UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

Combination of Techniques to Enrich Visualization in Dentistry

Rosane Minghim
Igor Prata Soares
Maria Cristina Ferreira de Oliveira
Luis Gustavo Nonato

N$^{\underline{o}}$ 127

RELATÓRIOS TÉCNICOS

São Carlos – SP
Dez./2000

SYSNO 1721466
DATA / /
ICMC - SBAB

Combination of Techniques to Enrich Visualization in Dentistry

Rosane Minghim
Igor Prata Soares
Maria Cristina Ferreira de Oliveira
Luis Gustavo Nonato
Departamento de Ciências de Computação e Estatística
ICMC/USP - São Carlos
Caixa Postal 668
13560-970, São Carlos - SP, Brazil
{rminghim, igor, cristina, gnonato}@icmc.sc.usp.br

Abstract

This paper reports on the implementation of two different approaches for visualization of dentistry data, discussing their role in providing adequate support for training, exploration, simulation and interaction in Dentistry applications. One of the approaches is a reconstruction technique based on the Delaunay Triangulation, and the other is a Direct Volumetric Rendering and Visualization technique with texture mapping. The results are evaluated in the light of the target applications, which are teaching and training in dentistry, as well as simulation of dental procedures and illnesses. Uses for multiple-technique visualization and exploration are discussed.

1: Introduction

Visualization plays an essential role in the way people conduct investigation and research in many fields of expertise. While this is particularly true in Medicine, for example, this trend has been less noticeable in the related field of dentistry. This is partly due to the different approaches taken to diagnosis in both areas, since images do not play the same role in diagnosis for dentistry as they do in other health sciences. It is very unusual to have to have CT or other imagery data from teeth. Slicing of real teeth photographed under the microscope is the most probable way of getting real data from teeth and neighboring structures. The construction of three-dimensional models to represent the body's internal parts detected by two-dimensional (2D) means such as scans is a major goal in many applications; they are useful to all sorts of tasks, from educational systems to technologically advanced surgery. These reconstruction processes are also applicable to dentistry, which can benefit from the visualization of 3D models of teeth and mouth. In this area, initial reconstruction tasks have been performed, using some general-purpose packages [Goe98] and building extensions to others [Shi98]. Some other activities in the field have been reported, including mandible reconstruction [Sei95] and chewing simulation [Mys97].

This paper describes part of the work performed in the context of a project for Virtual Dentistry, which intends to support education, training and assistance for dentistry professionals and students, as well as to perform simulations of dental procedures, using scientific visualization techniques.

Most of the graphical visualization approaches to presenting scalar data are classified under two categories [Kau98; Sch98]: surface fitting (SF) and direct volume rendering (DVR). In surface fitting methods, that sometimes include volume reconstruction as well, geometrical structures are created from data sets that represent the phenomenon under study. These geometrical structures are then presented to the user employing conventional computer graphics rendering. The second category of visualization methods, DVR, generates images straight from the data sets, without creating intermediate structures. Both types of techniques are being developed, implemented and tested within the scope of the Virtual Dentistry project.

This paper discusses the implementation and use of two algorithms for visualizing 3D structures in teeth. The first is a novel and very effective reconstruction method based on the Delaunay Triangulation (DT) [Non98], which is now adopted to build 3D models of teeth, instead of a previously used technique [Shi98] and of another DT-based method we tested [Non00b]. The other technique is based on a DVR algorithm that uses 3D texture maps (DVRT) to allow for use of appropriate hardware during the image generation process [Gel96], which we implemented as an extension to a general purpose visualization software, the VTK (Visualization Toolkit [Sch98]). Both approaches have their strengths. Reconstruction is usually necessary when performing computing tasks such as simulation and design, and DVR techniques are useful when observing both the whole and the details of the interior and boundaries of structures.

We conclude the paper by discussing the combined role of these two techniques for the Dentistry case and other medical areas, and the associations between them for exploration in the context of virtual reality applications.

In the following section we present the surface reconstruction technique implemented and summarize its advantages over the reconstruction approach previously adopted in the project. In Section 3 we present the DVRT technique and describe its VTK implementation. In Section 4 we discuss some aspects related to the use of each approach and how they can be integrated for further visualization, exploration and virtual manipulation of teeth and related structures.

2: The DT-Based algorithm

Many reconstruction techniques have been developed to create computer models from planar data. Those based on computational geometry usually employ Delaunay triangulations (DT) and Voronoi diagrams [For94] to obtain volumetric and surface models [Boi88]. Some of the problems encountered with such techniques include algorithm instability, that calls for perturbation of original data; difficulties handling singularities (or no handling at all, causing models to be unsuitable for some applications); and problems handling finely sampled data. Recently, a technique was developed is handling well all these issues [Non98], particularly for the tooth case [Non00a; Non00b], and is now adopted as the standard model generation procedure in our dentistry project. Working on 2D contours, its advantages include the guarantee of recovery of the original contour set under slicing, a good yet flexible decision mechanism for component connection, good handling of singularities, and capability of volume reconstruction. Additionally, it produces a very well behaved set of elements in the model, in that the tetrahedralization of the volume (and therefore the triangulation of the border) is not too large, does not contain elements too small or too large, and does not present distortions of shape. The results so far, some of which are reported here, are very satisfactory, and compare very positively with other reconstruction techniques.

The algorithm begins by generating a DT from the oriented set of contour points, and then classifying the formed tetrahedrons as internal, external or on the contours. After that, one of the main aspects of the technique takes place, the identification of reverse tetrahedrons. These are tetrahedrons that have one edge in each slice, without them being contour edges. They appear every time two contours are well positioned geometrically [Non98], as shown in figure 1. Existence of reverse tetrahedrons is used to define the individual components. Then, tetrahedrons with external edges between components are eliminated to disconnect them.

In each connected component, tetrahedrons with external edges must also be dealt with, because if left behind they make recovery under slicing (a sought requirement, as mentioned before) impossible, as figure 2a illustrates. If simply eliminated, they cause singularities, as shown in figure 2b. That is solved by subdividing such tetrahedrons rather than simply eliminating them, in such a way that the extra vertices are placed in-between slices, keeping original planar sections undisturbed and preventing singularity at the same time (see figure 3). Figure 4 shows the visual aspect of the resulting mesh and its rendered image, as it deals with singularities.

The volumetric model generated with the above process has an important property: it is a volumetric Picewise Linear manifold, so it is very appropriate to be used in numerical simulations like finite element and finite volume analysis. The size of the model produced is generally suitable for real time rendering and interaction. It also allows for surface orientation and it did not produce defects in any of the tests performed. Tests with finely grained data also conveyed stable behavior. For all these reasons, the method suits well the interaction and simulation purposes of the dentistry application. Its advantages compared to the other mentioned above reside mainly in the favorable topology of the model formed [Non00b].

The DT-based (DT) approach was implemented using C in a Unix environment, and the resulting meshes were translated into the VTK file format in order to be displayed. Additionally to the visualizations, VRML models of the reconstructed teeth were created using VTK exporters. Exploration of the structures formed was quite fast, even using modest machines. The algorithm will next be implemented within VTK, in order to be integrated with the other techniques implemented and under development for visualization in dentistry.

Section 4 shows some results of the application of this technique to tooth data. In the next section we present the DVRT technique and its implementation in the context of the toolkit VTK.

3: DVR with use of texture (DVRT)

Several texture mapping strategies have been proposed as alternatives for implementing fast Direct Volume Rendering of scalar data [Cab94; Grz98; Tes95; Gel96]. Theoretically, a texture mapping approach that performs a surface order sampling of the volume data has the same computational complexity of the traditional ray casting approach, which samples in ray order [Grz98]. Both approaches generate images of comparable quality by sampling the whole volume using interpolation and combining samples with a blending function to produce a pixel value. The advantage of a texture mapping approach is that graphics hardware can be employed for the implementation of the sampling and blending operations, with very good speeding up results. Some of the limitations of a hardware-based approach are the difficulty to use advanced shading techniques, and that the available texture resolution may limit the number of planes that can be sampled.

In this section we describe an implementation of a DVRT 3D algorithm – Direct Volume Rendering with 3D Textures - in VTK. In Section 3.1 we present an overview of the DVRT algorithm, introduced in [Gel96]. In Section 3.2 we describe how it has been implemented in VTK.

3.1 Overview of the DVRT Approach

The technique generates a 3D texture map that is applied to several planes, which are then combined to produce a volume image. The algorithm works in two stages: first a 3D texture map is generated, and then textures are sampled with planes parallel to the viewing direction and rendered. Prior to starting the DVRT process the user must perform a data classification that associates color and opacity values with data values of interest by providing appropriate transfer functions. The 3D texture map is defined by texels that store intensity and opacity (RGBA) values. We shall refer to the texel's *color* as the combination of its R, G and B intensity values, plus its opacity value. The texture volume is constructed from the data volume under analysis: a texel stores the color in which the scalar value of the corresponding voxel in the volume has been mapped by the classification process.

Once the texture map is available, it is sampled with parallel *sampling planes* perpendicular to the viewing direction [Gel96]. Each sampling plane collects color information in the texture map. After the sampling process, it contains the texture image that would result from cutting the volume texture with the sampling plane. The intersection between the volume texture and the sampling plane is called a *texture polygon*. The images of the texture polygons are then blended to generate the volumetric image of the data set.

The 3D texture generation process may be modified to incorporate directional lighting and reflection, so that texel values are influenced by the light sources in the scene. This requires, for each voxel, in addition to its scalar value, a gradient. It may be estimated from the voxel's neighborhood, and from its classification into either ambient or reflecting. A voxel must be classified as reflecting if it is in the boundary region between different materials in the volume, and as ambient if it is inside a homogeneous region of material. A material is defined by a range of scalars and the their illumination coefficients. Thus, the volume data must be pre-processed to obtain a classification for their voxels and also the quantized gradients for the voxels. Once this information is available for each voxel, the texture map can be created.

For an ambient voxel, its corresponding texel receives the RGBA value returned by the user-provided transfer functions multiplied by the ambient coefficient provided in its material definition. For reflecting voxels, their corresponding texel receives the RGBA values from the transfer function which are further combined with all reflection and ambient components of the voxel's material by an illumination model. The reflection components may be stored in a lookup table to speed up the process.

The process of creating the texture map is driven by different events. The voxel classification process is triggered whenever the transfer functions or the gradient parameters are modified. The reflection table is recomputed whenever the illumination parameters or the viewing direction change. Both processes imply in reconstructing the texture volume. Additional information on the DVRT 3D algorithm with shading is provided in [Gel96]. The implementation of this algorithm into the Visualization Toolkit is described in the following section.

3.2 VTK implementation of the DVRT algorithm with shading

VTK is a low cost visualization library with available source code. It is object-oriented, and can be expanded and changed to fit an application. It can be programmed in C++, Java and Tcl/TK.

A VTK program defines a visualization pipeline in which classes are instantiated to produce a visualization image. A pipeline connects data objects and processing objects that operate on the data to produce a graphical representation to be rendered. Many visualization techniques are implemented as processing objects that act as filters, producing transformed data objects from input data.

As an illustration, the Ray Casting algorithm for DVR is implemented in VTK as a class that takes as input a volumetric dataset of type *vtkStructuredPoints*. This class represents a geometrically and topologically regular dataset grid. The technique is associated to a *vtkVolumeRayCastMapper*, responsible for generating the graphical output for an object of the type *vtkVolume*. This class inherits the properties of an abstract class named *vtkVolumeMapper*.

The VTK DVRT implementation follows the same structure of the VTK implementation of ray casting as far as object connectivity is concerned. A C++ program that uses VTK to perform a DVRT has the structure presented in figure 5. Two new classes have been defined for VTK, named *vtkVolumeTexture* and *vtkVolumeTextureMapper*, whose implementations are described in the following text. The remaining classes in the program are the same that would be required for a conventional DVR with ray casting in VTK.

The *vtkVolumeTexture* class

The *vtkVolumeTexture* class implements the definition of the 3D texture map. The texture map is conceptually equivalent to the data volume, except for the fact that it stores RGBA values rather than scalar values. The definition of materials within the volume, required for the classification process, is done by calling the method *AddMaterial* defined for this class. The transfer functions are defined by using the appropriate VTK classes, and are connected to the volume by the methods *SetColor* and *SetOpacity*, as it can be observed in figure 5. Similarly to the data set, the texture map is represented as a VTK object of the type *vtkStructuredPoints*.

To incorporate directional lighting and shading, two other classes support the construction of the texture map. The class *vtkVectorIndex* creates and manages a table of unit vectors, which is constructed using the same approach described in [Gel96]. The class *vtkReflectionTable* creates the reflection table. As shading in DVRT is computationally expensive, the *vtkVolumeTexture* class provides the methods *ShadeOn* and *ShadeOff* so that users can choose directional ligthing, and the default is to have the shading option off. Users interact with these classes indirectly by using methods defined for *vtkVolumeTexture,* such as *SetGradientIndex* and *SetLevelOfRefinementToGradientIndex.*

The *vtkVolumeTexture* class must be associated to the class *vtkVolumeTextureMapper* for the sampling and rendering stages to be performed.

The *vtkVolumeTextureMapper* Class

As it has been mentioned, the class *vtkVolumeRayCastMapper* implements rendering by ray casting in VTK. By analyzing how the class *vtkRenderWindow*, which implements the rendering process, invokes the ray casting function, we realized that to implement DVRT it would be necessary to define a corresponding mapper class. Analogously to *vtkVolumeRayCastMapper*, this should be derived from the abstract class *vtkVolumeMapper.* To render a visualization model using texture mapping, the user must invoke the *Render* method defined for *vtkRenderWindow.* This will trigger a sequence of method invocations until the *Render* method of the *vtkVolumeTextureMapper* class is triggered, starting the texture rendering process.

The basic function of *vtkVolumeTextureMapper* is to fill in two arrays of real numbers named *RGBAImage* and *Zimage*, both with dimensions equivalent to the resolution of the resulting image. Such arrays contain information on each pixel of the image to be displayed. The first contains the pixel's RGBA values in the interval [0,1]. The second one contains depth information, thus defining hidden surface information for each pixel. Both arrays must be updated whenever the *Render* method is invoked as a consequence of modifications in the scene, or in the camera attributes. Updating either of them requires executing the whole sampling process if shading is off or the complete DVRT process if shading is on.

The sampling process is as described in [Gel96]. The sampling planes are positioned in a *bounding cube* that contains the whole texture map in any orientation, that is, the bounding cube must have side lengths equal to the diagonal of the texture volume. Each sampling plane is calculated parallel to one of the faces of the bounding cube. The distance between consecutive sampling planes is defined by the user, through the method *SetSampleDistance*. The sampling process could be accelerated if only the necessary points in the planes were sampled. So, only the points that are within the texture map are sampled, by defining texture polygons that delimit exactly the area of the each texture plane that has higher probability of collecting visible colors in the texture map.

Processing of the sampling planes proceeds in a back-to-front order. The array *ZImage* is initialized with the farthest distance to the viewer, and de array *RGBAImage* is initialized with completely transparent color. Each plane is sampled, thus setting RGBA values in *RGBAImage*. As sampling planes are processed, both arrays are updated. If the color collected for a given position of the sampling plane has opacity zero, nothing is done. Otherwise, the point's corresponding position in the array *RGBAImage* is updated with a blending operation, and the same position in the array *ZImage* is set with the plane depth. After the sampling, both arrays have the information required for the display on screen, that is, the volume image and corresponding depth values. Once these arrays are available, the appropriate VTK classes may be invoked for displaying the resulting image. The approach as implemented is suitable for rendering with a parallel projection, and therefore the VTK object that implements the camera must be adjusted accordingly. Figure 6 shows the result of a visualization without shading.

In the following section we present some of the visual results obtained in the tests.

4: Results

This section presents visualizations of the dentistry data, obtained with both algorithms presented here.

The data were obtained using a contour editor [Goe98], based on real tooth values. They correspond to a set of contours extracted from slices of two surfaces of a tooth. One was internal, and the other external, both preserving the canal of the tooth.

For the tests employing the DT based algorithm (Nonato's algorithm), these contours were processed directly. For the tests using DVRT, the contours were sampled on a regular grid (the type required by DVRT) using a distance-based sampling technique [Shi98]. These data sets were fed into the two algorithms, that generated the images analyzed in the following text.

Figure 7 shows some of the results for the tooth data. In 7(a), the set of contours is presented. In all, they have 6393 points, in 290 contours on 80 slices. Nonato's algorithm performed really well in all the tests. This data set provided many of the difficult cases in reconstruction, such as bifurcation and connection decisions, and the results were extremely satisfactory. Figure 7(b) shows the presentation of the mesh formed by the reconstruction of the whole model. Internal and external surfaces were reconstructed separately in this example, although reconstruction of nested contours was performed just as well, with similar visual results. The mesh produced is suitable to simulation, as discussed before, and produces a very even configuration of triangles on the surface, as can be observed from the view in figures 7(c) and 7(d). In all cases, it produced better geometrical and topologycal results than previous attempts [Non 00a] [Non00b]. Robustness of software behavior and correctness of the reconstruction were never a problem, and no errors were detected in any of the tests. The consistency of the program is probably a result of the fact that most tests to build the reconstructed volume and surface are tolopogycal tests rather than geometric, as opposed to many of the other triangulation strategies in the field. Interaction with the model formed is perfectly feasible in real time, and exploration using a conventional virtual world modeling language such as VRML is quite feasible. In fact, the models were exported to VRML using built-in exporters in VTK, and they allowed the visualization of internal structures of the surfaces, in real time.

Figure 8 shows some of the results from applying the implemented direct volume rendering (DVRT) on the sampled contours. The data set was sampled with the resolution 306x174x83. The figures on the left column present the results without shading, while on the left column they show the combined use of shading parameters and transparency. As the technique does not build intermediate structures, shading is produced by the estimation of gradients. However, the visual results are quite effective, as can be seen in figures 8(b), 8(d), 8(f) and 8(h). It can be noted that shading is better defined in side views. This is due to the fact that, for this specific data set, gradients are better defined in that view direction than in the top or bottom views. Both choices (shading off or on) present good results in terms of smoothing the data. While these images cannot be explored at the moment using virtual environment technology, it does offer a good representation of the interior of the structures represented by the reconstruction. The combination of both techniques, which is a goal of the Dentistry Project in its next steps, is promising as far as data analysis and understanding is concerned.

In the next section we discuss these and some other aspects of this work and its follow-ups.

5: Conclusions and Further work

Visualization has met with a series of challenges all through its lifetime, from handling large amounts of data to producing images that the user could rely on. Some problems we have been dealing with have to do with providing consistent yet useful models, which will allow for interaction and simulation of dental procedures.

The use of both direct volume rendering techniques (and in particular DVRT), as well as surface reconstruction is a target, due to the fact that both complement each other very well as far as data analysis is concerned. In the case of dentistry, this work has verified this fact. The ability to see volume interior brings out aspects that the surface methods cut off.

There are problems, though. In an ideal situation it should be possible to combine both techniques and be able to realize operations that depended on either during interaction in a virtual environment. One of the limitations to achieve that is an efficient data structure that enables indexing of all the information needed in either method. This is a problem we are working with. Interaction with dental structures in a virtual environment is also a target, and so is to allow access to the system over the internet.

The use of DVRT together with surface and volume reconstruction in the same display and interaction environment will hopefully concentrate the distinct characteristics of both of them simultaneously during handling of the tooth and mouth models.

6: Acknowledgements

We wish to acknowledge Mike Goetz and Andy Day, from the School of Information Systems, UEA, UK, for the data used to generate most of the models presented here. We also wish to acknowledge the funding of FAPESP and CNPq, Brazil, and the work of our undergraduate students Carlos Frederico Rocha and Luis Fernando Chavier, of ICMC-USP.

7: References

[Boi88] J-D. Boissonnat, "Shape reconstruction from planar cross sections", *Computer Vision, Graphics and Image Processing*, (44), 1-29 (1988).

[Cab94] B. Cabral, N. Cam, J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware", *Proc. 1994 Symposium on Volume Visualization*; pp.91-98 (1994).

[For94] S. Fortune, "Voronoi Diagrams and Delaunay Triangulation", in D.-Z. Du, F K. Hwang (eds.), *Computing in Euclidean Geometry*, Wor ld Scientific Pub. Co., 193-233 (1994).

[Gel96] A. V. Gelder, K. Kim, "Direct Volume Rendering with Shading via Three-Dimensional Textures", University of California, Santa Cruz, CA 95064 USA; Technical Report UCSC-CRL-96-16 (1996); also in *Proc. ACM/IEEE Synposium on Volume Visualization*, San Francisco, (1996).

[Goe98] M. R. Goetz, A. M. Day, "Surface Reconstruction for Teeth", in *Proceedings of EUROGRAPHICS UK*, 16 th Annual Conference, 25-27 March, Leeds , UK, (1998).

[Grz98] R. Grzeszczuk, C. Henn , R. Yagel, "Advanced Geometric Techniques for Ray Casting Volumes", *ACM SIGGRAPH'98; Course Notes 04*; (1998).

[Kau98] A. Kaufman, "Advances in Volume Visualization", *SIGGRAPH'98 Course Notes no. 24*, Orlando, USA, 1998.

[Lop99] H. Lopes, L.G. Nonato, S. Pesco, G. Tavares, "Dealing with Topological Singularities on Volumetric Reconstruction", submitted to Curves and Surfaces, Vanderbilt University Press.

[Lor87] W. E. Lorensen, H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithms", *ACM SIG. Comp. Graph.*, (21), 163-169 (1987).

[Mys97] K. Myszkowski, G. Okuneva, J. Herder, T. L. Kunii, T.L, M. Ibusuki, "Visual Simulation of the Chewing Process for Dentistry", in *Visualization and Modelling*, Academic Press, 419-438 (1997).

[Non98] L.G. Nonato. *Volumetric Manifold Reconstruction from Planar Sections*, Ph.D. Thesis (in Portuguese), Mathematics Department, Pontifical Catholic University, Rio de Janeiro, Brazil, (1998).

[Non00a] L. G. Nonato, R. Minghim, M. H. Shimabukuro, : "Qualitative Analysis of Reconstruction Techniques for Dentistry", accepted for publication in the *Journal of Electronic Imaging* (2000).

[Non00b] L.G. Nonato, R. Minghim; M.C.F. Oliveira, "Discussing Different Approaches for Delaunay 3D Reconstruction in the Light of Applications in Dentistry", submitted to *Pacific Graphics'2000*.

[Sch98] W.J. Schröeder, K. Martin, W. Lorensen, *The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics*, 2nd. ed., Prentice-Hall, (1998).

[Sei95] S. Seipel, I. Wagner, S. Koch, W. Scheneider, "Three-dimensional visualization of the mandible: A new method for presenting the periodontal status and diseases". *Comput. Meth. Programs Biomed.*, (46), 51-57 (1995).

[Shi98] M.H. Shimabukuro, R. Minghim; P. Licciardi, "Visualisation and Reconstruction in Dentistry", in *Proc. Int. Conf. on Information Visualisation IV'98*, London, UK, IEEE CS Press, 25-31 (1998).

[Tes95] Teschner, M.; Henn, C. - Texture Mapping in Technical, Scientific and Engineering Visualization; SGI, Aug. 4, 1995; also in ACM SIGGRAPH'98; Course Notes 17; Abr. 20, 1998

[Gei93] B.Geiger. "Three dimensional modeling of human organs and its applications to diagnosis an surgical planning", Technical Report 2105, INRIA Sophia-Antipolis, France,1993.

Figures

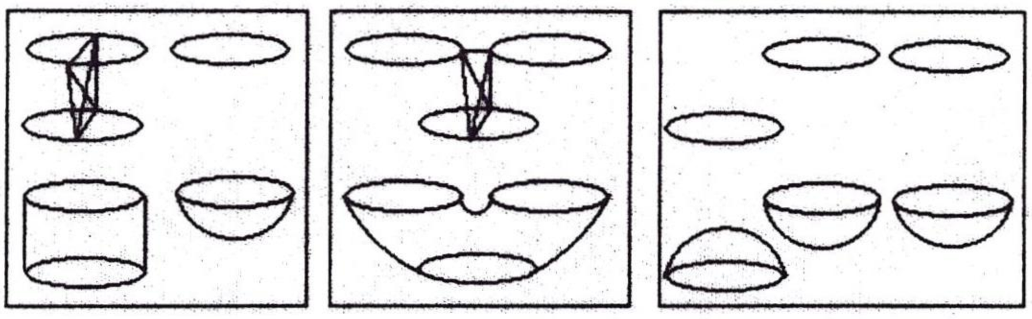
(a) (b) (c)

Figure 1: Decision cases of connected components using reverse tetrahedrons.

(a) two contours are connected.

(b) a single connected component is formed from the three contours.

(d) there are no reverse tetrahedrons. Each contour generates a separate component.

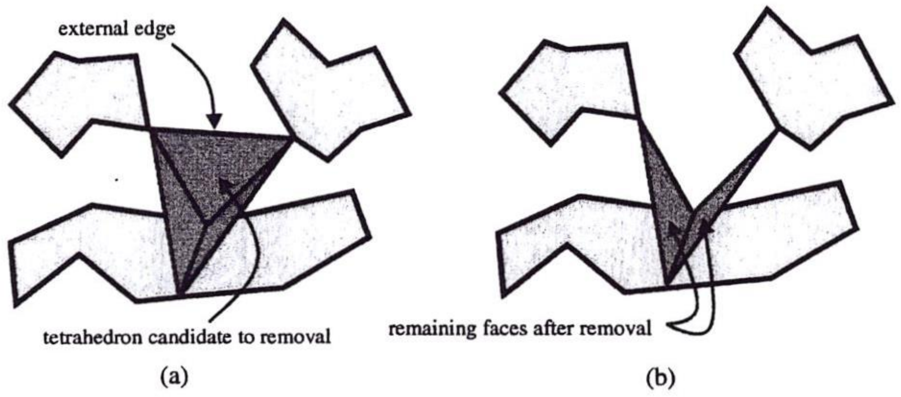
external edge
tetrahedron candidate to removal
remaining faces after removal
(a)
(b)

Figure 2: (a) external edge in the triangulation on the sampling plane

(b) singularity generated if the tetrahedron is simply removed.

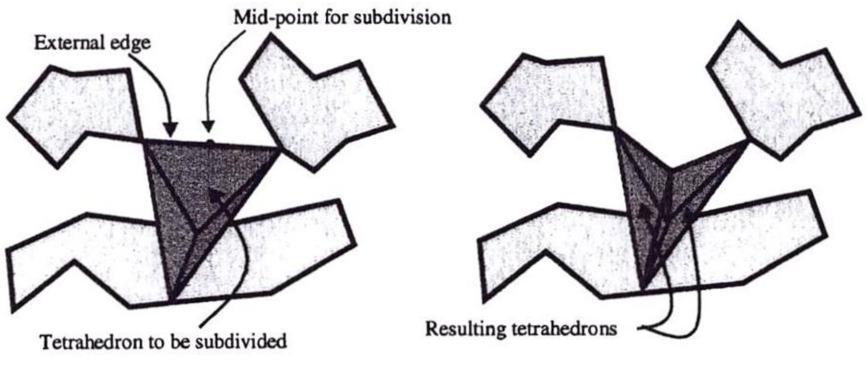

Figure 3: Tetrahedron subdivision. Sampled contours are undisturbed and singularity is properly dealt with.

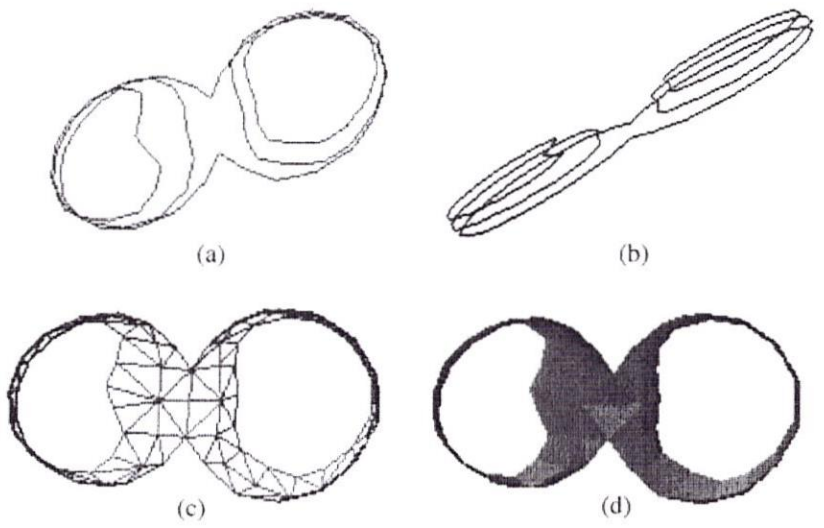

Figure 4: Reconstruction of singularity using the implemented DT-based algorithm. (a) and (b) two views of the contour set. (c) the resulting mesh (d) rendering of the mesh in 4(d).

```
#include "vtk.h"

vtkRenderWindow       *renderwindow;
vtkRenderWindowInteractor *renderwindowinteractor;
vtkRenderer              *renderer;
vtkStructuredPoints      *structuredpoints;
vtkPiecewiseFunction      *piecewisefunction;
vtkColorTransferFunction   *colortransferfunction;
vtkVolumeTexture       *volumetexture;
vtkVolumeTextureMapper      *volumetexturemapper;
vtkVolume              *volume;

volumetexture->SetVolumeTexture(structuredpoints);
volumetexture->SetColor(colortransferfunction);
volumetexture->SetOpacity(piecewisefunction);

volumetexturemapper->SetVolumeTexture(volumetexture);

volume->SetVolumeMapper(volumetexturemapper);

ren->AddVolume(volume);
ren->GetActiveCamera()->ParallelProjectionOn();
renderer->GetActiveCamera()->SetParallelScale(x);
renderwindow->AddRenderer(renderer);
renderwindowinteractor->SetRenderWindow(renderWindow);
```

Figure 5: Structure of a C++ program for DVRT in VTK.

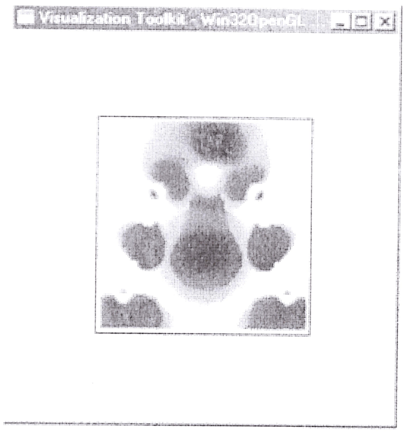

Figure 6: Image generated with DVRT without shading, with five sampling planes.

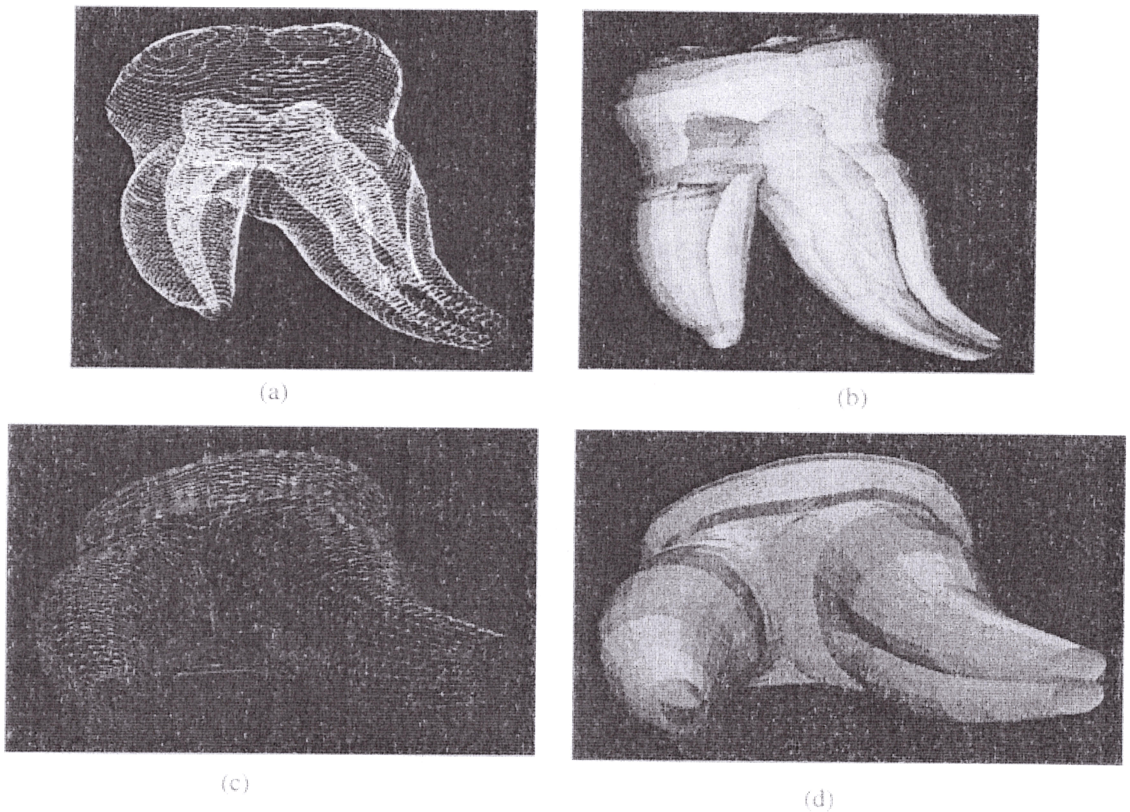

Figure7: (a) set of contours from tooth data; (b) reconstruction by Nonato's algorithm, with two nested surfaces, (c) resulting mesh from the reconstruction, bottom view; (d) rendering of the mesh in 7(c).

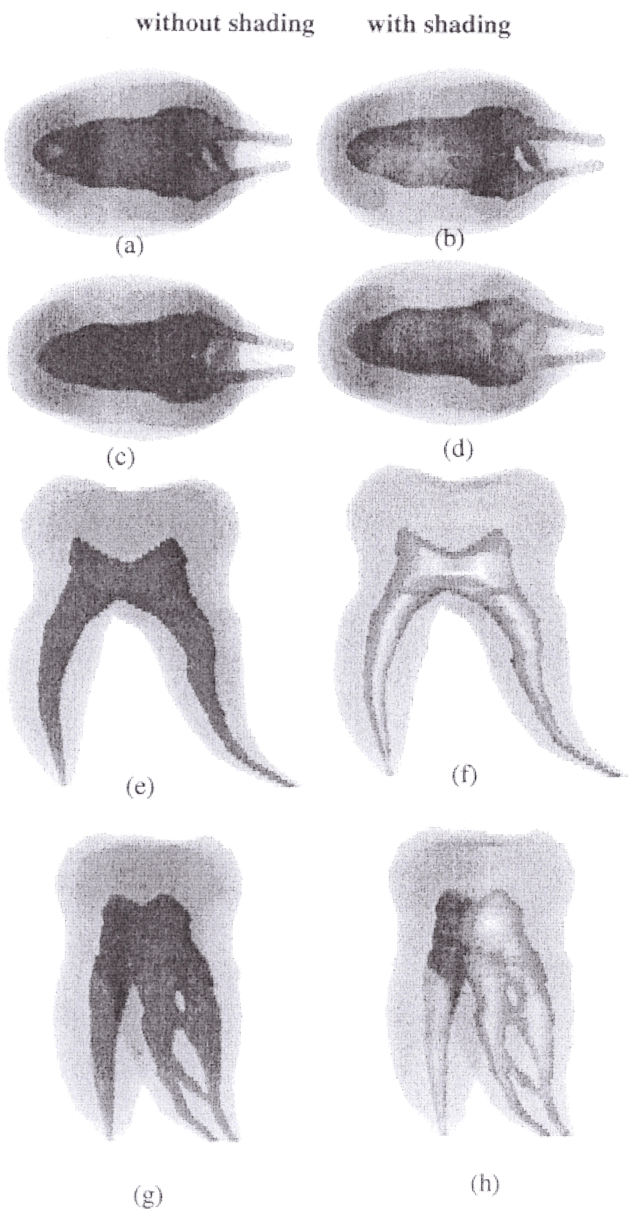

Figure 8: DVRT of tooth data.
(a) and (b) bottom view; (c) and (d) top view;
(e) and (f) side view; (g) and (h) rotated side view.