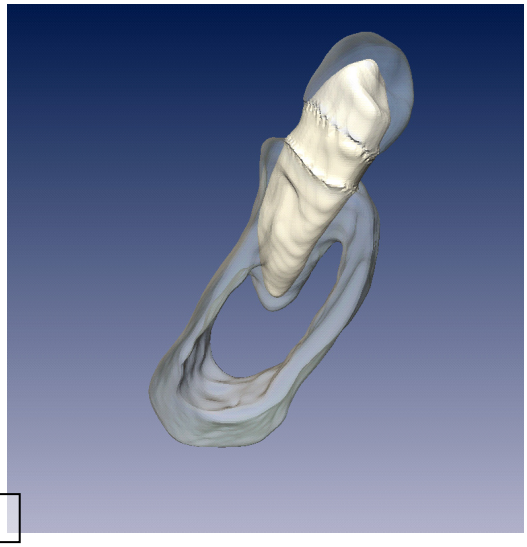


Figure 30b: *The same model shown with enamel and kortikalis in transparent view. Note how the dentin surface resides inside the kortikalis and enamel surface*

2.3.2 Visual evaluation

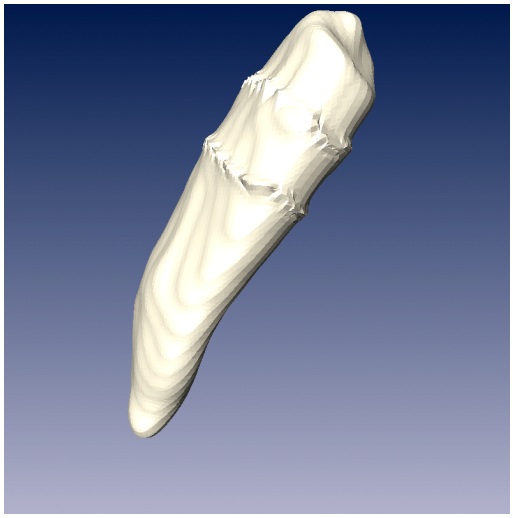


Figure 31a: *Dentin as it looks like when generated with the original GMC*

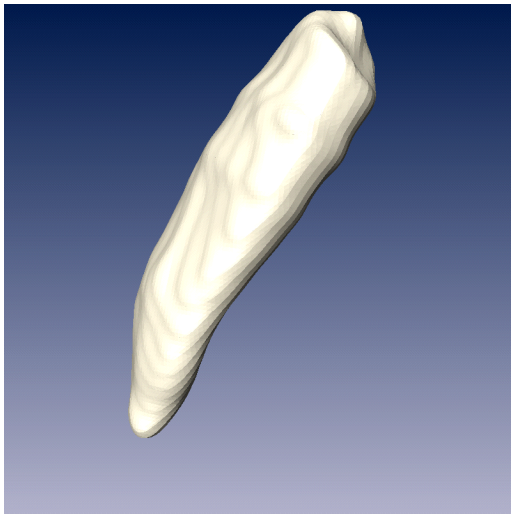


Figure 31b: *Dentin when generated with our modified GMC and selected to be smooth*

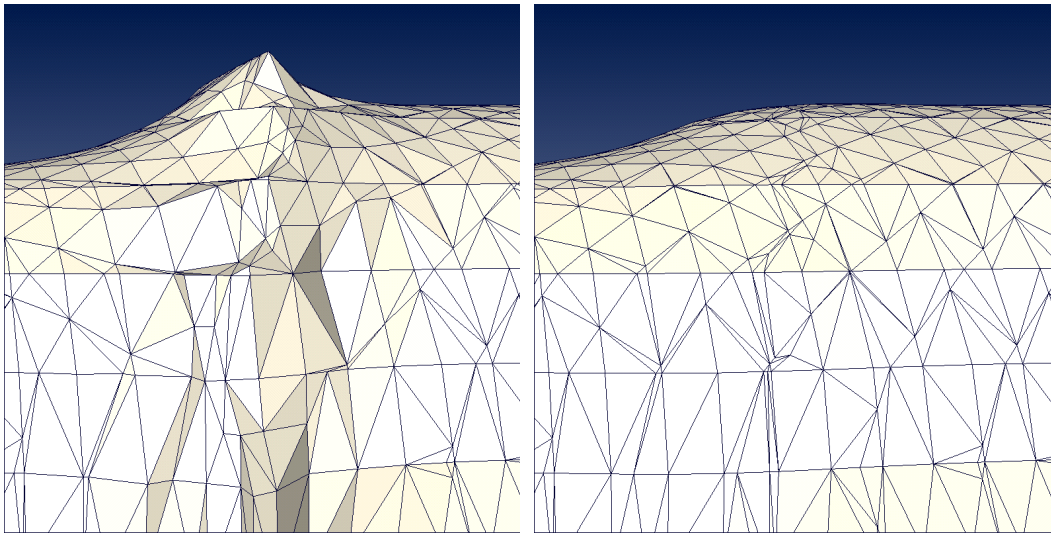


Figure 32a and 32b: *Zoom in on the lower ridge on dentin with the triangle edges visible. The left picture shows the result when the original GMC is used. The picture on the right shows the same situation when the modified GMC is used.*

As seen in these pictures the difference of the surfaces generated by the original and the new version of the GMC are quite big. The original version gives a very strong ridge effect where the three materials meet, and the new GMC gives a result without these effects.

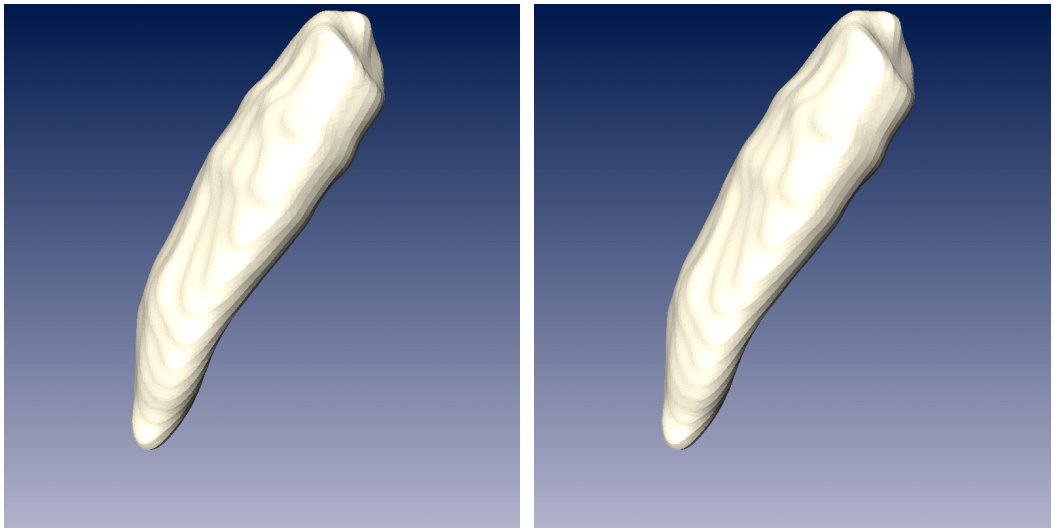


Figure 33a and 33b: *The ideal dentin surface is shown on the left and the surface generated by the modified GMC on the right*

As seen in these two pictures, there is virtually no difference between the ideal surface and the one generated by our modified GMC. It is very hard to point out any differences.

2.3.3 Mathematical evaluation

The visual evaluation is not always fair as it can be very difficult spot the differences without previous knowledge of the models. More exact results can be obtained if the difference between the surfaces is evaluated statistically.

To the right a snapshot of the material dentin is shown, when generated with the original GMC from a labelfield with just this material. To be able to interpret the statistical results in a better way regarding the distance between the surfaces generated by the original GMC and the modified one, the dentin surface has been measured to its diameter and length with the results 0.6717 for the diameter and 2.0452 for the length.
Due to the fact that every model has its own coordinate system deviations in the surface models can't be compared directly between them.

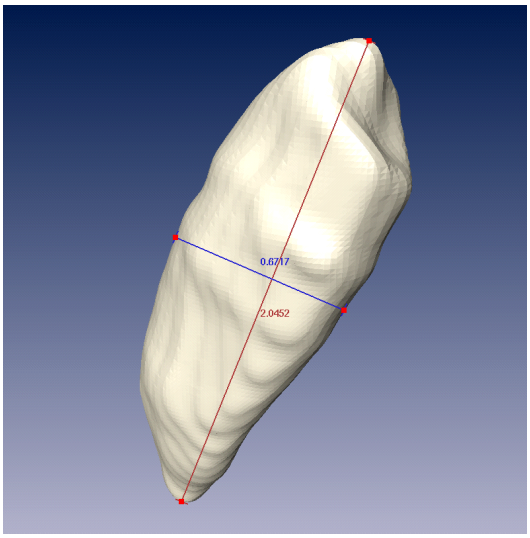


Figure 34: *The figure shows the dimension of the dentin surface. The blue line has the length 0.6717 and the red one 2.0452*

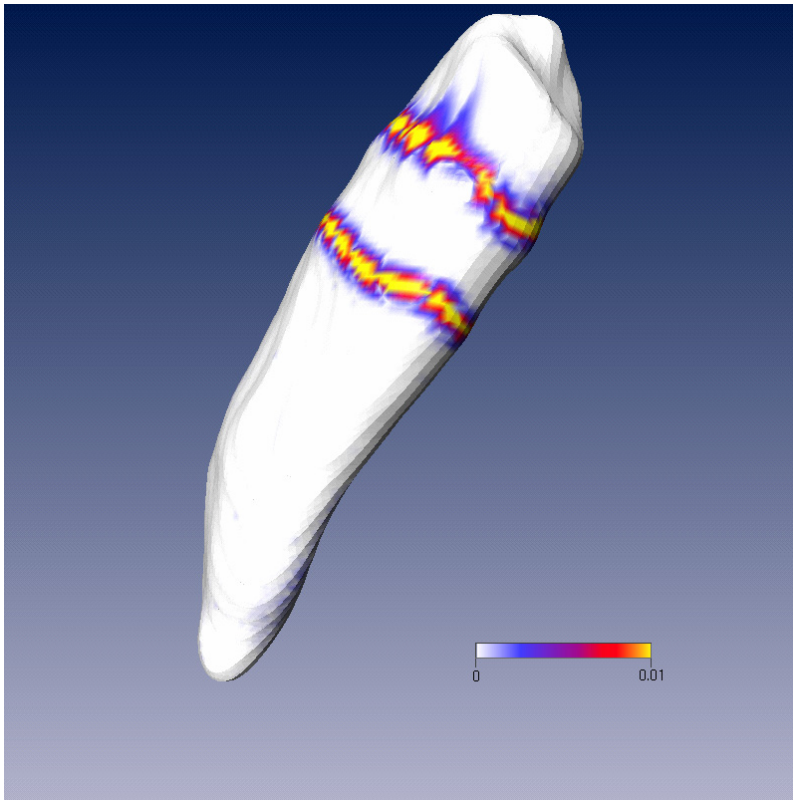
Moreover, due to the lack of length units in the coordinate system, some reference values have to be established to make the comparison meaningful.

2.3.3.1 Surface distance

In Amira there exists a module for measuring the distance between two triangulated surfaces. For each vertex of the surface the distance to the closest point on the other surface is computed. From the histogram of these values the following is computed:

Mean distance: $\overline{\delta}$
Standard deviation from the mean distance: σ
Root mean square distance: rms
Maximum distance: $\delta_{\max}$

The output of the module is a distance field, meaning a field of values that for every vertex in the source surface contains the minimal distance to the target surface. This distance field can be connected with the surfaceViewer module to produce a visual result of how the surfaces differ from each other.



Surface distance:

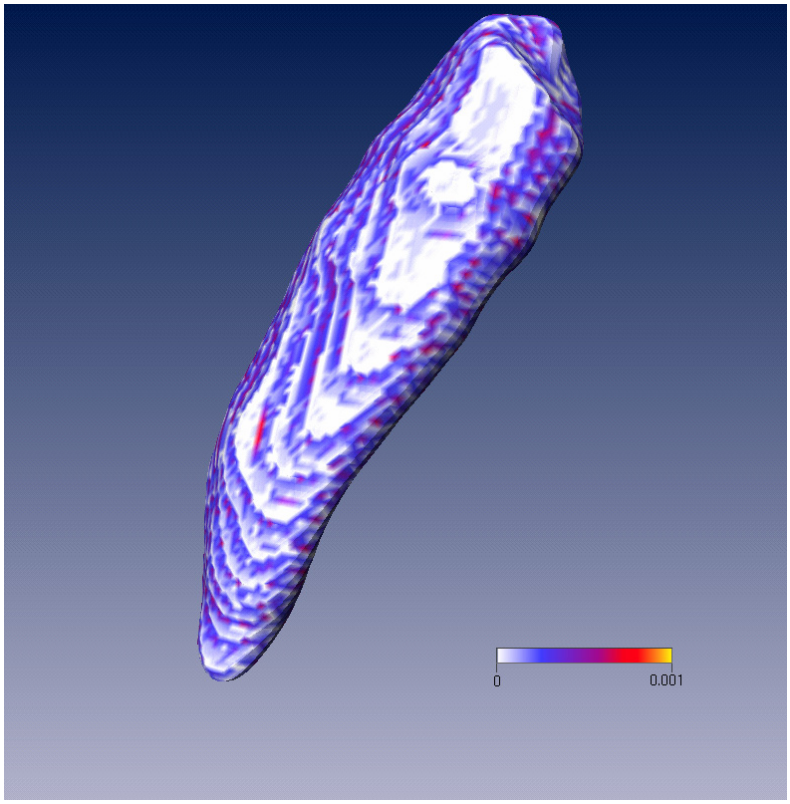
Ideal surface → surface generated by original GMC with all materials present

$\bar{\delta} = 0.00142141$
$\sigma = 0.0034648$
$rms = 0.0037449$

$\delta_{\max} = 0.0203537$

Figure 35: *The distance field visualized together with the ideal surface. Note that the scale is set to be ten times bigger in the next picture. The surface distance is a color marked indicator which spans from white (no difference [0]) through blue and red to yellow (maximal distance [0.01]).*

As seen in the picture, the ridges present a serious distortion to the model, even though the color scale is set to be ten times bigger. Setting the scale to the same values as in the following example is useless since a lot of values bigger than the scale. The mean distance should not be considered since the surfaces are exactly the same everywhere but where the ridges occur, and therefore gives no fair comparison. Although this is the case, the mean distance is 0.21% of the dentin diameter which is bigger than the mean surface distance when comparing the results of the modified GMC with the ideal surface. This gives an idea about how strongly distorted the model is by the ridges.



Surface distance:

Ideal surface → surface generated by modified GMC with all materials present

$\bar{\delta} = 0.000529077$
$\sigma = 0.00172168$
$rms = 0.00180108$

$\delta_{max} = 0.0108175$

Figure 36: *The distance field visualized together with the ideal surface. Note that the scale is ten times smaller than in the last picture. The surface distance is a color marked indicator which spans from white (no difference [0]) through blue and red to yellow (maximal distance [0.001]).*

As seen in the picture, the deviation is not especially strong at any part of the surface, but more generally distributed all over. The deviations are not stronger where the ridges earlier were found. The small deviations found that are scattered all over are results from the difference in the smoothing process and also the triangulation. In the smoothing process the different materials can affect each other giving a slightly different result than the two material case. The differences in the triangulation come from that placing a vertex on a straight line, may cause it to be distorted a little bit because of the limited precision of the floating point numbers in computers.

All in all, the goal of recreating the dentin surface as in the ideal case with all materials present has been reached. The differences are so small that very strong zoom has to be applied to see them. The mean distance is just 0.079% of the dentin diameter which is a very low value.

2.3.3.2 Surface Curvature

Another way to study the results is to look upon the curvature of the surface. This can be done with a surface curvature module which looks the surrounding to each vertex and calculates the maximum shifting at this vertex. Therefore it is

a measurement of how irregular the surface is. The intensity of the shifting is used for color marking the surface.

However in this case this method is not useful because of the similarities between the compared surfaces. But as we will see later this method comes to use during the comparisons in the post-processing step.

2.3.4 Advantages

The initial goal of generating surfaces without ridges that was set out in the beginning has been reached. The surfaces generated by our modified version of the GMC shows virtually no difference from the ideal ones. The advantage of taking on the problem as it occurs is obvious. The speed of the algorithm is only affected in linear time, as well as the space. With this version of the GMC smooth surface models can easily be created without ridges. Furthermore, the advantage of changing the GMC is that the process can be automated, without the user having to bother to look for ridges in a second step when generating surfaces. Some changes to the other surfaces in the labelfield also occurs, at most times to the better, while the other surfaces connect at the boundary where the ridge appears, making the boundary between the two materials more smooth.

2.3.5 Drawbacks

Even though the objective has been reached and smooth surfaces without ridges can be generated automatically with the new GMC, some drawbacks exist. The most import one is that only one material can be chosen to be smooth. This is due to the fact that if two or more materials could be chosen to be smooth, and if they lie close to each other in the labelfield, ambiguities would occur when smoothing the labelfields and when triangulating. It may also result in surface intersection. Therefore, choosing more than one material is out of the scope in this thesis.

2.4 Further work for the GMC

There exist several ways of improving the GMC. One improvement might be to allow the user to select more than one material to be smooth. As this might introduce problems such as surface intersections, the materials have to be chosen with care. One way to do this would be to check in the labelfield how close the materials chosen to be smooth are, and if there's a risk for any surface intersection. Another approach might be to create one labelfield for each material and look if they intersect each other after the smoothing. After this is done, the ability to select materials can be based on this information. A third approach could be to give every material a "hardness" value, where the material with the highest value in every grid cell is treated as the special material is treated now. These improvements are however not very easy to implement and may require time costly computations.

Another improvement could be to add a point in face shift when four materials meet. In this way, the sometimes sharp edges on the triangles defining the border of the materials could be avoided. This of course also leads to that new points have to be introduced elsewhere in the grid cell, depending on its triangulation.

3 Post-processing

The smoothing of polygonal models is an area where a lot of improvements have been done in the last 10 years. The reason is the rising demands for good visualisations from more and more researchers in different scientific areas. The fast development of computer processors and graphics card has made scientific visualization possible on even a cheap personal computer. The users demand correct visualisations of the studied datasets including smoothing and removal of noise from the models. As 3D-visualisation was a totally new way to interpret your data in the beginning of the 1990´s, suitable algorithms for smoothing polygonal meshes had to be developed. Lots of different methods have been suggested, but looking at the smoothing of polygonal meshes as a signal-processing problem has won the greatest achievements so far. An overview over the current research in this area can be found in [1]. Visualisation systems like AMIRA create large polygonal meshes when constructing isosurfaces from volumetric medical data or multiple range images. Because of the size of the typical data sets, only algorithms using linear time are of practical use as computation time has to be kept within the range of seconds, not minutes or hours. Different geometric signal processing approaches have proven to be successful in smoothing surfaces given the linear constraint. The algorithms treated in this section refer to this most common strategy of smoothing surfaces. We have implemented a modified version of an iterative algorithm invented by Taubin [5]. We will in the next sections show how the algorithm is derived, how it works and what modifications has been done to fit the special case of ridge smoothing. First we will present an ancestor to Taubin's algorithm, called the Laplacian smoothing algorithm and in doing so we will use a parallel between meshes and graphs. The smoothing of a mesh can in fact be viewed as a smoothing of a graph signal. Then we will show the limitation of Laplacian smoothing and motivate the use of Taubin's smoothing algorithm. At the end we will present our modifications to the algorithm and suggest other ways to solve the problem of post-processing ridges.

3.1 Introduction

3.1.1 The ridge

In the current version of the GMC-algorithm problems will arise when more than 3 materials meet in the same grid cell (see the chapter about the Generalized Marching Cubes algorithm). The unwanted result is the occurrence of ridges and deformations on the boundaries where many materials meet.

We will present an example of a ridge to demonstrate the ridge problem. If we for example look at the tooth-dataset we can clearly see the ridge on the visualized material 'dentin'. The ridge stands out quite a bit from the rest of the surface and this looks unnatural for a material known to be smooth, in fact the ridge deform the other materials as well. The red marked edges in the pictures shows the boundary between different patches on the 'dentin'-surface. A patch

is the term for all the triangles with the same combination of out and inside material, the 3 patches in figure 37a are: ‘dentin/enamel’, ‘dentin/air’ and ‘dentin/kortikalis’, also marked with different colors.

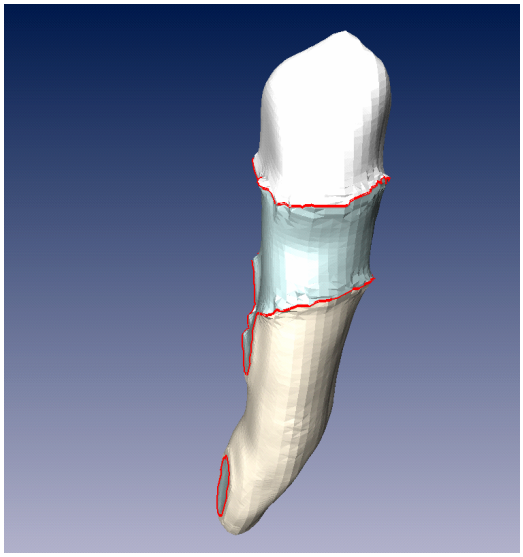


Figure 37a: *Visualized material ‘dentin’ from a tooth-dataset. The boundary between different patches is marked with red.*

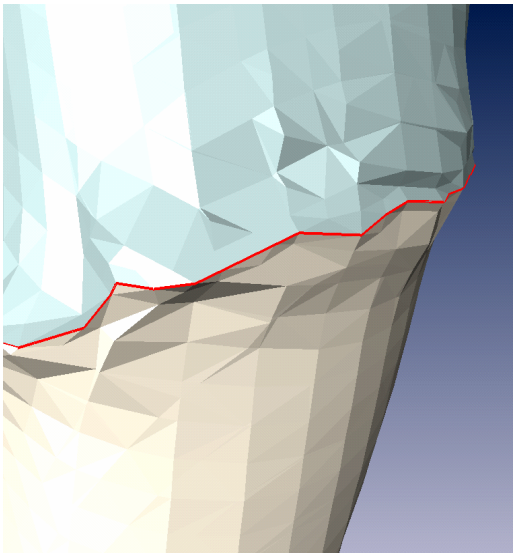


Figure 37b: *‘Dentin’ magnified. The ridge does not only affect the triangles closest to the boundary but surrounding triangles as well.*

The first step in the ridge smoothing is to identify these boundaries between the patches. When the boundaries are known are the locations of the ridges also known which are the places where the smoothing needs to be done. The ridge does not only affect the triangles closest to the boundary but the surrounding triangles as well as seen in picture 37b. This makes it hard to know if the underlying undulations belong to the original data or if they are a part of the incorrect visualization. The width of the ridge is not known and it is something that can change from dataset to dataset, making it hard to find an automated solution of the problem.

The purpose of the post-processing is to remove the ridge from the patches of the special material and in doing so also trying to not deform the patches from the other materials further.

3.1.2 A simple overview of the smoothing

A simple approach to reduce the ridge is to move all vertices lying on the boundary to the center of their neighboring vertices. By the neighboring vertices are meant the vertices which fulfill both of the following conditions:

- The vertex share the same triangle edge as the vertex on the boundary;
- The vertex is a corner on a triangle which patch has the special material on one of its sides.

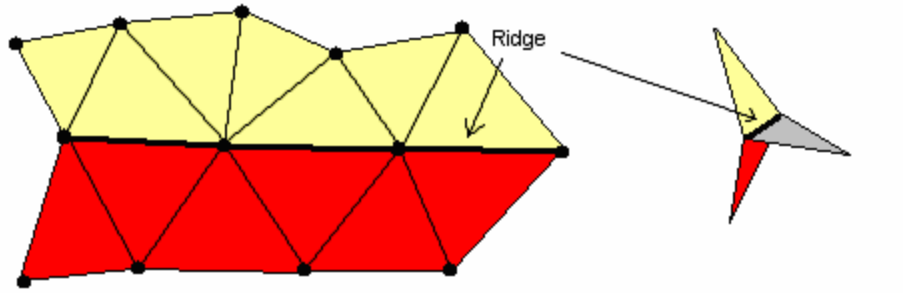


Figure 38: *An example of the different patches along the ridge; The yellow, red and grey patches are all different combinations of inside/outside materials. The yellow and red patches have the same inside material, in this case the special material.*

This simple smoothing step will remove some of the irregularities and can also be applied for the found neighbors and their neighbors and so on. This smoothing can be performed on a certain surrounding depth from the boundary vertices. The depth should be chosen large enough to cover the whole ridge. When all vertices on the ridge have been moved is it possible to start the smoothing all over from the boundary points again. In pseudo-code this would look something like:

For a number of iterations:

- Find all vertices lying on the boundary

  For each found vertex:

  - Find the neighboring vertices
  - Calculate the mean of the neighboring vertices

    For each neighbor:

    - Move the neighbor and its neighbors of a certain depth to a new position (recursive step)
  - Move the vertex position to the mean of its neighbors.

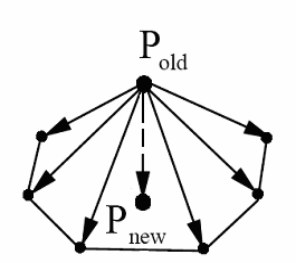


Figure 39: *An example of how a point on the boundary is moved to a new position calculated from the positions of the neighbors.*

This is roughly what happens during the smoothing, even though the actual smoothing algorithm is more sophisticated. In the next section we will consider the vertices of the mesh as nodes on a graph, and moving the vertices will be the same as smoothing the graph signal.

3.2 Theory

3.2.1 Graphs and graph signals

Taubin [5] introduces graphs and discrete graph signals as tools for describing the polygonal curves and polyhedral surfaces. From the mesh of vertices a *directed graph* is constructed, where each node represents a vertex on the curve or the surface. The nodes are ordered from 1 to n. A *discrete graph signal* is the vector $v = (v_1,...,v_n)^t$, with the value v_i for the $i:th$ node in the graph. Depending on if the curve or surface is two- or three-dimensional the vertices will have the representation $v_i = (x_i, y_i)^t$ or $v_i = (x_i, y_i, z_i)^t$. The graph signal can later be smoothed using different filters.

A graph *G=(V, E)* represented as set of vertices $V = \{v_i : 1 \le i \le n_v\}$ and a set of edges $E = \{e_k : 1 \le k \le n_e\}$. Each edge has an associated pair of vertices, $e_k = (v_1^k, v_2^k)$.

The *first order neighborhood* to the node i are all nodes $j_1 \dots j_n$ connected to i by an edge (i, j). If the node *j* belongs to the neighborhood i^* it is called a neighbor to node i. A node is not allowed to be a neighbor to it self and if a node j is a neighbor to node i is the node j also a neighbor to node i.

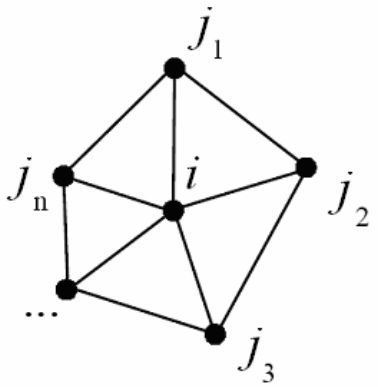


Figure 40: *Figure showing the first order neighborhood to the vertex i.*

3.2.2 Laplacian Smoothing

Most smoothing algorithms smooth the polygonal mesh by moving its vertices without changing the connectivity of the faces or removing and adding vertices to the mesh. This is also the way the iterative Laplacian smoothing algorithm works.

Laplacian smoothing is a well-known technique used for removing geometric irregularities from 2D-meshes. The algorithm works by moving the vertices of the grid to the barycentre (center of mass) of its neighbors. This is done for a

number of iterations, which will lead to that the high frequencies of the treated mesh, will be removed. The Laplacian smoothing algorithm calculates the convolution of the graph signal and the discrete Laplacian operator, which is the weighted sum of the neighbourhood vectors.

The Laplacian operator of a general discrete graph signal is defined as:

(1) $$\Delta v_i = \sum_{j \in i^*} w_{ij} (v_j - v_i)$$

where w_{ij} is the weight for the vector $v_j - v_i$. The weights sum up to 1 for each vertex:

(2) $$\sum_{j \in i^*} w_{ij} = 1$$

There are many ways to choose the weights and a good and simple one is to give all vectors equal weights: $1/|i^*|$. Another strategy could be to use fujiwara weights, which calculate the weights as an inverse of the edge length.

Equation (1) can also be written in matrix form as:

(3) $$\Delta v = -(I - W)v = -Kv,$$

where I is the identity matrix and W is the weight matrix. The $ij:th$ element of the weight matrix is w_{ij} if the node j is a neighbour to node i, and 0 otherwise.

When the convolution of the Laplacian operator and the graph signal of all vertices of the mesh have been calculated the vertices are updated by adding the displacement vector to the original vertex positions. The displacement vector is the product of the vector average and a scale factor $0 < \lambda < 1$:

(4) $$v_i' = v_i + \lambda \Delta v_i$$

or written as a matrix equation:

(5) $$V' = (I - \lambda K)\, V$$

The algorithm functions by carrying out the smoothing step for a number of iterations, each iteration making the surface smoother. The iterative step can be expressed as:

(6) $$V^N = (I - \lambda K)^N\, V$$

A Study of the Laplacian Smoothing

Let's now see how the Laplacian smoothing algorithm from above treats a disturbed sinus signal. We wrote a Matlab program to study the behavior of different smoothing algorithms and to compare them. Consider a closed curve with consecutive vertices v_i and equal weights $w_{ij} = 1/|i^*|$. The first order neighborhood to every vertex will then be $i^* = \{i-1, i+1\}$ and (1) can be written as:

$$\Delta v_i = \frac{1}{2}(v_{i-1} - v_i) + \frac{1}{2}(v_{i+1} - v_i) = \frac{1}{2}(v_{i-1} - 2v_i + v_{i+1})$$

The convolution of the Laplacian operator and the graph signal can be written as: $\begin{bmatrix} 1 & -1/2 & 0 & \dots & 0 & -1/2 \end{bmatrix} * v$, or in matrix form as:

$$\Delta v = -\left[\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & \ddots & \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & 1 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 1 & & \\ 0 & 1 & 0 & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 1 & & & 1 & 0 \end{pmatrix}\right] v = -\frac{1}{2}\begin{pmatrix} 2 & -1 & 0 & \cdots & -1 \\ -1 & 2 & -1 & & \\ 0 & -1 & 2 & \ddots & \\ \vdots & & \ddots & \ddots & -1 \\ -1 & & & -1 & 2 \end{pmatrix} v = -Kv$$

In the following diagrams from the Matlab-simulation we have added noise similar to a ridge to a simple sinus signal. The noise added consists of a sinus signal with the double frequency compared to the original added to an interval of the original sinus signal together with some stochastic noise. The signal has then been filtrated 30 respectively 200 times with the filter coefficient $\lambda = 0.35$. Already after 30 iterations we can see the effectiveness of the Laplacian smoothing algorithm. After 200 iterations the ridge is almost gone, the result is similar to the original signal but not quite the same. The amplitude of the signal has been reduced and it is somewhat distorted. This is a sign of the so-called shrinkage effect which will treat later in this chapter. Another interesting observation is that the double frequency sinus signal is also gone.

Let's now look at the Fourier Transform of the signal. The simple sinus signal consists of a single low frequency. The ridge adds noise to the higher frequencies of the signal which makes it distorted. After 30 iterative steps with the Laplacian smoothing algorithm a lot of the high frequency noise is gone and after 200 steps is the filter even sharper leaving almost only the single low frequency of the sinus signal.

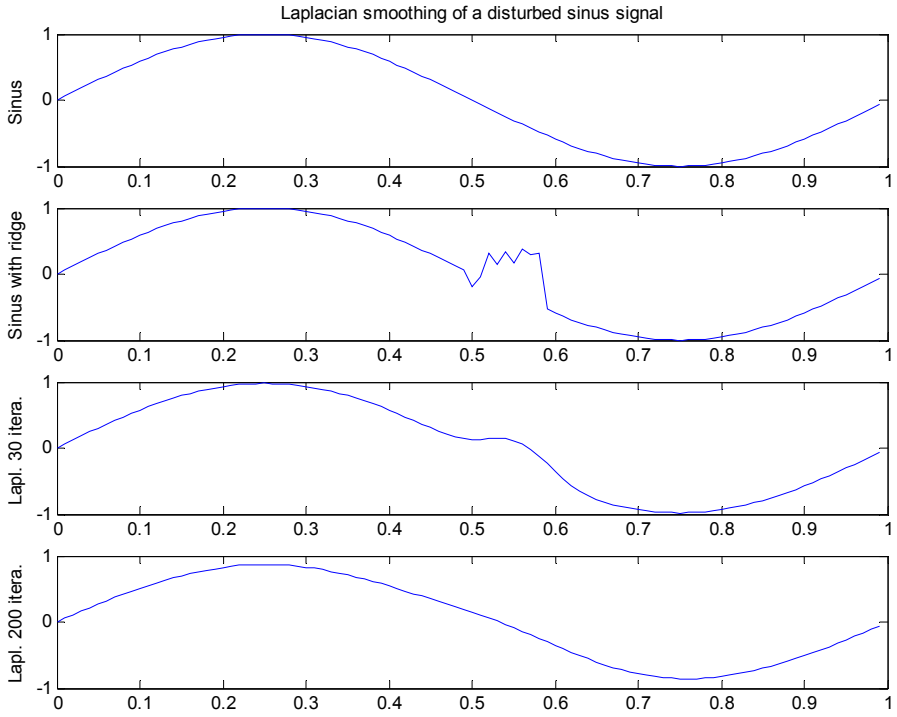


Figure 41a: *The upper diagram shows an undistorted sinus signal. The next shows the sinus signal with an added distortion, lets call it a 'ridge'. In the two lowest diagrams the distorted signal has been smoothed with 30 respectively200 iterations with the Laplacian smoothing algorithm.*

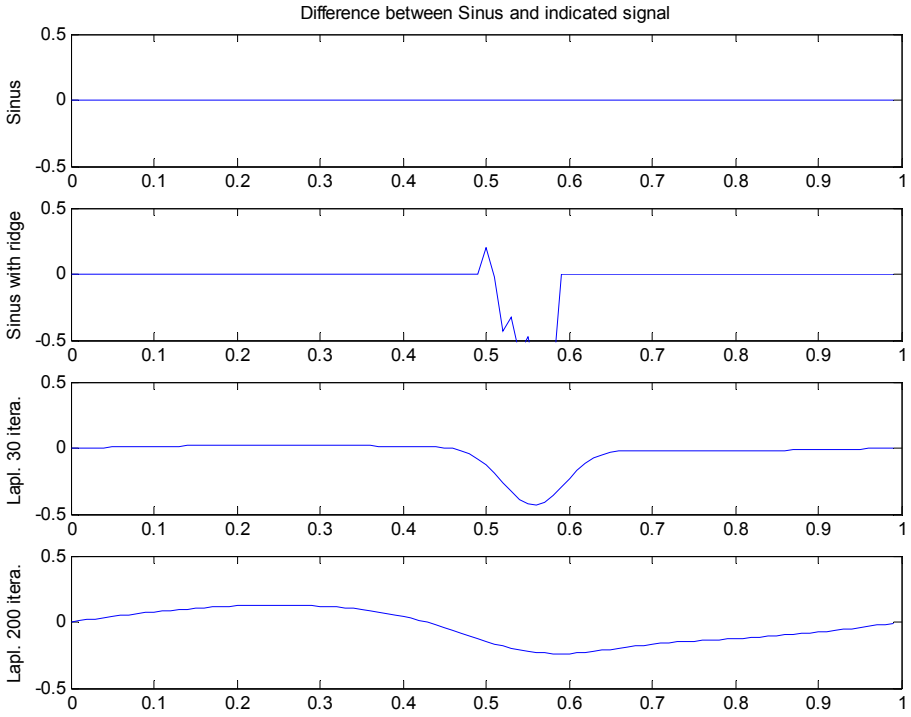


Figure 41b: *This figure corresponds to the previous, but here are the difference between the original signal and the other signals studied.*

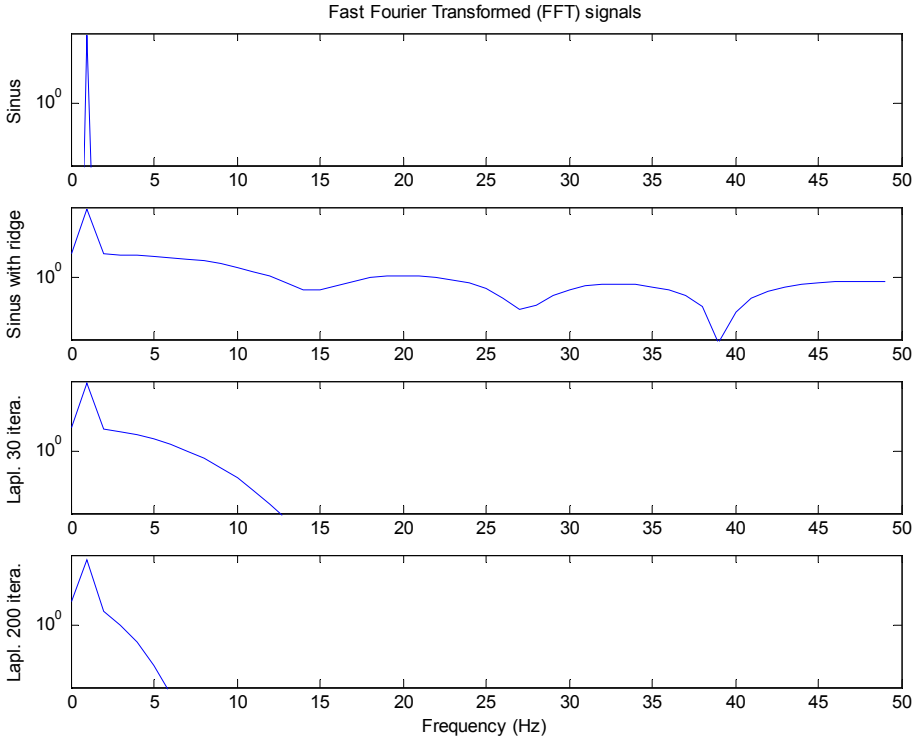


Figure 41c: *This figure shows the Fast Fourier Transform of the studied signals. Note that the scale on the y-axis is a logarithmic one.*

3.2.3 Shrinkage effect

An unwanted result of the Laplacian smoothing is the so-called shrinkage effect. The shrinkage effect is a result of continually moving the vertices in the direction of their statistical average. For a curve this phenomenon will cause the curve to converge into a circle, which then shrinks in size and eventually to a single point, the average centre of mass for all the points on the curve.

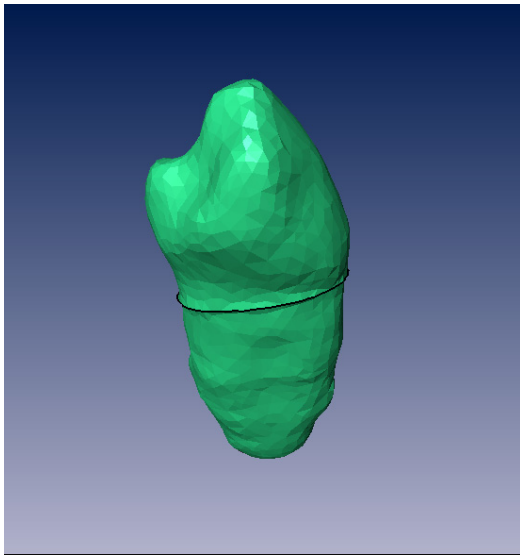


Figure 42a: *Original data. The black curve marks the original location of the ridge.*

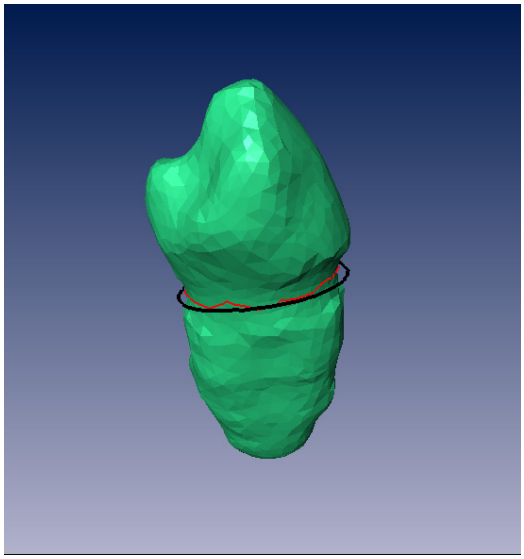


Figure 42b: *Smoothed data with occurrence of shrinkage. The red curve marks the new location of the ridge.*

This phenomenon can also be observed by looking at the transfer function of the Laplacian smoothing, introduced in the next chapter. We will also introduce the transfer function for Taubin-smoothing which has no shrinkage effect.

3.2.4 Signal processing on graph signals

A good way to solve the smoothing problem would be to move our point of view from the space of signals and instead study the frequency subspaces of the signals. We can smooth the graph signal by calculating its DFT and attenuate the high frequencies of the signal. This approach has been suggested by [9] and is called the *Fourier descriptors method*. The low frequencies are considered as the original or 'correct' data and the high frequencies as noise. A perfect filter would set all high frequencies above a certain threshold to 0 while letting frequencies below the threshold pass by unchanged. This kind of filter can't be implemented efficiently but there are fast linear filters that come close to that behavior, among them the Taubin-filter and FIR-filters [6] with different kinds of windows. As we will see later the Laplacian filter does not produce good results.

Taubin makes some valuable observations about the matrix K in his paper [5]. He derives the transfer function for algorithms using first order neighbourhoods with equal weights and he makes the observation that the weight matrix W is a normal matrix which has real eigenvalues. By its construction W is also a stochastic matrix with eigenvalues bounded between -1 and 1. As a consequence the eigenvalues of the matrix $K = I-W$ are also real and bounded between 0 and 2, where the high eigenvalues correspond to the high frequencies of the graph signal. The eigenvalues of K have a close relationship to the filtering of the graph signal if the signal is written in the basis of the eigenvectors of K. To attenuate the high eigenvalues of the matrix K is the same as to attenuate the high frequencies of the signal [5].

If a set of right eigenvectors $e_i,\ldots,e_N$ that corresponds to the eigenvalues $0 \le k_1 \le \ldots \le k_N \le 2$ of the matrix K and $\delta_1,\ldots,\delta_N$ are the associated dual basis of the right eigenvectors $e_i,\ldots,e_N$ is chosen it is possible to write the discrete graph signal v as a linear combination of the eigenvectors $e_i,\ldots,e_N$ as

$$v = \sum_{i=1}^{N} \hat{v}_i e_i = E\hat{v} \tag{7}$$

where $\hat{v}_i = \delta_i^t v$ is the DFT of the graph signal. The filtering of the graph signal v with the transfer function $f(K)$ can then be written as

$$v' = f(K)v = f(K)E\hat{v} = diag(f(k))E\hat{v} = \sum_{i=1}^{N} f(k_i)\hat{v}_i e_i = \left(\sum_{i=1}^{N} f(k_i)e_i\delta_i^t\right)v \tag{8}$$

or iteratively in matrix form as

$$V^N = f(K)^N V \tag{9}$$

The eigenvalues of the matrix K can be calculated with $diag(k) = E^{-1}KE$, where E are the right eigenvectors of K. The right eigenvectors of K; $e_i,\ldots,e_N$ are also the right eigenvalues of the smoothing matrix $f(K)$ and the eigenvalues can be found with linear algebra once the eigenvalues of K are known. Ideally one would like to find a transfer function that is $f(k) = 1$ for the frequencies lower than a desired pass-band frequency $0 \le k \le k_{PB}$ and $f(k) = 0$ for $k_{PB} < k \le 2$. Calculating the eigenvalues of K this is however impossible in the most cases. The computations of eigenvalues of K and DFT of the graph signal are not linear in time complexity which makes them practically

unusable for larger data sets. Instead Taubin suggested a low-pass filter $f(K)$ that simulates the desired behavior.

If we go back to the Laplacian smoothing matrix from equation (6) $f(K)^N = (1-\lambda K)^N$ we can see that it does not fulfill the desired criteria's:

$$\lim_{N\to\infty}(1-\lambda k)^N = \begin{cases} 1 & for\ k = 0 \\ 0 & for\ 0 < k \le 2. \end{cases}$$

This observation was done by Taubin and he introduces another transfer function that fulfils the desired criteria's.

3.2.5 Taubin-filter and filter design

The improved smoothing filter of Taubin is based on the Laplacian smoothing, but doesn't generate shrinkage. If we go back to the Laplacian smoothing equation in (6) we can see that it uses a factor λ to move the vertices into the barycentre of the neighbors. Taubin [5] introduces another factor μ to move the points back again. The filtering could be seen as continually moving the vertices back and forth for a number of iterations. The filtering is expressed as a two-step iterative equation:

$$(10) \qquad \begin{cases} v_i' = v_i^n + \lambda \Delta v_i^n \\ v_i^{n+1} = v_i' + \mu \Delta v_i' \end{cases}$$

or in matrix form as

$$(11) \qquad V^N = \left((I-\lambda K)(I-\mu K)\right)^N V$$

The factor μ is larger than λ and negative: $0 < \lambda < -\mu$. In order to study the differences between the Laplacian filter and the Taubin filter we implemented a Matlab program that simulates the two filters.

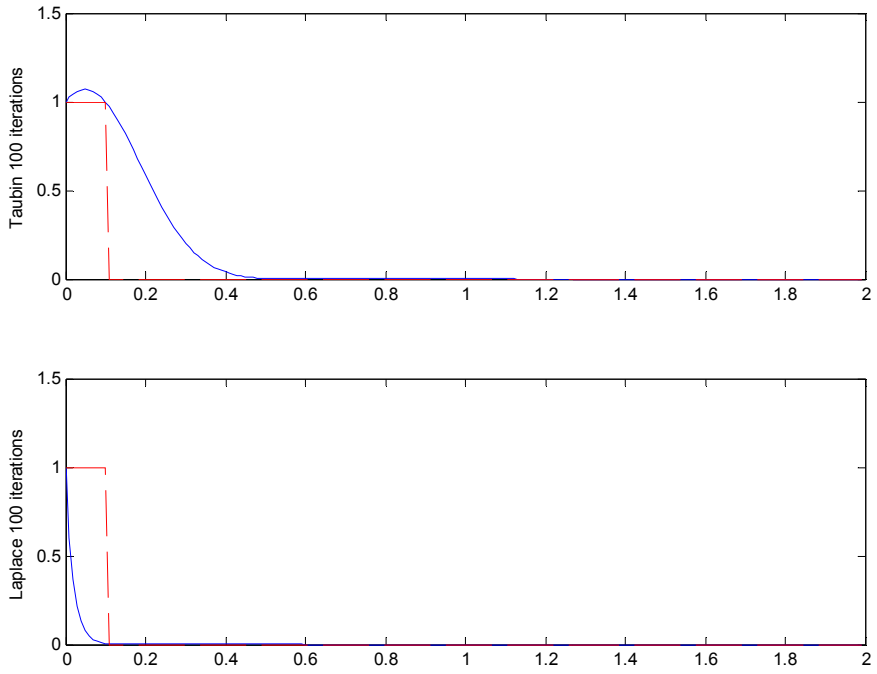


Figure 43: *Diagrams showing the transfer functions of the algorithm invented by Taubin (Taubin-filter) and the Laplacian smoothing algorithm. In this example the Taubin-filter is designed to have a pass band frequency of* $k_{PB} = 0.1$

The Laplacian transfer function lets only the zero frequency pass unchanged, while the Taubin-filter transfer function lets more frequencies pass. Frequencies up to the pass band frequency are slightly enhanced and frequencies over the pass band frequency are attenuated.

Designing a Taubin-filter with suitable parameters can be done fairly easily. What we want to do is to let the pass band frequency pass unchanged: $f(k_{PB}) = (1 - k_{PB}\lambda)(1 - k_{PB}\mu) = 1$. This second degree equation has the solutions;

(12) $\quad k_{PB} = 0$ and $k_{PB} = \frac{1}{\lambda} + \frac{1}{\mu}$

We must also choose the parameters so that we get a stable transfer function. The rule of BIBO, bounded input – bounded output applies, which gives that the absolute value of the transfer function has to be less than 1: $|f(k)| < 1$. This is given by the fact that $f(k)^N$ will increase towards eternity if $|f(k)| > 1$. Since the function $f(k)$ is decreasing for values above the pass band frequency we get the following conditions that have to hold for a stable and fast transfer function:

$$\begin{cases} f(2) = (1-2\lambda)(1-2\mu) \overset{[12]}{=} (1-2\lambda)(1-\dfrac{2}{k_{PB}-1/\lambda}) > -1 \\ f(k_{PB}) = (1-k_{PB}\lambda)(1-k_{PB}\mu) \overset{[12]}{=} (1-k_{PB}\lambda)(1-\dfrac{k_{PB}}{k_{PB}-1/\lambda}) = 1 \end{cases}$$

Solving this equation system gives:

$$(13) \qquad \begin{cases} \lambda < \dfrac{-k_{PB}}{2(2-k_{PB})} + \dfrac{1}{2(2-k_{PB})}\sqrt{k_{PB}^2(2-k_{PB})^2+4} \\ \mu > \dfrac{1}{k_{PB}-1/\lambda} \end{cases}$$

The pass band frequency should be chosen so that the original frequencies of the signal are kept and the noise is removed. A too low pass band frequency will generate shrinkage and a too high will not remove the noise. A useful property is Parseval´s theorem, which states that the sum of the squares of a signal is equal to the sum of the square of its DFT:

$$(14) \qquad \|x\|^2 = \frac{1}{N}\|X\|^2$$

So if we sum the frequency subspaces of the DFT from 0 up till the pass band frequency we get the energy preserved by the filter. The pass band frequency should be chosen so that the most energy of the 'correct' part of the signal is kept. To compute energy of the signal with DFT is, once again, in most cases not possible, but could be done with some kind of power spectrum estimation like in [6]. A typical good value of k_{PB} is 0.1 which gives $\lambda = 0.5024$ and $\mu = -0.5289$.

The Taubin-algorithm gives good results, but a large number of iterations are normally necessary to remove bigger distortions like heavily pronounced ridges.

3.3 Implementation

To smooth the ridge we have developed a small toolbox of algorithms. They have different advantages and are useful in different cases.

3.3.1 Selection of vertices

We are not interested in smoothing the whole surface but rather a smaller part of it around the ridges of the boundary curve. The set of point collected for the smoothing are the vertices lying on the boundary curve and all the surrounding vertices to certain distance from each vertex on the curve. These are collected in a recursive way by following the connecting edges between the vertices. Each vertex is visited several times but is only added once to the set of vertices to be smoothed.

We found it valuable to divide the smoothing into 2 different categories:

1. Boundary algorithm - smoothing of the points lying on the boundary.
2. Surrounding algorithm - smoothing of the points surrounding the boundary to a certain depth.

This allows us to put special constrains on the smoothing of the vertices on the boundary. This is used by the algorithm “Following ridge and normal vector”.

3.3.2 Taubin filter

This algorithm is the implementation of described Taubin-filter from above. It uses the first order neighborhood structure to the vertices to compute the Laplacian operator. We have selected equal weights for the Laplacian operator as this has proven to be a good enough selection for our needs. The smoothing is carried out on the set of all vertices not separating between the ones lying on the boundary curve.

3.3.3 Following ridge and normal vector

This algorithm uses a first order neighborhood structure with equal weights just as the Taubin-filter. The movement of the vertices on the boundary curve is however constrained, only allowing the vertices to move in the direction of two vectors; the normal vector of the plane fitted to the first order neighborhood and the local approximation for the boundary vector at this vertex. The motivation for using this constrains is that it keeps the boundary form and in some cases treats the other materials apart from the special material better than the Taubin-filter (see figure 44). The displacement vector is calculated as follows:

(15)
$$\bar{d} = (\Delta v \cdot \bar{n})\bar{n} + (\Delta v \cdot \bar{r})\bar{r}$$
$$\bar{r} = ridge\ vector$$
$$\bar{n} = normal\ vector$$
$$\Delta v = laplacian\ operator\ \ from\ (1)$$

To avoid the shrinkage effect the algorithm 'moves back' the smoothed points towards their original position a factor $0 \le \beta \le 1$ in the direction of the normal vector:

(16)
$$\bar{d}' = -\beta(\Delta v \cdot \bar{n})\bar{n}$$

This gives the following update expression for the vertices:

(17)
$$v_i^{'} = v_i + \bar{d} + \bar{d}' = v_i + (1-\beta)(\Delta v_i \cdot \bar{n})\bar{n} + (\Delta v_i \cdot \bar{r})\bar{r}$$

The algorithm is combined with Taubin-filter as surrounding algorithm. A way of designing the value for parameter β is not specified in this paper.

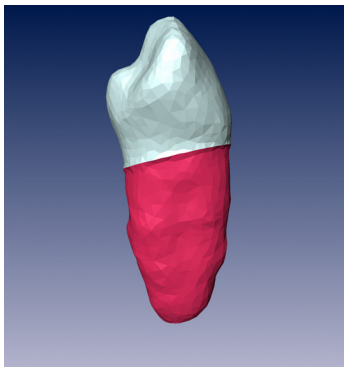


Figure 44a: *Original ridge on surface of the tooth shown in twisted view. White marked patches are tooth/air and red marked patches are tooth/gum.*

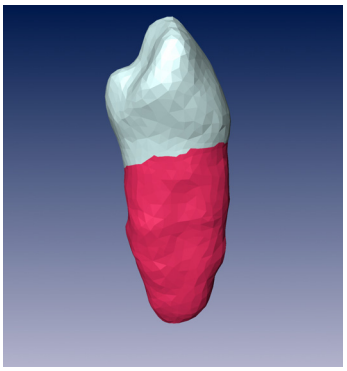


Figure 44b: *The ridge was smoothed 10 iterations, depth 3 with the Taubin-algorithm as boundary algorithm. Irregularities from the originally straight boundary form can be seen.*

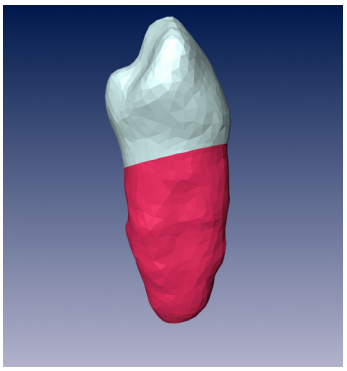


Figure 44c: *The ridge was smoothed 10 iterations, depth 3 with the boundary algorithm following normal and ridge vector. The straight boundary form is kept.*

3.3.4 Neighbors on boundary

This algorithm is used for obtaining a smoother boundary between the special material and the neighboring materials. The filter uses the first order neighbors lying on the boundary curve: The filter actually uses the same Laplacian operator as in the closed curve example in the section about the Laplacian smoothing algorithm. The Laplacian operator uses equal weights. It is not combined with any surrounding algorithm.

Figure 45a: *A smoothed surface with boundary.*

Figure 45b: *The boundary is straightened out with 10 iterations with the boundary neighbors Taubin-filter.*

3.4 Results for the Taubin-filter

The results in this section refer to the results obtained with the implemented smoothing module from the post-processing step. The results in the first section are shown in 3 different views: visual, surface distance and the surface curvature. The visual view is the results as they are shown to the user of the module. The surface distance shows how much the smoothed surface differers from the ideal. The surface distance is a color marked indicator which spans from white (no difference [0]) through blue and red to yellow (maximal distance [0.0214]).

Figure 46: *The color scale of the surface distance*

The last view is the curvature view which shows the shifting in the surface. This color indicator also spans from white (flat local surface [0]) through blue and red to yellow (edge [1]).

3.4.1 Visual results

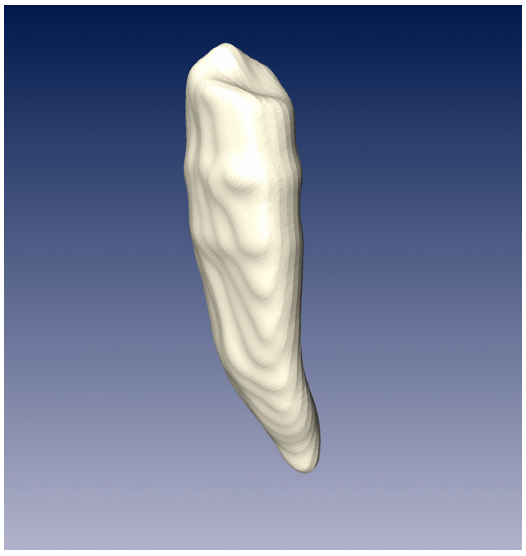


Figure 47a: *Dentin visualized without any other materials present. The other materials have been removed from the dataset in order to prevent occurrences of ridges. This is the ideal surface.*

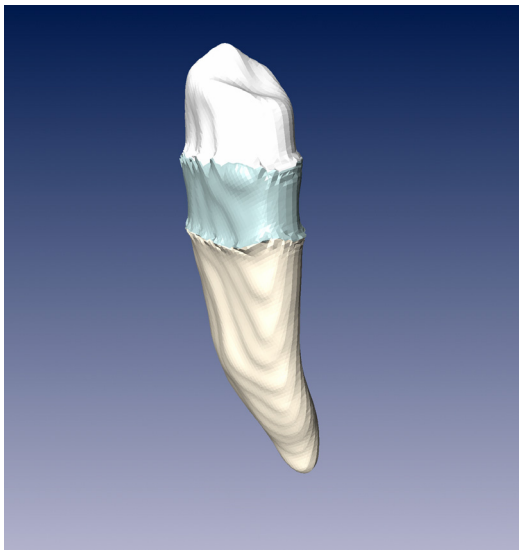


Figure 47b: *Dentin visualized together with other materials. Ridges occur where 3 different materials meet. The surface is shown in twisted mode where the different colors indicate the different patches.*

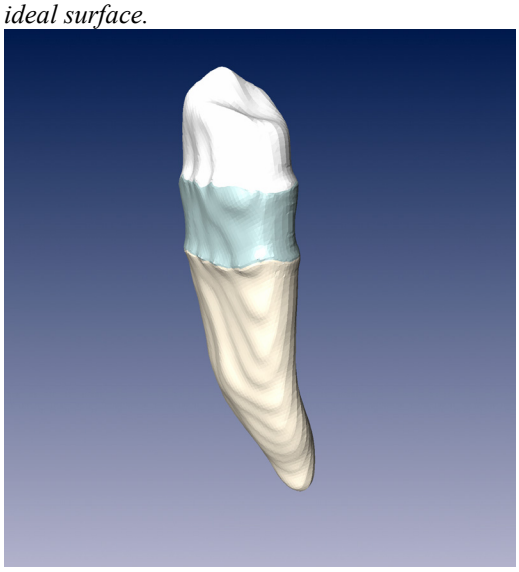


Figure 47c: The surface from figure 47 b smoothed with the Taubin-filter: 15 iterations, depth 10 and pass band frequency $k_{PB} = 0.1$.

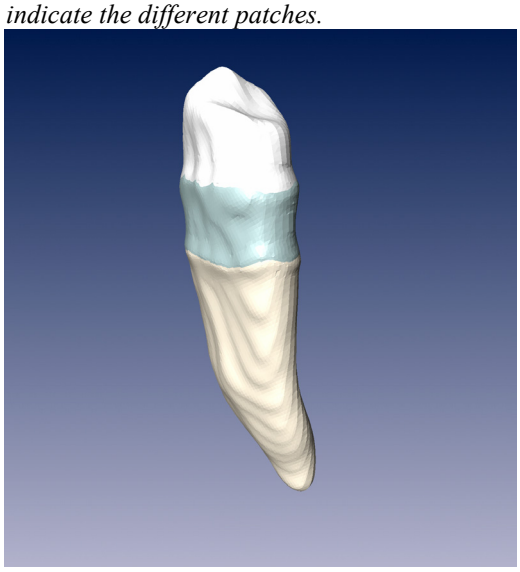


Figure 47d: The surface from figure 47 b smoothed with the Taubin-filter: 100 iterations, depth 10 and pass band frequency $k_{PB} = 0.1$.

Figure 47e: *The surface from figure 57 b smoothed with the Taubin-filter: 100 iterations, depth 3 and pass band frequency* $k_{PB} = 0.1$ *plus 5 iterations, depth 10 and pass band frequency* $k_{PB} = 0.1$.

Already after 15 iterations the surface looks considerably smoother. After 100 iterations the ridge has shrunken even more but the remains are still visible in the model. This is how far we come as removing the ridge in this case. The low frequency component of the broad ridge can't be removed without distorting the rest of the underlying low frequency information of the surface. We have found that the ridge can be totally removed in models where it is not so strongly pronounced as in this case.

3.4.2 Surface distance

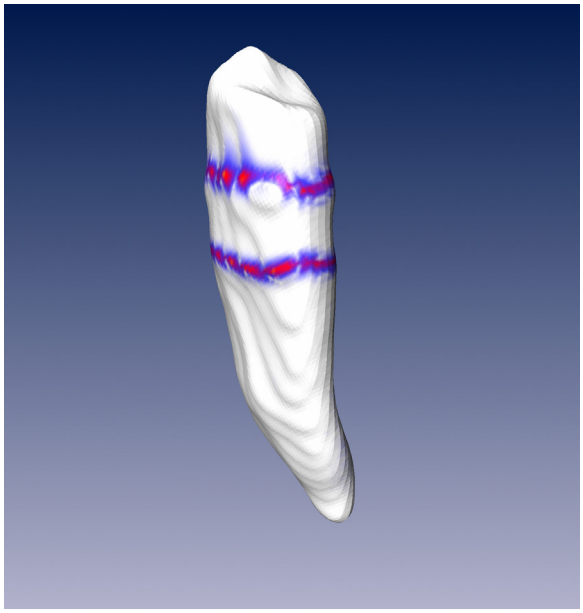


Figure 48a: *Surface distance between the ideal surface and the surface with ridge. The distance field is displayed on the ideal surface with a color marker on the triangles. The intense red and yellow parts show where the ridge is as biggest. The blue parts show mild differences between the surfaces. Some statistical values for the surface distance comparison:*

Mean, $\overline{\delta} = 0.00142138$
Deviation, $\sigma = 0.00346552$
$rms = 0.00374555$
Maximal distance, $\overline{\delta}_{\max} = 0.021400$

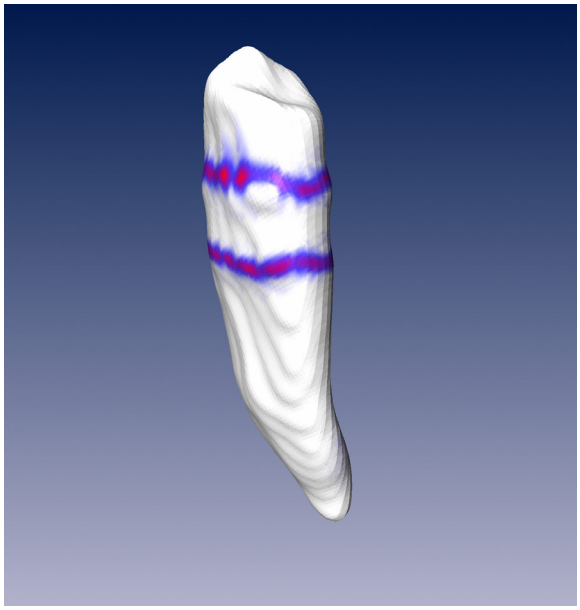


Figure 48b: *Surface distance between the ideal surface and the ridge smoothed 15 iterations, depth 10 with the Taubin-filter. The intense red parts of the ridge are gone and replaced with milder nuances of red and blue. This indicates that the ridge has shrunken. If we look at the statistics we can see that the maximal distance has sunken from 0.21400 to 0.018837:*

Mean, $\overline{\delta} = 0.00150376$
Deviation, $\sigma = 0.00343285$
$rms = 0.00374764$
Maximal distance, $\overline{\delta}_{\max} = 0.018837$

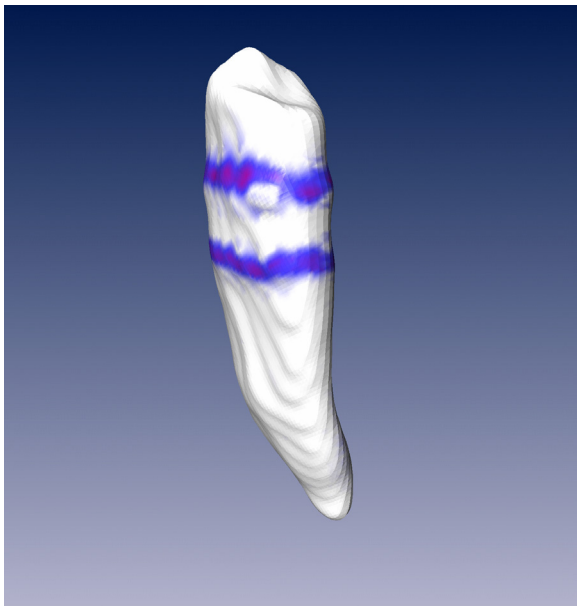


Figure 48c: *Surface distance between the ideal surface and the ridge smoothed 100 iterations, depth 10 with the Taubin-filter. The red parts are almost gone. The ridge has shrunken further. The mean distance is higher than for the original ridge as a result of the big depth, which is wider than the ridge. This makes the surface look smoother but also moves correctly placed vertices not belonging to the ridge.*

Mean, $\bar{\delta} = 0.00153481$
Deviation, $\sigma = 0.00322405$
$rms = 0.00357061$
Maximal distance, $\bar{\delta}_{\max} = 0.018835$

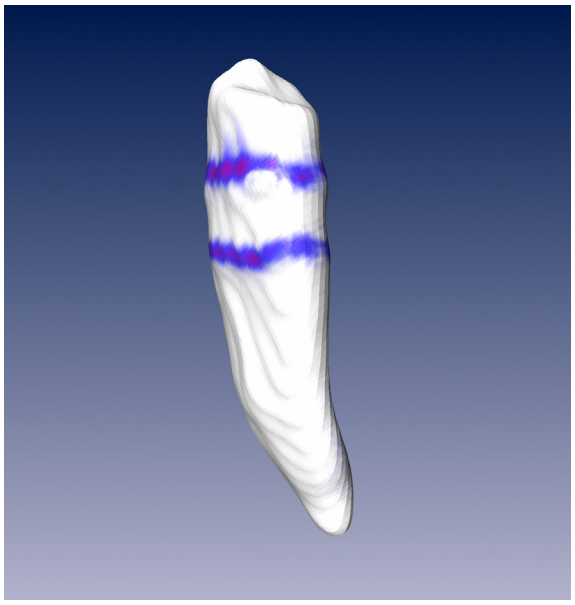


Figure 48d: *Surface distance between the ideal surface and the ridge smoothed 100 iterations with the surrounding depth 3 and additionally 5 iterations with the surrounding depth 10 with the Taubin-filter. This approach generates even better results than 58 c at 1/3 of the computational cost, but the user has to estimate the width of the ridge. The surrounding of the ridge is less modified. The mean distance is smaller and so are the deviation, root mean square and maximal distance as well.*

Mean, $\bar{\delta} = 0.00130501$
Deviation, $\sigma = 0.00299228$
$rms = 0.00326436$
Maximal distance, $\bar{\delta}_{\max} = 0.0170987$

As anticipated a high number of iterations improve the smoothing of the surface, but an interesting result is that an equally good or better result can be achieved if we choose the depth around the boundary properly from the beginning and finish off by slightly smoothing with a larger depth.

3.4.3 Curvature

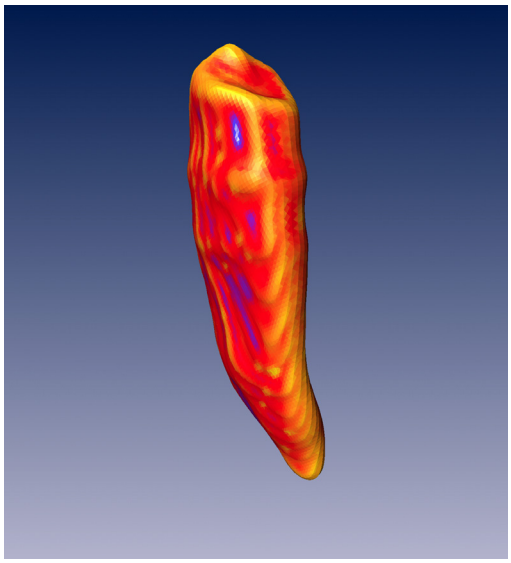


Figure 49a: *Max curvature of the ideal surface. The shiftings on the surface are seen as yellow marked. Note that some of the yellow marks are seen where the original 2D-slices lie.*

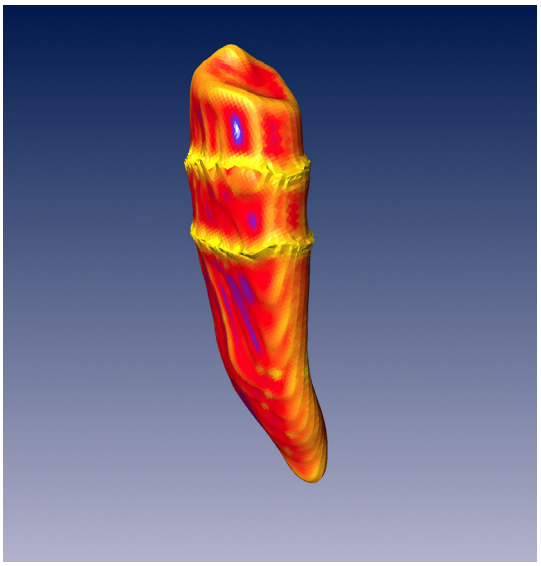


Figure 49b: *Max curvature of the surface with ridge. The big shifting on the ridge is seen as the yellow mark alongside the ridge.*

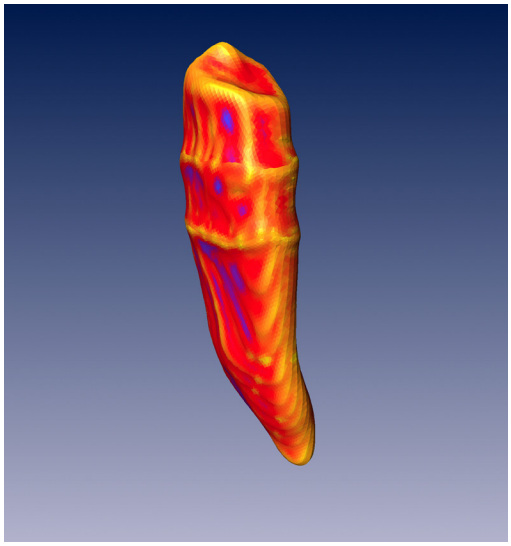


Figure 49c: *Max curvature of the ridge smoothed 100 iterations, depth 10. It is still possible to see where the ridge was even though it is less pronounced.*

3.4.4 Advantages

One of the advantages with the post-processing is that it can be used with an existing version of Amira without modifying other existing modules. In this way the post-processing module differs from the other approach as the module is merely an addition to the existing package. The smoothing module can remove ridges from any material simply by connecting a module to a surface. Another advantage is that the smoothing can be done consecutively for many special materials in a data-set, excluding the option when many special materials border to each other. For example this could be useful in a dataset containing several teeth connected to jaw, gum and so on. The teeth can be smoothed separately as long as they do not border to each other. The smoothing can be done fairly fast and produces good results if the ridge isn't too pronounced.

Compared to traditional surface smoothing strategies which consider a whole surface our solution locates exactly the problem areas, namely alongside the patch boundaries. The rest of the polygonal mesh is left unchanged, leaving the details in the areas that are not infected by ridges. Our solution also reduces the computational needs or redirects the computational powers to sharpen the Taubin-filter with even more iterations in the areas where needed.

3.4.5 Drawbacks

A disadvantage is that the post-processing smoothing is not a fully automated solution, it demands some user interaction. Even though identification of the materials and the boundary between them is done automatically the user has to select a material to be smooth and judge from the looks of the ridge how much smoothing that is necessary. It may take a while to learn to set the parameters properly. Another disadvantage is that ridges can be too big to remove entirely. This happens when the ridge has pronounced underlying low frequency component.

3.5 Further work

Some other approaches could be tried like implementing a sharp and fast FIR-filter. An attempt was made in the end of the project, and the filter seemed do the calculations fast, but we could not get it to function properly. A functioning FIR-filter could be implemented from the description in [6] with some modifications.

Another good feature would be to introduce a possibility to smooth many materials instead of just one. This would allow the user to view many different materials as smooth at the same time. This is for example useful for data set 2, kieferstuck in the following section which consists of several teeth that are labeled to belong to different materials. One ambiguity would arise when 2 special materials border to each other and some kind of strategy would have to be developed to solve this situation.

4 Comparisons

The purpose of this section is to compare the results obtained by using the new GMC and the post processing module with the original surfaces generated when just using the original GMC. The comparisons are purely visual, meaning that no statistical data is extracted of the deviation of the surfaces or such. Comparisons are made using three different data sets, with one material selected in each and one of them that is to be specially treated to avoid ridges. Each data set is compared in the same manner using five snapshot images named a to e. The purpose of the different images:

a) *Presentation of the data set.* The whole data set is visualized with all materials present. This is done with the original GMC, giving an overview of the data set and an idea of what the special material is and how this special material is placed in the model.

b) *The ideal result.* The result obtained when running the original GMC on the labelfield with just the special material present. This result is obtained by deleting all the materials from the labelfield but the selected one. The reason for using this result as an ideal reference is that there are just two materials present, the special one and the exterior. This means that the ridge problem can't occur, since no points exist where three materials meet.

c) *The original GMC.* This is what the surface of the selected material looks like when the original GMC is run on the labelfield with all materials present. Just the special material is shown in the picture.

d) *The new GMC.* The result obtained when the new GMC is run on the labelfield containing all materials and the special material. Comparing the results with the images in b and c one can see how the result from the new GMC differs from the old ones. To make the comparison easier, just the selected material is shown.

e) *The result from post processing.* The result obtained after running the post processing module on the surface obtained by the original GMC. Comparing this result with b gives a good idea of how the post processing works in general. Also here, the special material is the only one shown.

All data sets have been obtained using a MR scanner. The image slices from the scanner have later been segmented and labeled in order to create the labelfields. It is important to point out that with the access to more powerful equipment the detail level of the models can be significantly improved using more images and images of a higher resolution.

4.1 Data set 1, zahn mit kortikalis

This data set contains a tooth with its separate parts and also a piece of jaw bone where the tooth resides. The model consists of four materials; Enamel, dentin, kortikalis and the exterior. The labelfield is of the size 143x239x24 and the resulting surface contains 57486 points and 115350 faces in 5 patches. The special material is dentin and it has two ridges. One of the ridges occur when the exterior, enamel and the dentin meet and the other when kortikalis, dentin and the exterior meet.

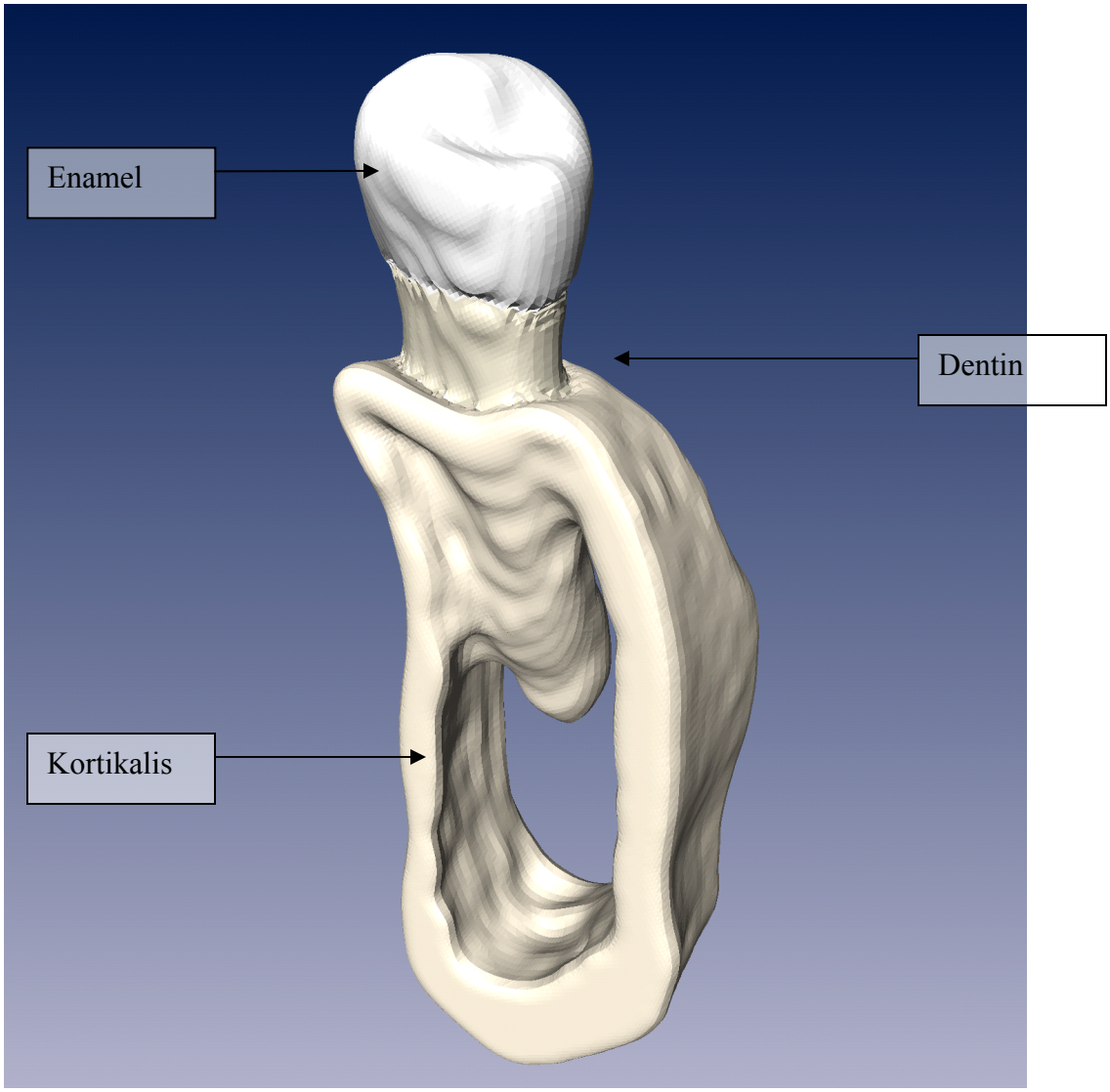


Figure 50a: *Full view of the model with all materials present.*

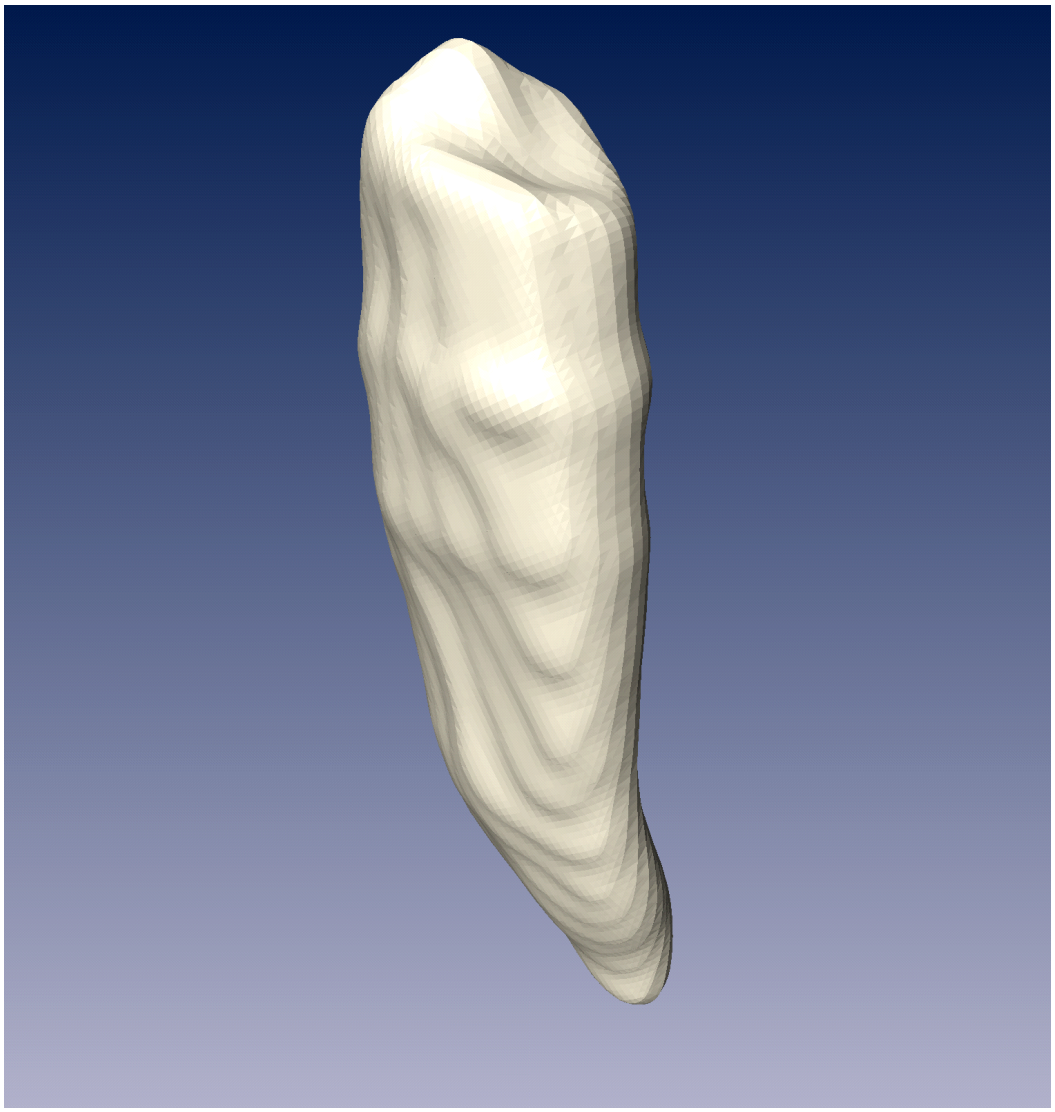


Figure 50b: *This is the ideal result for dentin, the one we are striving for. It is obtained by deleting all the materials from the labelfield but dentin and then generating the surface with the GMC. Since just two materials are present (dentin and exterior), no distortions occur in neither the smoothing process nor the triangulation process.*

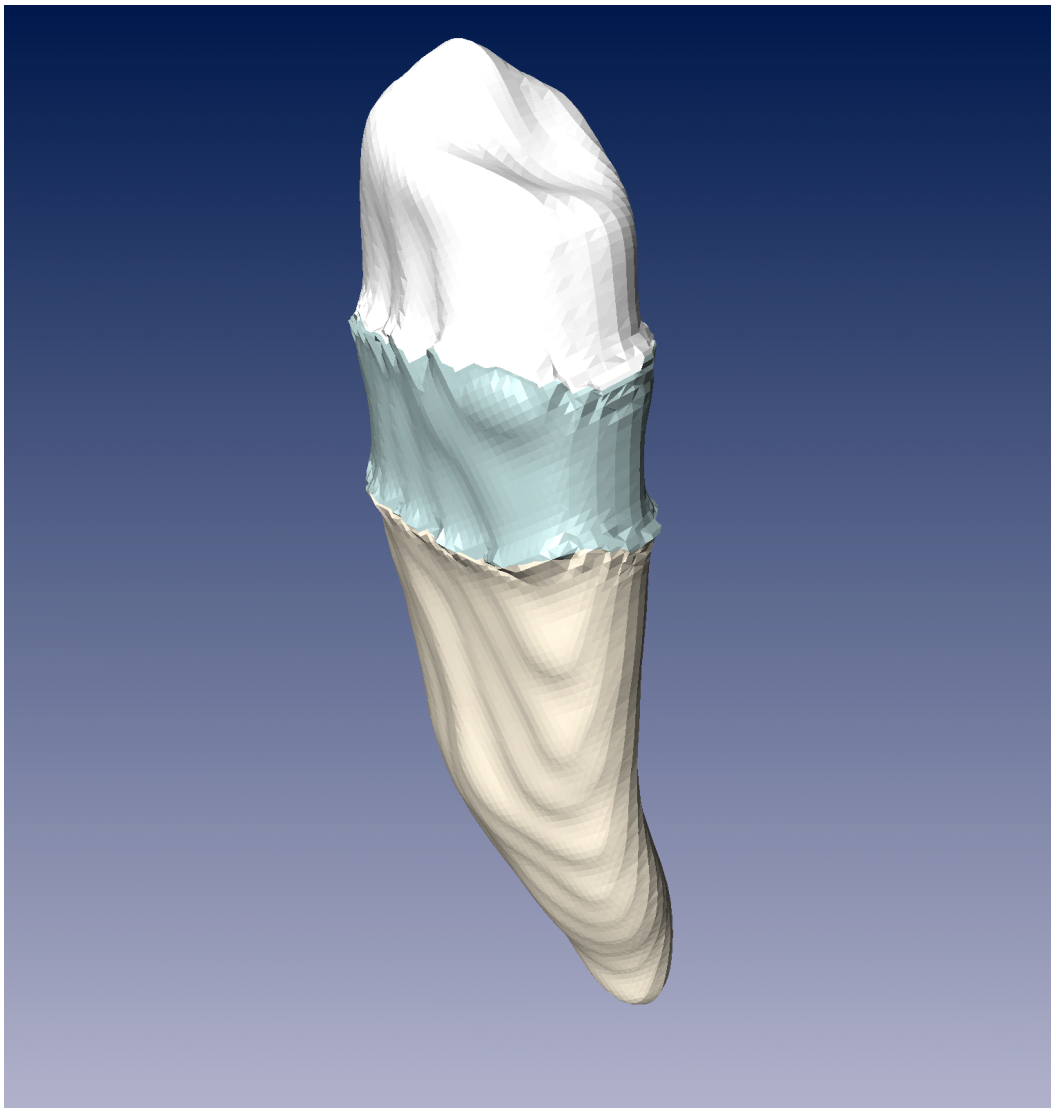


Figure 50c: *This is what dentin looks like when its surface is generated together with the other materials using the original GMC. The snapshot is taken in "twisted view" meaning that all the patches have been colored separately. That way the boundaries between them are easy to distinguish and it's exactly at the boundaries that the ridges will occur. The ridge distortions are as seen quite strong in this model and they present a serious deviation from the ideal result.*

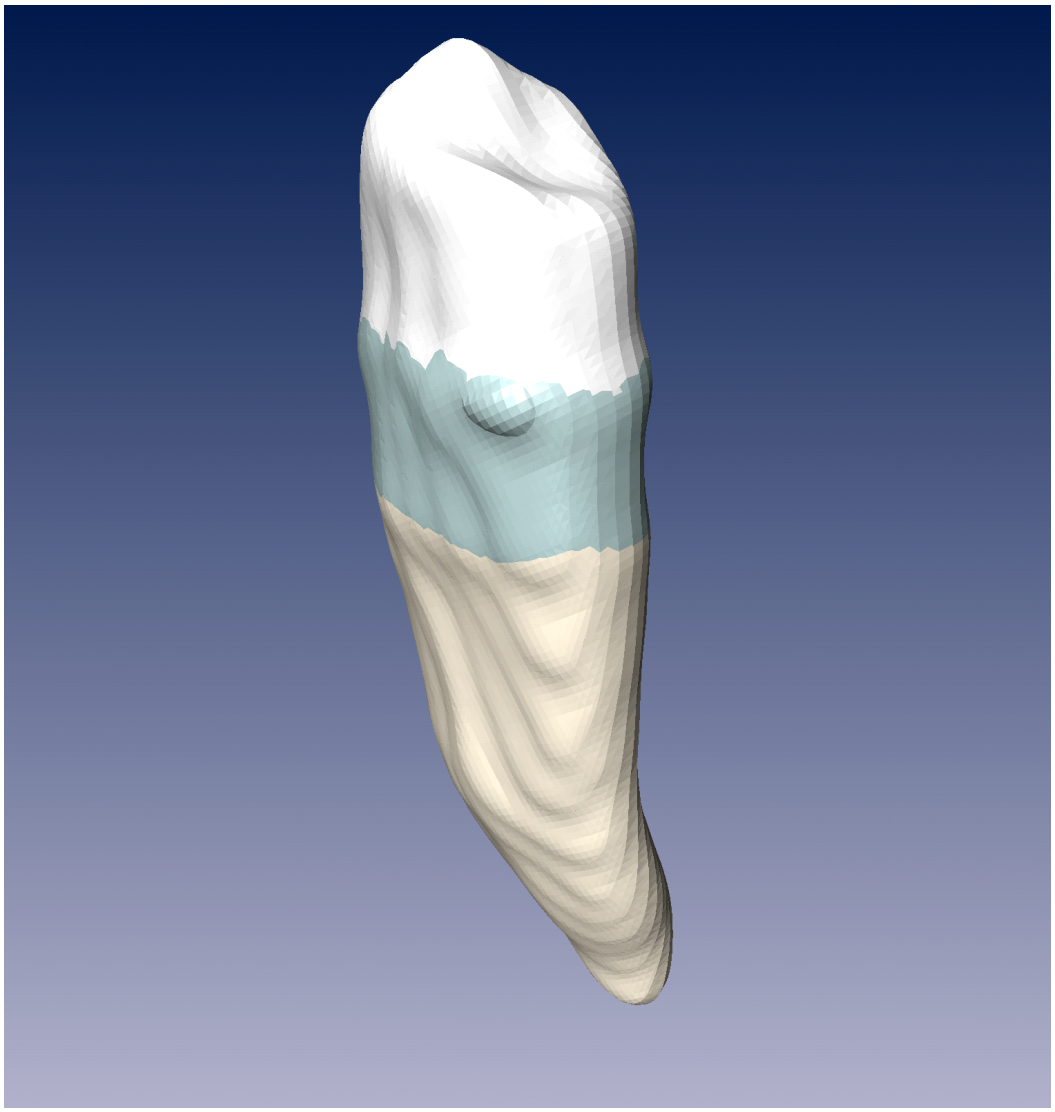


Figure 50d: *The result obtained from the new GMC module run on the complete labelfield with all materials present and dentin selected to be smooth, visualized in "twisted view". The ridges seen in the previous image are gone.*

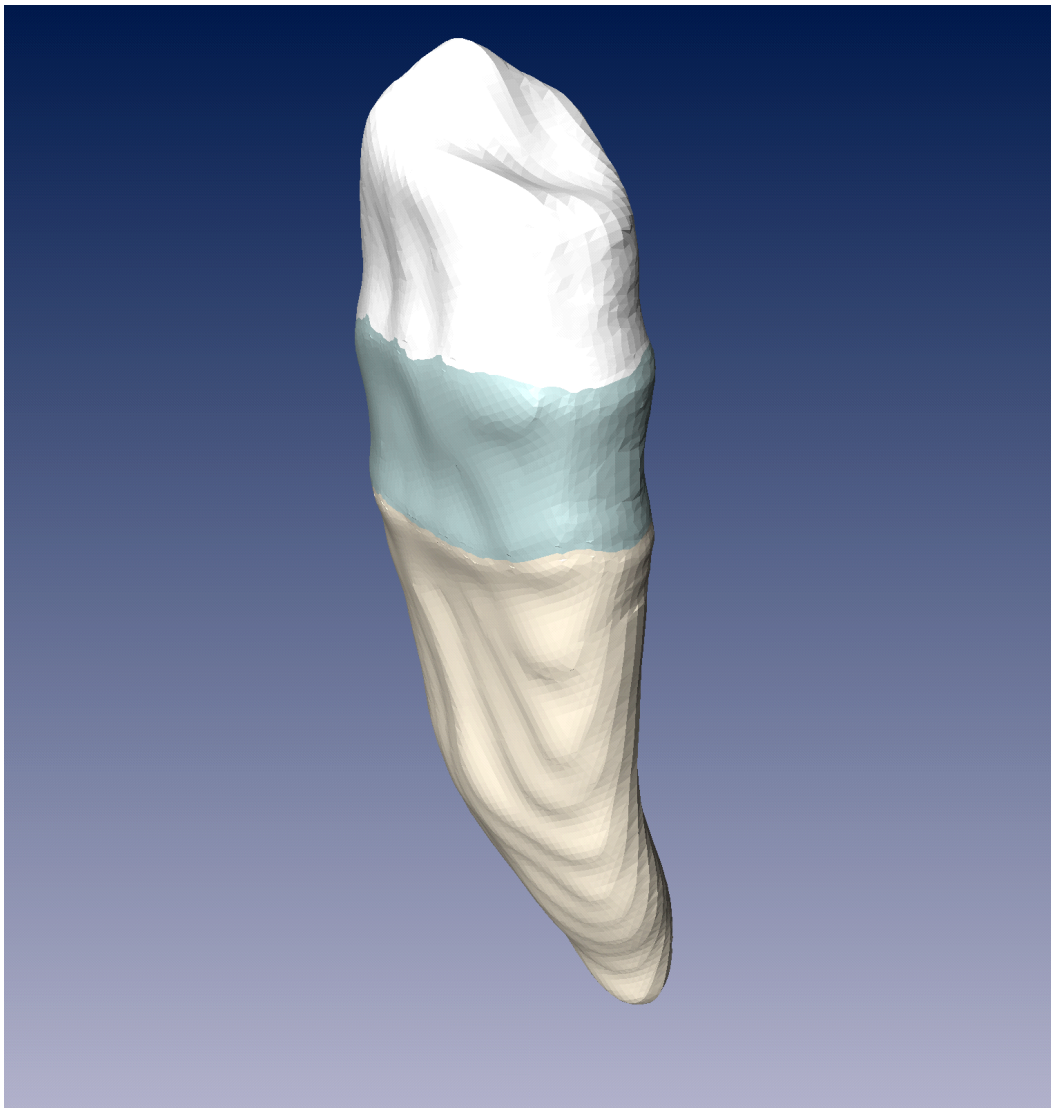


Figure 50e: *The snapshot is taken after running the post processing module on the surface shown in figure 1c. The surface smoothing was done using the Taubin method with 100 iterations and the neighborhood depth set to 20. A better result might be obtained by using different smoothing algorithms after each other and varying the parameters of the filter, neighborhood depth and so on.*

4.2 Data set 2, kieferstuck

This data set contains a piece of a jaw with the teeth belonging to it. It contains eleven materials; Kortikalis, spong, zahn-1 to 8 and the exterior. The labelfield is of the size 267x471x150 and the resulting surface contains 565168 points and 1132568 faces in 32 patches and the material selected to be smooth is zahn-4. There exist several ridges in this model but in order to make it easier to compare, just the ones on zahn-4 are treated. The ridges on zahn-4 occur when the exterior, kortikalis and zahn-4 meet and when the exterior, spong and zahn-4 meet.

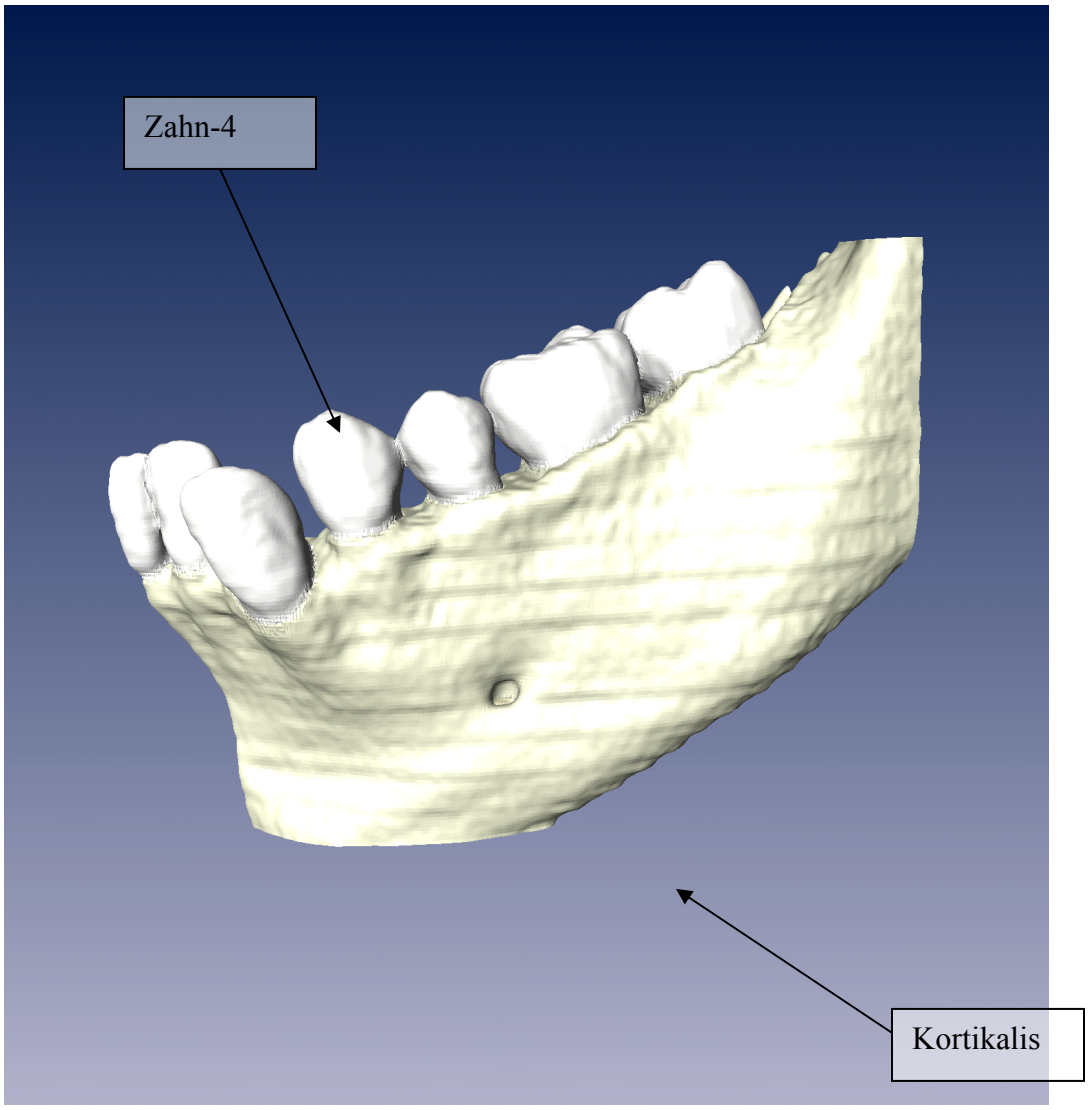


Figure 51a: *All materials are visualized in this snapshot. All the materials are not visible, for example spong is located inside the kortikalis.*

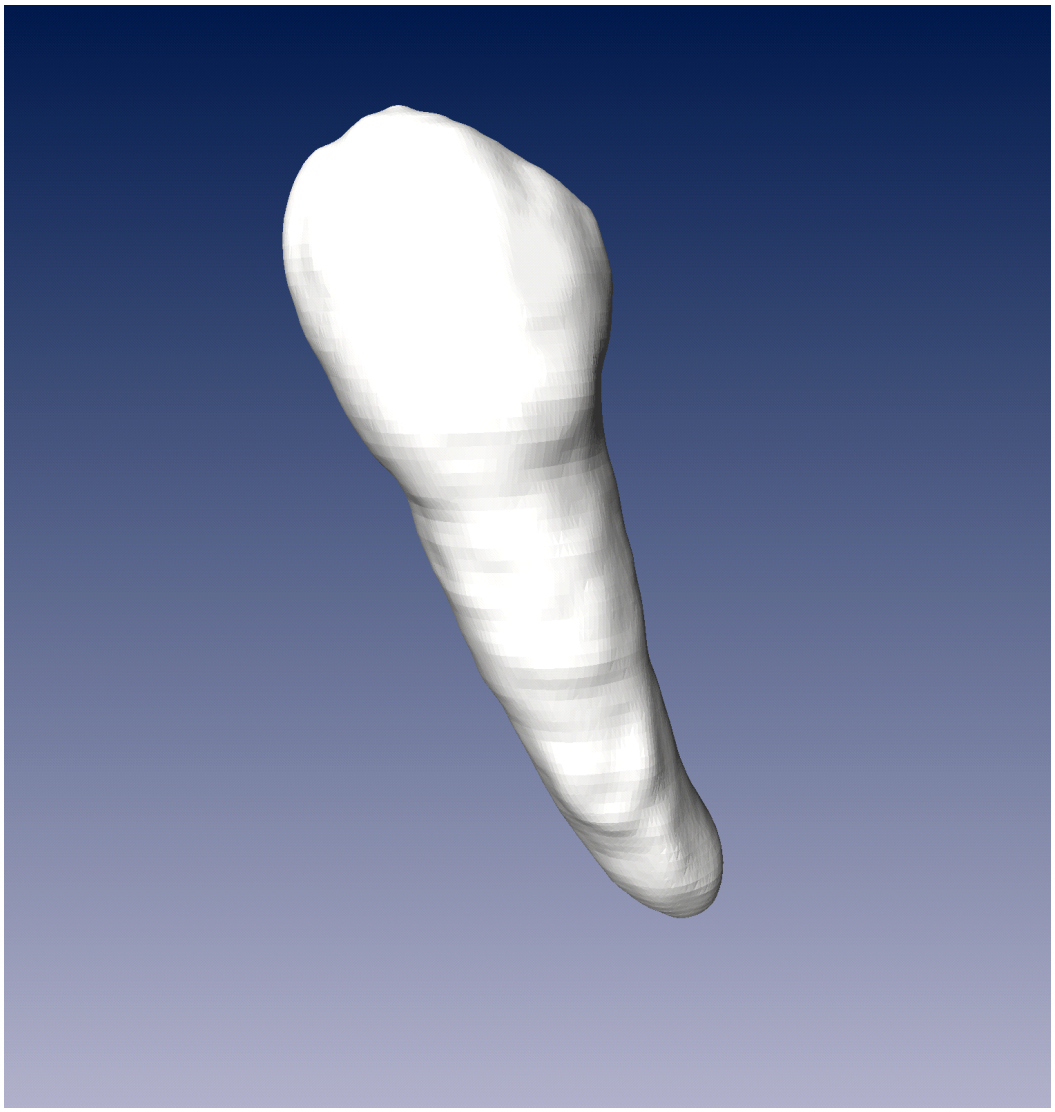


Figure 51b: *The ideal result for zahn-4. Obtained by deleting all the materials from the labelfield but zahn-4 and then generating the surface with the original GMC.*

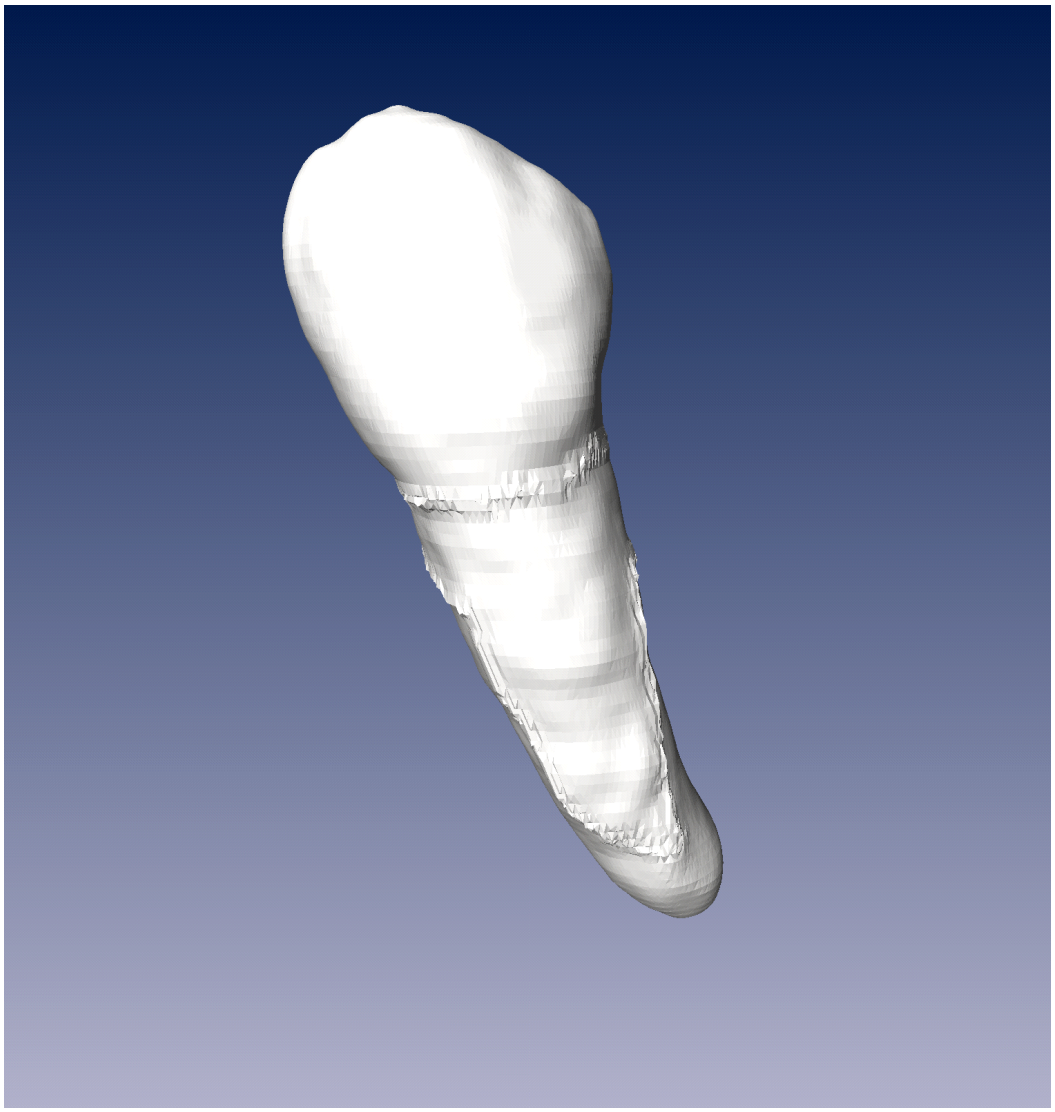


Figure 51c: *Result obtained when using the original GMC with all materials present. Note the strongly marked ridges that give a serious deviation from the ideal case.*

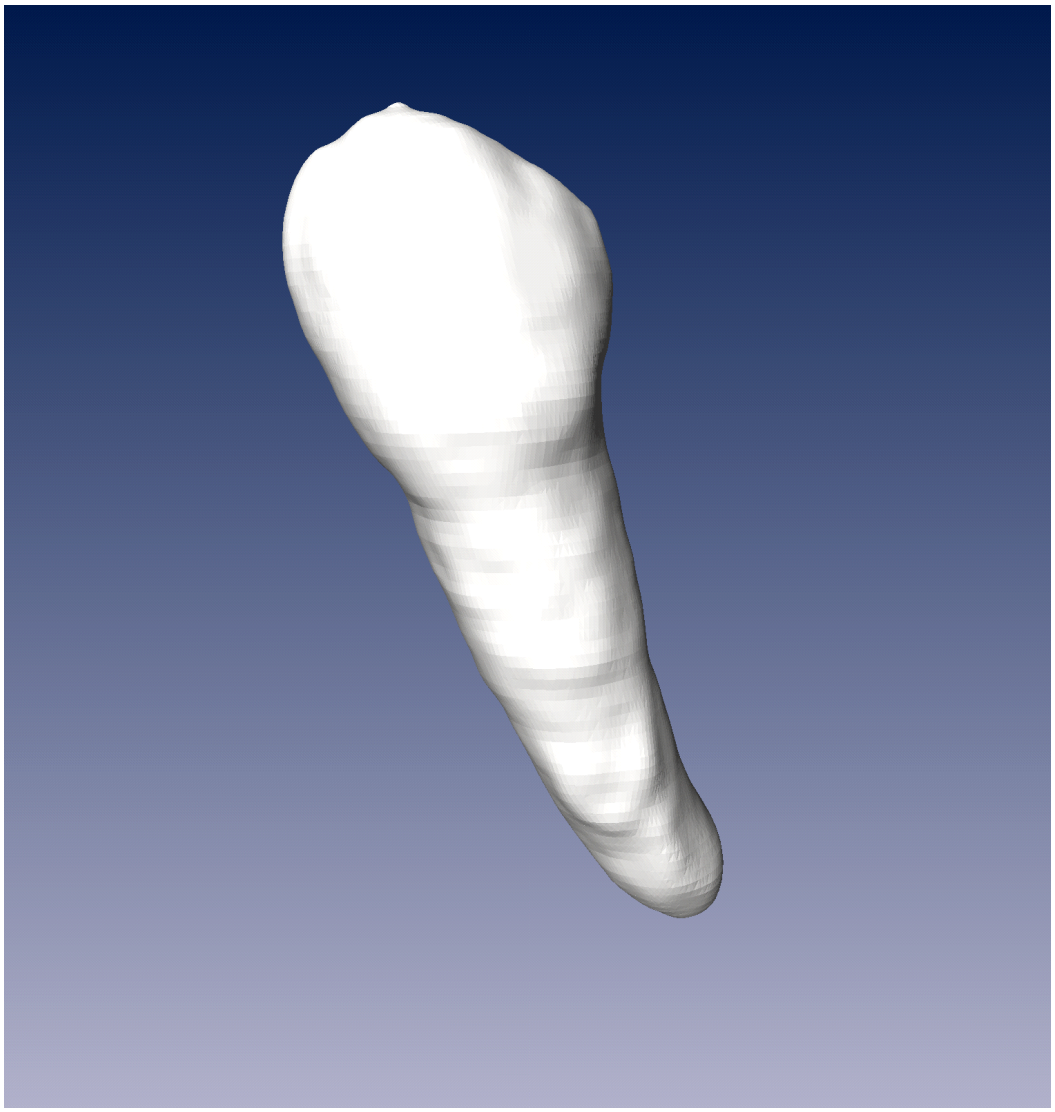


Figure 51d: *The result obtained when the new GMC is used and all materials are present in the labelfields. The ridges are completely gone.*

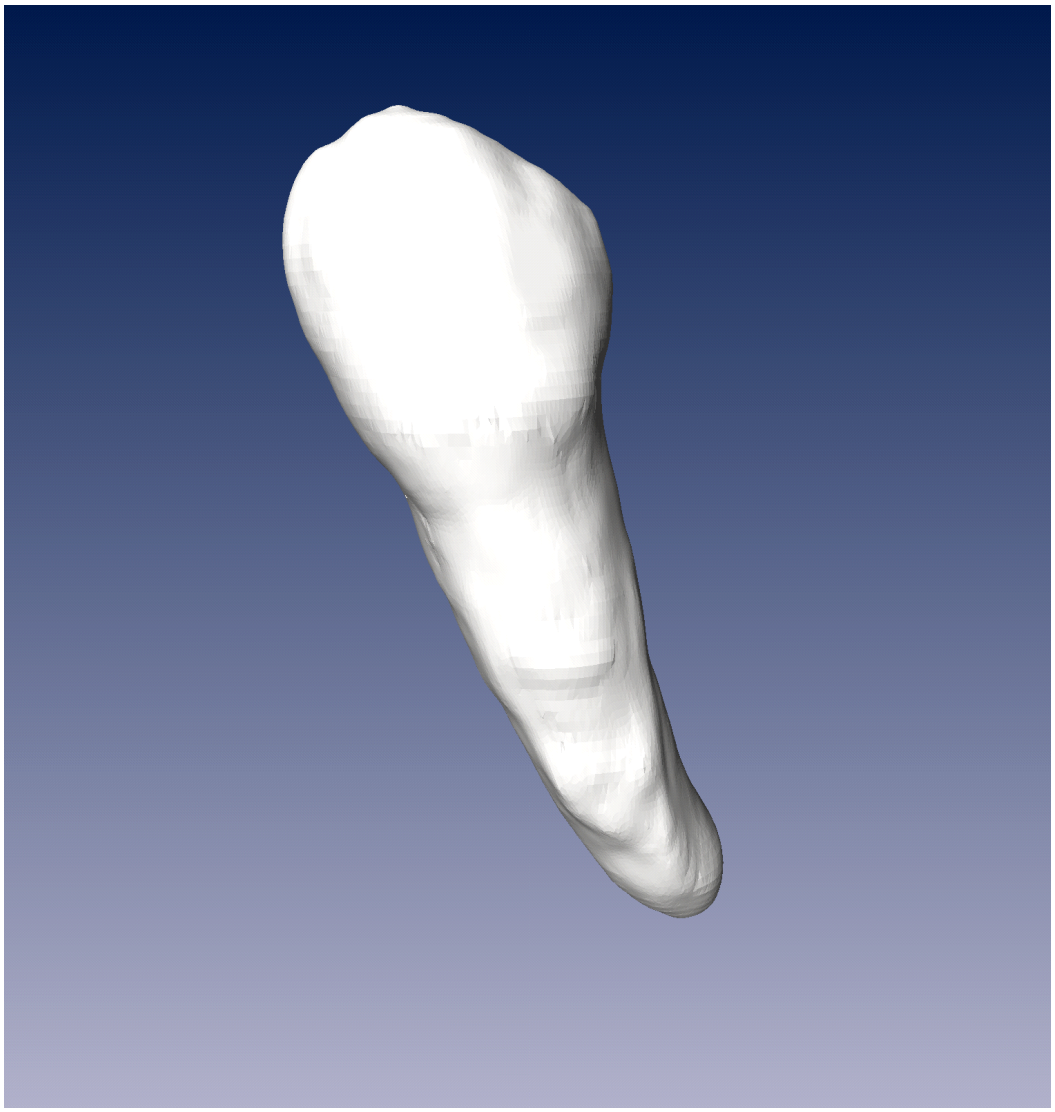


Figure 51e: *Result obtained after running the post processing module on the surface in figure 2c. The surface smoothing was done using the Taubin method with 100 iterations and the neighborhood depth set to 20.*

4.3 Data set 3, human

This data set contains thirteen materials representing the inner structure of the body such as bone, muscle, fat, veins and liver. The labelfield is of the size 256x256x67 and the resulting surface contains 317257 points and 639445 faces in 41 patches. The material selected to be smooth is bone. This is a very complex model to generate as there exist many boundaries between the different materials. Especially for the bone, as it touches a large number of different materials. This is the only model of the three that contains cases when four materials meet on the face of a grid cell during the triangulation. Thus, it's a very complex data set.

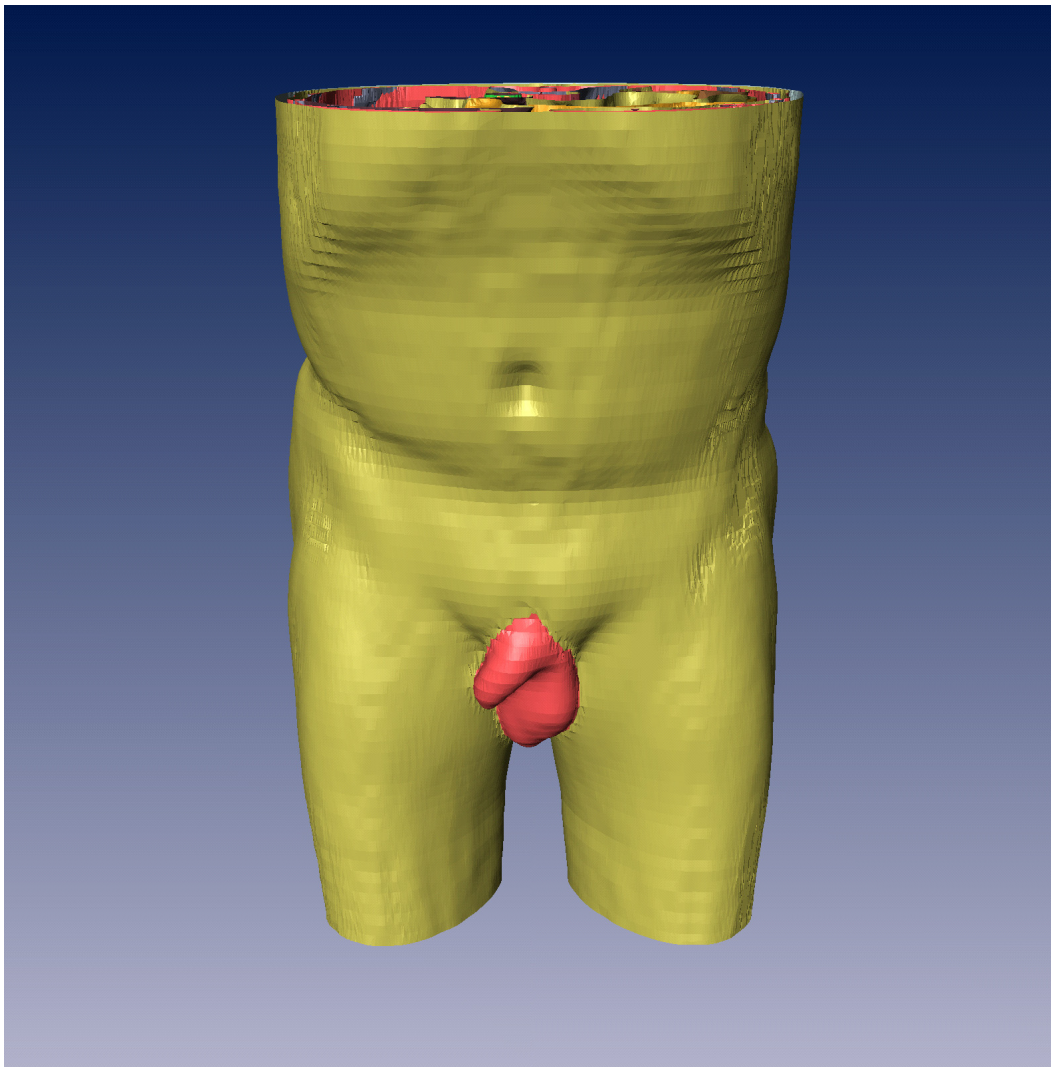


Figure 52a: *In the image all materials in the labelfield are visualized. Fat blocks the view of the other materials.*

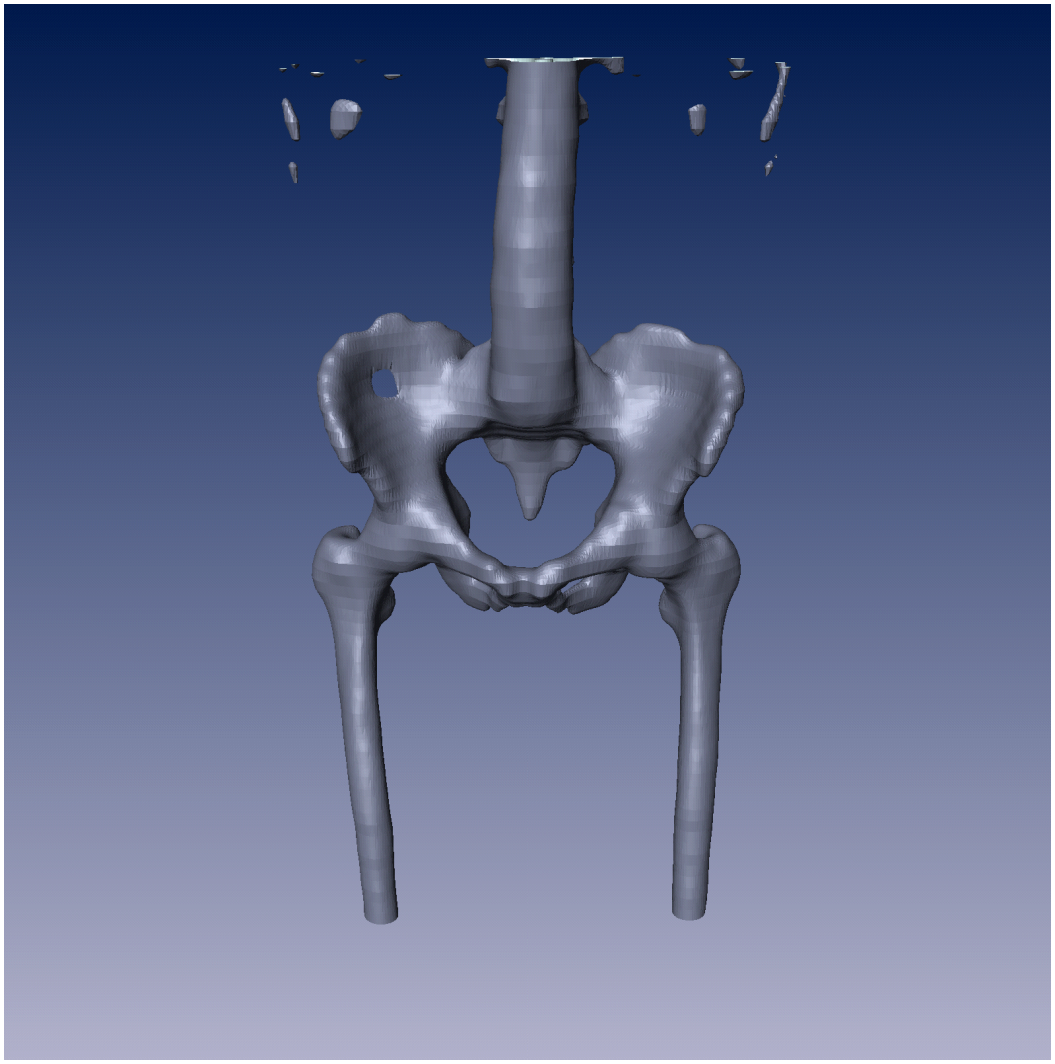


Figure 52b: *The ideal result for bone. Obtained by deleting all the materials from the labelfield but bone and then generating the surface with the original GMC.*

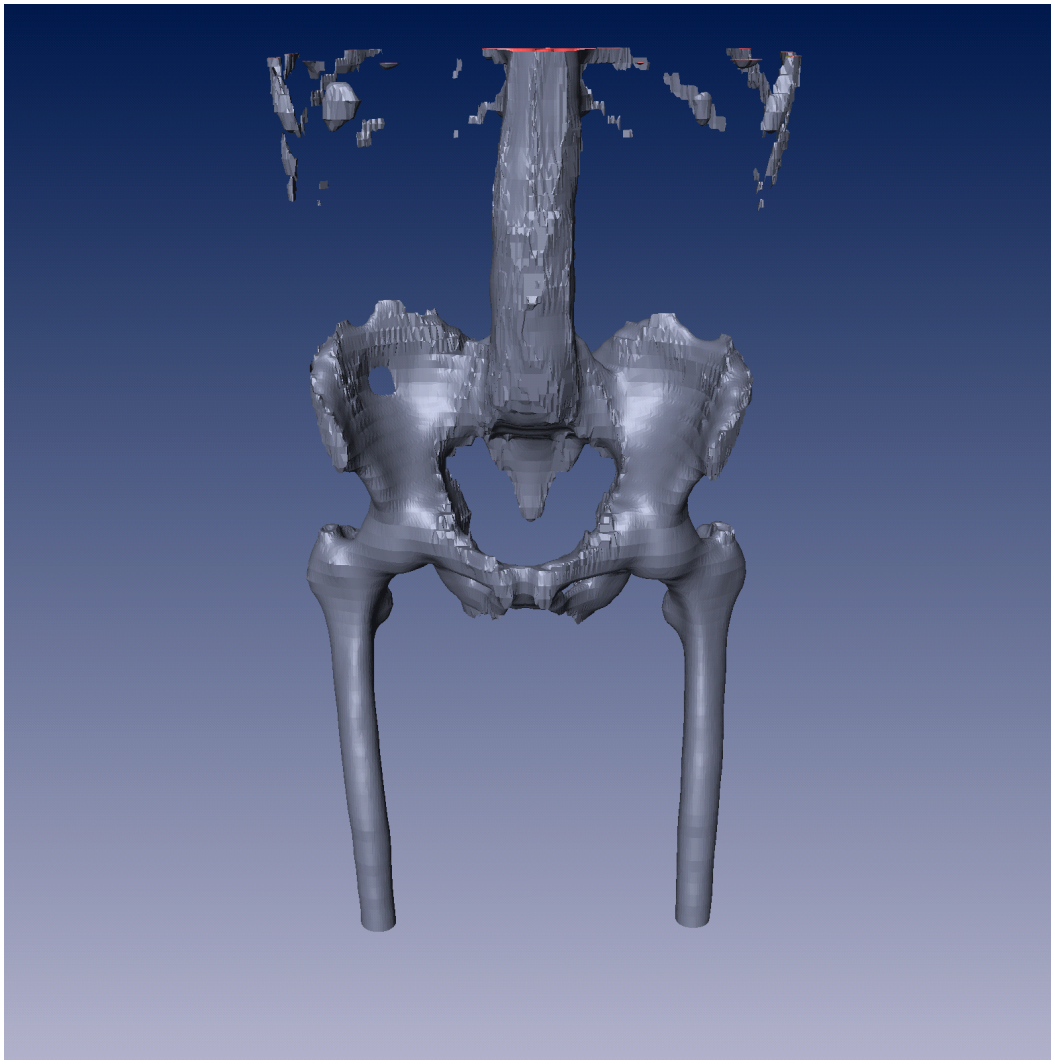


Figure 52c: *Result obtained when using the original GMC with all materials present. Since the model contains so many materials, ridges appear all over the surface distorting it in many places.*

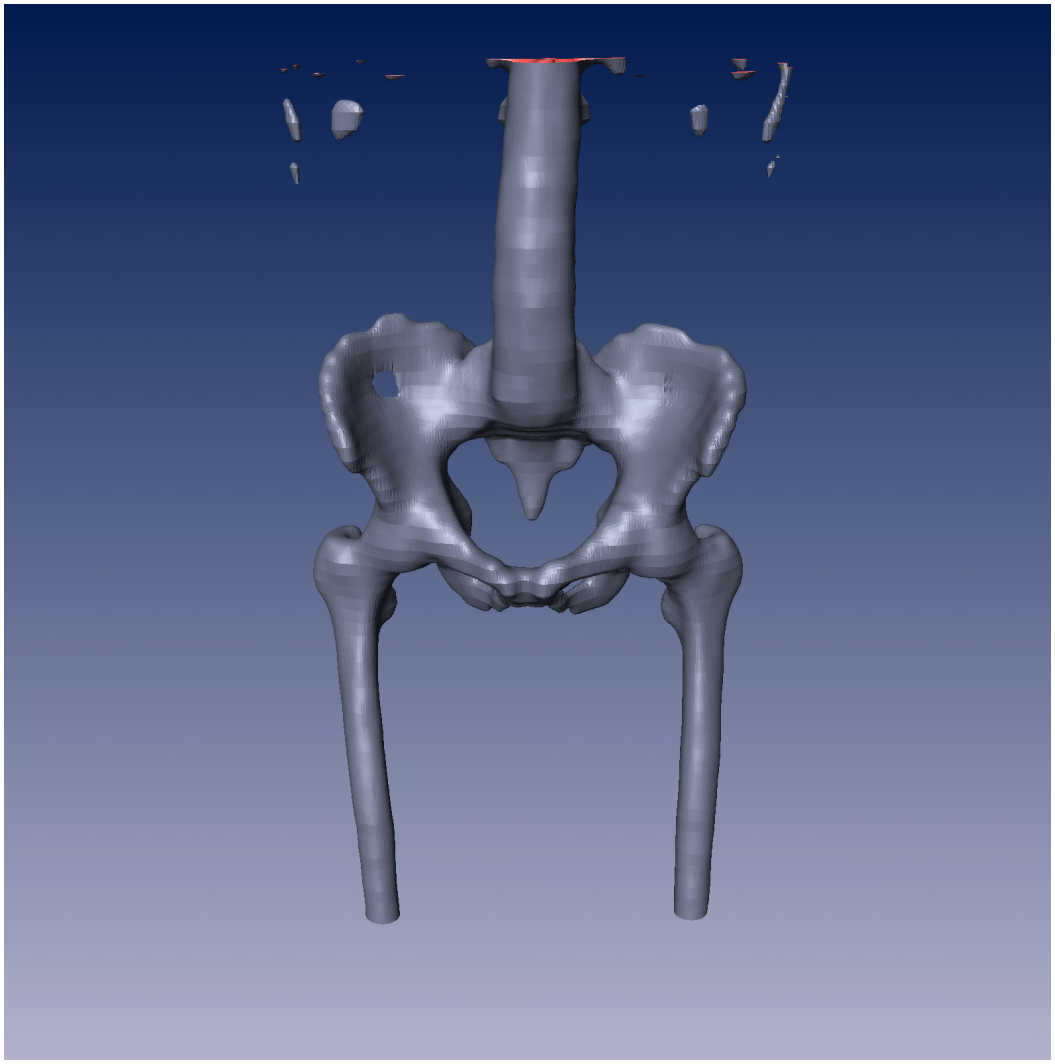


Figure 52d: *The result obtained when the modified GMC is used and all materials are present. The model is now remarkably smoother and very close to the ideal one. There exist only small deviations from the ideal surface which not are visible to the eye. Those have their origin in the differences in the smoothing process.*

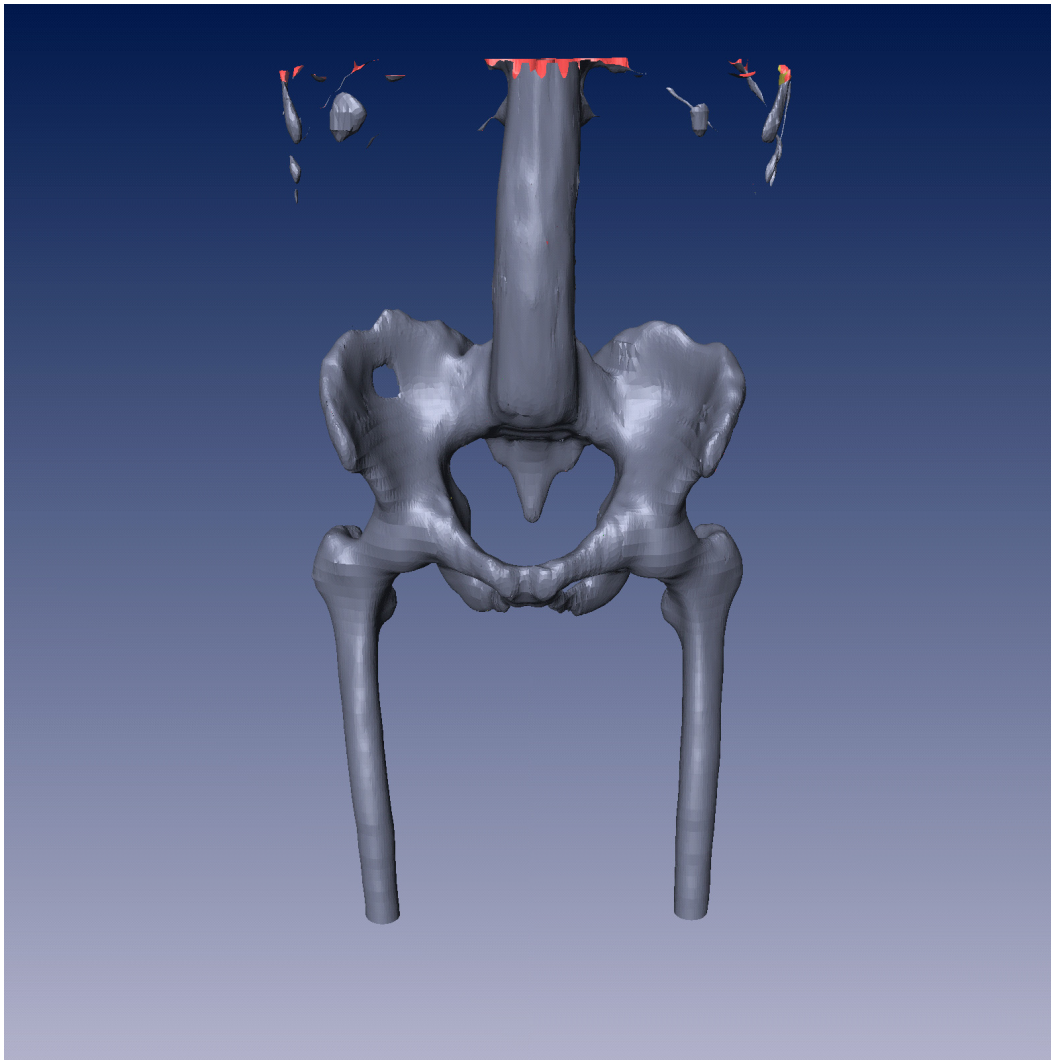


Figure 52e: *Result obtained after running the post processing module on the surface in figure 3c. The surface smoothing was done using the Taubin method with 100 iterations and the neighborhood depth set to 7. Using a smaller neighborhood depth is motivated by the size of the ridges. A too big neighborhood flattens the surface too much and suppresses detail.*

5 Conclusions

Removing surface irregularities such as ridges may seem like an easy task at a first glance, but it has several quite hard problems associated. Solving the problem in a good way calls for a solution that does not only gives a good visual result, but also a correct one regarding how the surface follows the labeled data.

The two approaches sought out to be evaluated as possible solutions to the ridge problem have been successfully implemented and are available for integration within the Amira framework. This means that the possibility now exists of removing the ridges in a computer assisted way in means of the post processing module. When the characteristics of the materials are known an automated generation of ridge-free surfaces using the modified GMC is available, which is a very interesting approach.

Both of the approaches together offer a good means of removing ridges in the general case. Together the post processing module and the modified GMC give the user the ability to correct a wide range of irregularities occurring where three or more materials meet.

After looking at the test results and the ease of use, the modified GMC gives very good results with minimal user interaction. This gives an interesting solution to the ridge problem as generation of ridge free surfaces can be automated when just one material is considered to be hard and smooth. However, when more than one material with these properties exists in a model, post process smoothing has to be applied. If the proposed changes to the GMC are implemented they might remedy this problem in some of the models and open the way for a real automated solution.

Even though the ridges may be small in size in the surface models, the impact of the ridges can be quite big when using the models in real-life applications such as simulations of stress, strains and pressure and for surgical planning. With the increasing speed of graphics processors, models of very high quality will be available in the future making medical imaging a very important field.

6 Appendices

Appendix A, Terminology

This appendix presents the necessary terminology for understanding the presentation of the visualization system. The terminology is divided into 5 groups: radiology, image extraction, visualization, the Generalized Marching Cubes algorithm and Amira specific.

Radiology

Figure A1: *Radiology is the first step in the 3D reconstruction pipeline.*

Computed tomography (CT)
Computed tomography is an examination method that uses x-rays to get cross sectional images of the body. A detector array connected to a computer receives the x-rays that pass through the body. The computer generates a picture showing the tissues with high absorption of x-rays as white areas and the tissues with low absorption of x-rays as black areas. A CT-scanning can create consecutive images with millimeter distance.

Magnetic resonance imaging (MR)
Magnetic resonance imaging uses magnetic fields and radio waves to obtain cross sectional images of the body. The MR works by exiting the hydrogen atoms of the body with radio waves. The energy is returned from the atoms when the radio waves are removed and is picked up by a sensor. As the hydrogen atoms belong primarily to the water in the body tissues containing the most water will send out the most energy and can be seen as white areas on the resulting image. A MR-scanning can create consecutive high resolution images with millimeter distance.

Single-photon emission computed tomography (SPECT)
This imaging technique consists of injecting a patient with a radioactive tracer that emits high-energy photons attached to a natural substance like for example glucose. When the substance travels through the body it can be registered by detecting the emitted radioactivity. That way a series of images can be obtained describing this process.

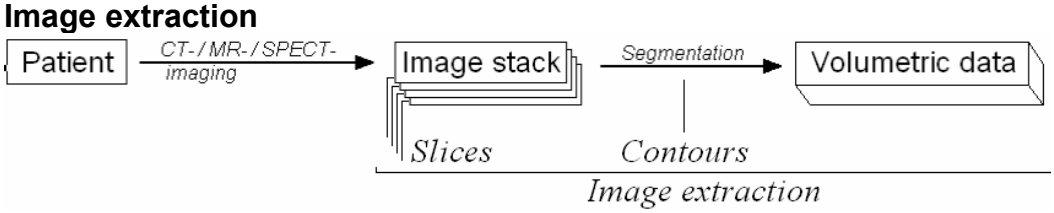


Figure A2: *The image stack generated from the radiology step is segmented into volumetric data during the image extraction step..*

Slices
Slices are cross sectional images taken by a MR- or CT-scanner or other acquisition device. The acquisition device scans a part of a body or object and takes several hundreds of pictures with small distances. When the images are aligned on top of each other they form an image stack.

Data points
The data points are the raw data from simulations or a MR- or CT-scanner or other acquisition device. In the case when the data is obtained from a MR- or CT-scanner the pixels in the slices correspond to the data points.

Materials
In a dataset the points are set to belong to different material classes, the materials are the computer representation of the real world tissues like bone, muscle, fat and so on.

Labels
A label indicates which material a point belongs to. Labeling is done during the segmentation step.

Segmentation
Segmentation is the process of separating a slice into regions with labels corresponding to the material. This way the data points from a slice are set to belong to different materials.

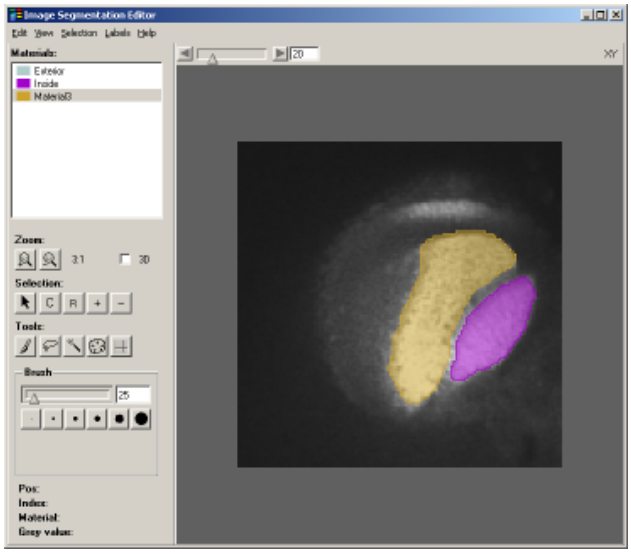


Figure A3: *A screenshot of the segmentation editor in Amira.*

Contours
The contours are the borders that separate the different materials from each other. The contours are found during a manual, automatic or semiautomatic segmentation step.

Volumetric dataset
A volumetric dataset is the 3D-representation of all 2D-data points from the pile of slices. A special case of the volumetric dataset is the lattice where the points are regularly arranged in space.

Lattice
A lattice is a regular arrangement of points in space.

Labelfield
A labelfield is a lattice where every point has a label associated, also called label-lattice.

Region
A labelfield is divided into different regions. In a region all points are labeled with the same label.

Visualization

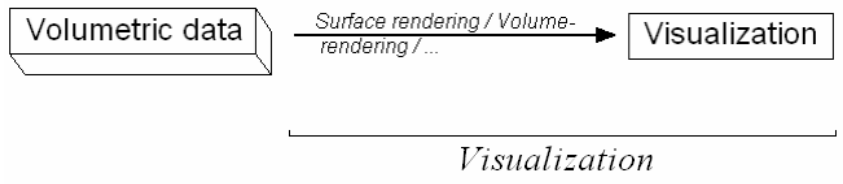


Figure A4: *In the last step of 3D reconstruction pipeline the data is finally visualized.*

Isosurface
An isosurface is a surface that separates two regions.

Polygonal mesh
The polygonal mesh is the graphical representation of the isosurfaces. The isosurface is divided into a mesh of polygonals that can be rendered by the graphical hardware in the computer.

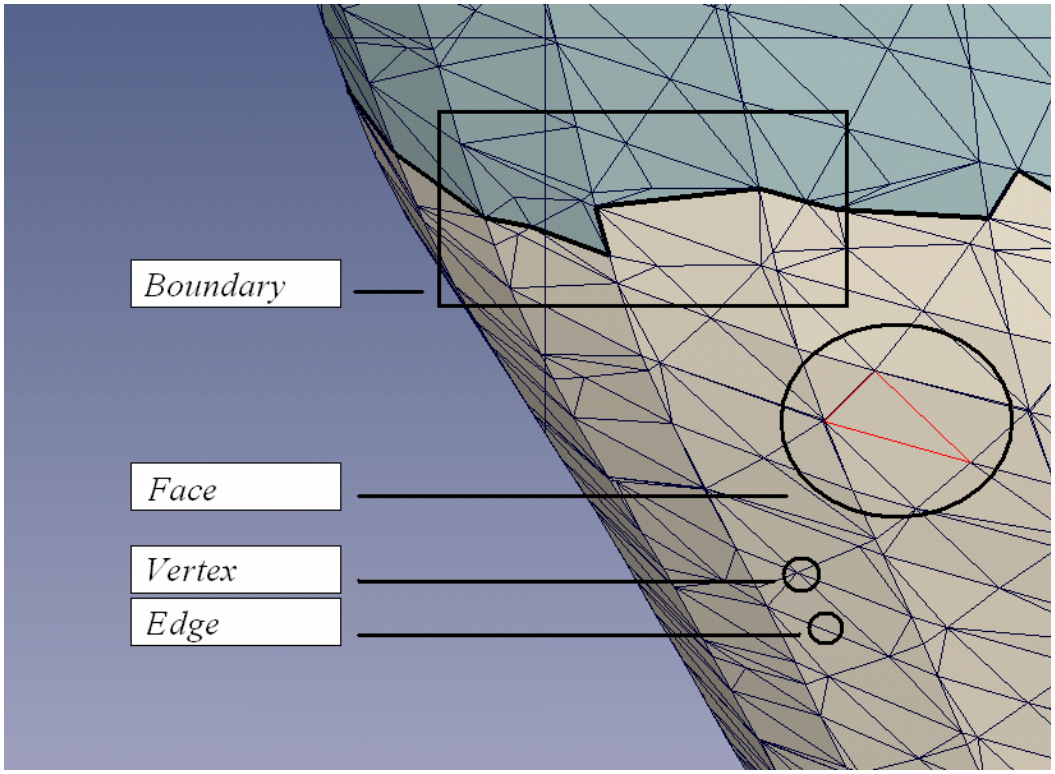


Figure A5: *A polygonal mesh*

Vertices
The vertices are the corners of the faces on the polygonal mesh. The positions of the vertices are determined by the generalized marching cubes algorithm.

Faces
Faces are generated during the polygonization of a surface. The faces are generated as triangles in Amira. A face consists of a set of vertices, normally three for a triangle.

Edges
An edge is non repeated pair of vertices. An edge can belong to one or more faces.

Patches
Faces that have the same combination of inside and outside material are said to belong to the same patch.

Boundary edge
A boundary edge is an edge that is not shared by two faces of the same patch. The edges are instead shared by 2 or more faces with different patches.

Boundary
The boundary is the closed curve created by all the boundary edges of the same type.

Manifold and non-manifold surface
By a manifold surface is meant a surface where the all the edges of the faces are of degree 1 or 2. In other words; an edge is a part of only one face or 2 faces share the same edge. A non-manifold surface has edges that are shared by 3 or more faces.

The Generalized Marching Cubes algorithm (GMC)

The GMC-algorithm generates isosurfaces from a labelfield with multiple vertex classes. The algorithm has been developed at the Konrad Zuse Zentrum for Informationstechnik in Berlin.

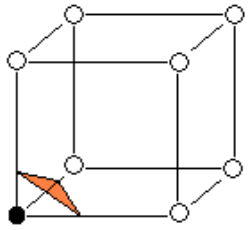


Figure A6: *The figure shows a grid cell where an isosurface passes through. The grid cell has 2 different vertex classes. Seven of the vertices belong to the white vertex class and one vertex belongs to the black vertex class. The isosurface separates the 2 vertex classes. The part of the isosurface inside of the grid cell consists of a single face which has been generated by the marching cubes algorithm. The corners of the face lie on the edges of the grid cell. The locations of the corners on the edges have been computed by using bi-linear interpolation for the weights of the grid cells vertices. The computed face configuration inside the grid cell is called the triangulation for the grid cell.*

Grid
The GMC-algorithm generates surfaces from a label-lattice of data points. The label-lattice is arranged as a uniform grid of points.

Grid cell
A grid cell is created from eight data points arranged like in figure A6. These data points are sometimes called the corners or the vertices of the grid cell. The

grid cell is the heart of the GMC-algorithm; here are the form and position of the isosurfaces calculated. The GMC-algorithm marches grid cell by grid cell through the grid.

Vertex classes
The vertices of the grid cell are divided into to different classes depending on which labels the vertices have. The configurations of vertices within a grid cell dictate how the resulting isosurfaces are divided by different patches. The GMC-algorithm can handle up to eight different vertex classes within the same grid cell, in other words; it is possible for up to eight different materials to meet in the same grid cell.

Weights
Weights are assigned to the vertices of the grid cell to specify the weight for the vertex classification. The weights are generated either at creation of the labelfield or in a later smoothing step.

Bilinear interpolation
When an isosurface passes an edge in the grid cell the point of intersection is calculated with bilinear interpolation. This is the case when the edge of the grid cell is shared by 2 vertices belonging to 2 different vertex classes. The intersection point is found by examining the weights of the 2 vertices.

Triangulation
Faces are generated after the calculation of the isosurface intersections within the grid cell and on its edges. The computed face configuration inside the grid cell is called the triangulation for the grid cell.

Topological cases
The topological cases are the different ways in which a grid cell can be triangulated. Figure B1 in appendix B shows the possible triangulations for up till 3 different vertex classes.

Look-up table
The most common topological cases are stored in a look-up table in order to speed up the generation of the isosurface. The more uncommon topological cases are computed on the fly by the GMC-algorithm and are later stored in the look-up table for reuse.

Amira

Tcl
Tcl is a popular scripting language with a quite simple syntax used to control components in Amira.

TGS Open Inventor
A graphics toolkit built on top of OpenGL allowing describing and rendering of 3D scenes.

Qt
A multi-platform GUI software toolkit that adds portability to graphical user interfaces, making applications compile on UNIX, Windows and other platforms.

Twisted view
Visualizing the patches in different color, making the boundaries between them easy to distinguish

Appendix B, Topological cases

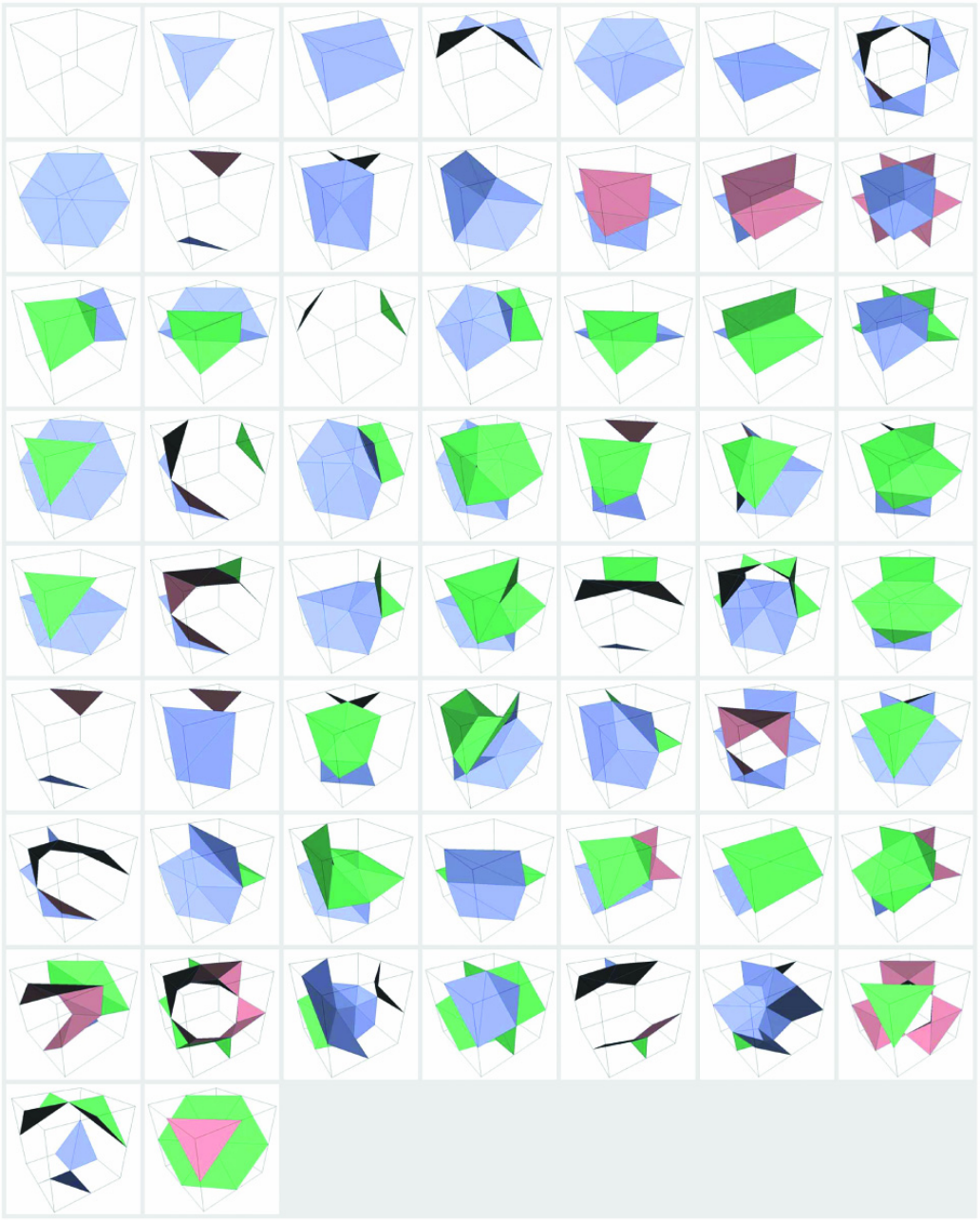


Figure B1: *The image shows the triangulation for up to three vertex classes. The first two rows contains the cases with only two classes. The image shows the look-up table for up to three different vertex classes. The first two rows.*

Appendix C, A brief user guide to the new GMC

The modified version of the GMC contained in the SurfaceGen module shows very few differences in the user interface from the original one. A new multi menu exists with the ability to select the material to be smooth. Apart from this, the module works in the same way as the original one, different smoothing techniques can be applied, the ability to add a border material exists and so on. To the right the interface is shown.

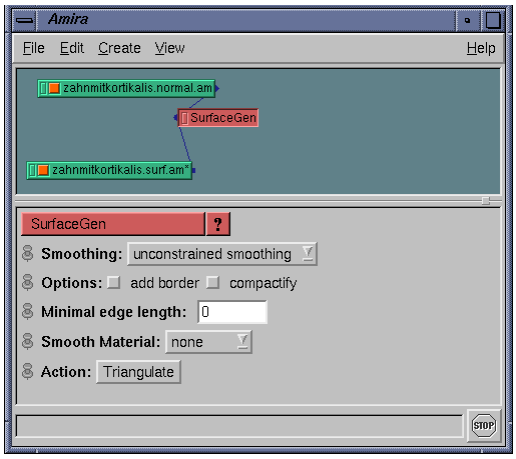


Figure C1: *Snapshot of the new GMC modules graphical interface*

The upper part shows the SurfaceGen module together with the module in its context, connected to a labelfield and surface module as its output. The lower part is the interface to the module, showing the different options that exists for the user

Appendix D, A brief user guide to the post processing module

The post processing module can be attached to a surface module. Then the user has the ability to choose from a number of options.
The picture on the right shows the interface with the module connected to a surface in the upper half and in the lower half the different options that exist for smoothing.
Below follows a description of the different ports and what they do.

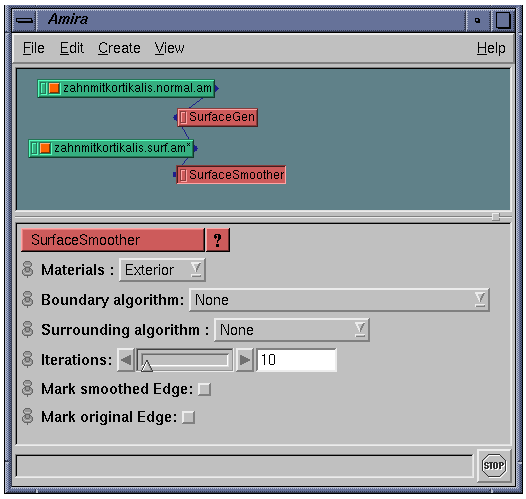


Figure D1: *Snapshot from the post processing module*

Description of the ports

Materials
The port where the material containing the ridge to be smoothed can be selected.

Boundary algorithm
The algorithm to be used at the boundary points, the ones that have edges between them separating patches.

Surrounding algorithm
The algorithm to be used at the points surrounding the ridge to a defined depth.

Iterations
The number of iterations steps the algorithms should be performed. The more steps, the smoother the result. For smaller ridges not containing so many triangles a smaller value can be used, down to 1. For bigger ridges containing more triangles a bigger value has to be used. This also depends on the structure of the surroundings to the ridge.

Surrounding depth
This option only appears when a surrounding algorithm is chosen. Here the user can specify which neighbouring points that should be moved in the smoothing step. The depth is defined such that a depth of 1 gives just the points that can be reached from the boundary following just one edge (the first order neighbours). A depth of two gives all the points that can be reached following 2 or less edges (the second order neighbours) and so on.

Mark smooth edge
Toggle to mark the final smoothed edge with red lines.

Mark original edge
Toggle to mark the edge as it was before the smoothing with black lines.

Beta
This option appears when "following normal & ridge vector" is chosen. It defines a percentage that the points should be moved back in the direction of the surface normal extracted from the neighbours to the points chosen for smoothing. This is to avoid a shrinkage effect. Normally 0.9 is a good value when many iterations are made, but for fewer iterations the value should be lower to move the points more and in that way flatten out the ridge faster.

Lambda and Micro
These two parameters are only visible when choosing the Taubin algorithm. They define the pass band frequency and the standard values are optimized for a fast and stable filter. These values can be manipulated to change this frequency.

Description of the algorithms

The following algorithms are available. For treating the boundary points between the patches all the algorithms can be chosen, but for the surrounding points just the Taubin filter algorithm is available:

Taubin filter (Lambda/micro filter)
This technique simulates a low pass filter, the filter parameters are also available but there is no need to alter them if you are not an advanced user. They could be re-calculated to produce a better filter for giving a special pass-band frequency and so on. Normally a large number of iterations are needed in order to see good results when treating large ridges, between 40-100 iterations, but with small ridges a smaller number of iterations, down to as few as 5 can produce a good result.

Following normal & ridge vector
This algorithm moves every point towards the mean value of its first order neighbours, but just in the direction of the surface normal and the ridge vector at this point.

Neighbours on boundary Taubin filter
This algorithm is mainly used for straightening out the boundary that exists between two patches as a post processing step to the smoothing. It works in the same way as the normal Taubin filter, but it just considers the points on the boundary. To give a good result it should be combined with a surrounding algorithm.

7 References:

[1] Gabriel Taubin, Eurographics 2000, *Geometric Signal Processing on Polygonal meshes* – http://www.research.ibm.com/people/t/taubin/publications.html (link validated 2004-05-21)

[2] D. Stalling, M. Zöckler, O. Sandler, H. E. Hege, 1998, *Weighted labels for 3D image segmentation* – http://www.zib.de/bib/pub/pw/index.en.html (link validated 2004-05-21)

[3] H.C. Hege, M. Seebass, D. Stalling, M. Zöckler, 1997, *A generalized marching cubes algorithm based on non-binary classifications* – http://www.zib.de/bib/pub/pw/index.en.html (link validated 2004-05-21)

[4] William E. Lorensen, Harvey E. Cline, 1987, *Marching cubes: A high resolution 3D surface construction algorithm*

[5] Gabriel Taubin, 1995, *Curve and surface smoothing without shrinkage* – http://www.research.ibm.com/people/t/taubin/publications.html (link validated 2004-05-21)

[6] Gabriel Taubin, Tong Zhang, Gene Golub, 1996, *Optimal surface smoothing as filter design* – http://www.research.ibm.com/people/t/taubin/publications.html (link validated 2004-05-21)

[7] Kondrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), *Amira User's Guide* – http://www.amiravis.com/usersguide31/usersguide/index.html (link validated 2004-05-21)

[8] C. Kober, B. Erdman, R. Sader, H.–F. Zeilhofer, 2003, Sim*ulation of the Human Mandible: Comparison of Bone Mineral Density and Stress/Strain Profiles due to Masticatory Muscles' Traction* – http://www.zib.de/bib/pub/pw/index.en.html (link validated 2004-05-21)

[9] C.T. Zahn, R.Z. Roskies, IEEE Transactions on Computers 1972, *Fourier descriptors for plane closed curves*

[10] J Bloomenthal, K Ferguson, 1995, *Polygonization of Non-Manifold Implicit Surfaces* – http://www.unchainedgeometry.com/jbloom/pdf/SIG95-NonManifoldPolygonizer.pdf (link validated 2004-05-23)

[11] G.M. Nielson, Richard Franke, *Computing the Separating Surface for Segmented Data*, IEEE Visualization '97, Oct., pp. 229-233, 1997.