Accurate Measurements in Volume Data

Javier Oliván, Marco Bosma and Jaap Smit

University of Twente, Department of Electrical Engineering
PO Box 217, 7500 AE Enschede, The Netherlands

ABSTRACT

An algorithm for very accurate visualization of an iso-surface in a 3D medical dataset has been developed in the past few years. This technique is extended in this paper to several kinds of measurements in which exact geometric information of a selected iso-surface is used to derive volume, length, curvature, connectivity and similar geometric information from an object of interest.
The actual measurement tool described in this paper is fully interactive. The highly accurate iso-surface volume-rendering algorithm is used to describe the actual measurement that should be performed. For instance, objects for which volumes should be calculated, or paths from which the length should be calculated can be selected at sub-voxel resolution. Ratios of these quantities can be used to automatically detect anomalies in the human body with a high degree of confidence.
The actual measurement tool uses a polygon-based algorithm that can distinguish object connectivity at sub-voxel resolution. in exactly the same manner as the iso-surface algorithm. Segmentation based on iso-surfaces geometrical topology can be done at this point.
The combination of the iso-surface volume-rendering algorithm and the polygon-based algorithm makes it possible to achieve both visual interaction with the dataset and highly accurate measurements. We believe that the proposed method contributes to the integration of visual and geometric information and is helpful in clinical diagnosis.

Keywords: measurements, volume, length, medical data, iso-surface

1. INTRODUCTION

In this paper, we are going to present the ongoing research on the development of a software toolkit to visualize and to extract geometrical information from medical data sets obtained from CT or MRI scanners. This software represents a useful and efficient combination of different techniques. Some of these are voxel-based techniques, and polygon based ones. The visualization process will be done by using a ray casting algorithm that will be a voxel-based algorithm. The measurements will be done mainly in the triangle space. An effective interface between the two spaces will be required in order to achieve the maximum benefit from the combination of the techniques.

The acquisition of geometric information from medical data can play a very important role in a wide range of clinical applications. Several approaches can be found in the literature to measure different magnitudes of interest. In most of the cases, a dedicated algorithm is developed for each different case, even when the magnitude is the same [3,7]. Usually, an algorithm to perform volume calculations of the brain is different than an algorithm to measure the volume of the heart. In our case, our goal is to develop one unique algorithm for each geometric magnitude: an algorithm valid for the same magnitude measured in different regions of the human body. The concept of an iso-surface plays a very important role at this point.

The complete application is based on the concept of an iso-surface. An iso-surface in a continuous three dimensional field F(x,y,z) is defined as the set of three dimensional points for which the equation F(x,y,z)=Iso-value is satisfied (where Iso-value is a certain pre-defined value). In the case of 3d medical datasets obtained with CT or MRI scanners, the data consists of a three-dimensional set of scalar values. This set can be described as a discrete function D(i,j,k), where i, j, k are integer numbers. This discrete data field is the result of convolution of the point spread function of the acquisition device with an object at discrete sampling locations. As pointed by Bosma [1], a re-sampling function R(x,y,z), that might be interpolating or approximating, can be used to compute the values of the discrete data field at arbitrary (x,y,z) locations. Then, this re-sampling function extends the discrete data field to a continuous data field C(x,y,z). We will define the iso-surface in the discrete field D(i,j,k) as the iso-surface in this continuous data field C(x,y,z).

Visualization, Display, and Image-Guided Procedures, Seong Ki Mun, Editor,
Proceedings of SPIE Vol. 4319 (2001) © 2001 SPIE · 1605-7422/01/$15.00

In order to perform a measurement, we will extract one iso-surface from the region of interest, and then we will apply our algorithms. For instance, if we want to measure the volume of the heart, we will extract one iso-surface that describes the geometry of the heart, and then, we will compute the volume enclosed by that iso-surface. The iso-surface that describes a brain is different from the one belonging to the heart. In some situations, we will probably need two different algorithms to compute the two iso-surfaces. But once we have the two iso-surfaces, the same algorithm will be used to estimate the volume.

2. ISO-SURFACE VOLUME RENDERING

A highly efficient implementation of the iso-surface volume-rendering algorithm introduced by Bosma [2] is used in our tool to visualize the iso-surfaces in the data set. With the current implementation, up to 7 frames per second can be achieved in the visualization process, in a 256 by 256 by 109 dataset, in a 600 MHz Pentium III, with very high quality images almost free of artifacts. The reconstruction filter used is the tri-linear interpolator. This implementation provides us with a very powerful mechanism to visually interact with the data set. Other approaches, such as rendering a mesh, will not give us such a good ratio between quality and speed in low cost workstations. Figure 1 shows an example of the iso-surface volume rendering algorithm.

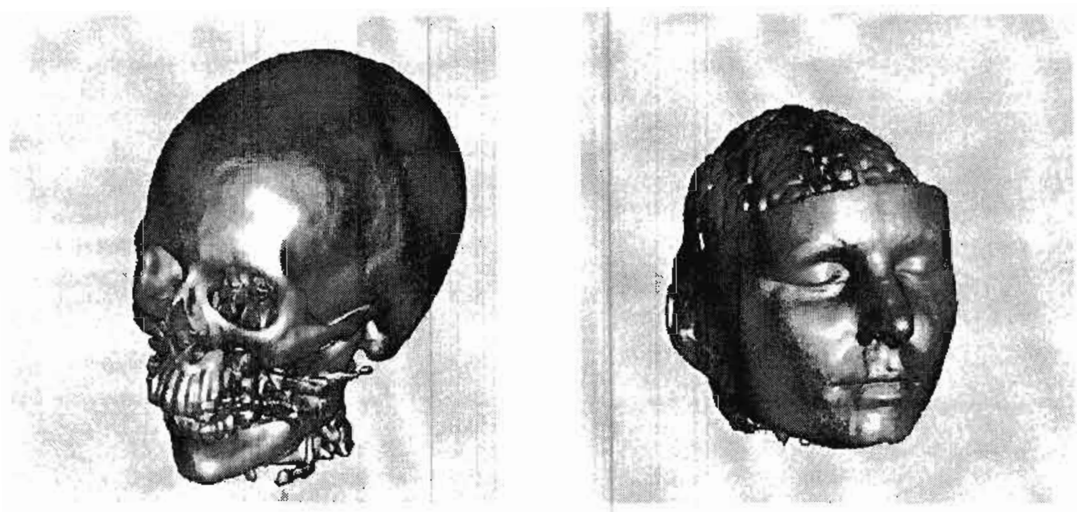

Figure 1. Iso-surfaces rendered with the algorithm introduced by Bosma.

3. THE TRIANGLE MESH

Several methods in order to extract a triangle mesh from a data set already exist in the literature [6]. In our first attempt to perform a vectorization of the iso-surfaces, a fast implementation of the standard marching cubes algorithm [5] has been used. Despite some drawbacks of this algorithm, like the presence of ambiguous configurations [4], we decided to use it because of its simplicity. Another characteristic of this algorithm is that marching cubes makes use of the linear interpolator to compute the vertices of the triangles. This is consistent with our iso-surface render algorithm implementation, that makes use of the linear interpolator to reconstruct the continuous field. Moreover, it is not difficult to implement and it is fast. In the current implementation, it takes 0.25 seconds to compute marching cubes for a 256 by 256 by 109 eight bit data set, in a 600 MHz Pentium III, so interactive speed is feasible when changing the iso-value. This timing information corresponds to the iso-surface shown in the right picture of Figure 1. A last benefit is that the measurement algorithms are independent of the algorithm used to extract the triangle mesh. This allows us to select between different triangularization algorithms without changing the rest of the application. In the future, different techniques to extract a triangle mesh from the data set will be tested.

4. COMBINING THE TECHNIQUES

As said before, the visualization tool we use is a voxel-based application, and the measurement tool is a triangle-bas application. The data input for the visualization tool is a three dimensional array, and the data input for the measurement to is a collection of triangle meshes. We obtain one triangle mesh for each connected iso-surface. By visual inspection, obje presented in the output image of the iso-surface volume-rendering algorithm are chosen by the human expert. To perform measurements on one object present in this image, the human expert selects one pixel in the image. This pixel is part of rendering of one particular iso-surface. Now, the program needs to know the triangle mesh that belongs to the object select in this manner. In order to do so, we render the complete polygon mesh once, without any lighting, and using one color non-connected polygon mesh. In this process, we will use the same transformation matrices that we used in the visualizati process. This rendering is done in a window of the same size as the window of the voxel space rendering, so we have a one one correspondence between the pixels of the two images.

One of the outputs of this triangle based rendering process is an image that will not be visible for the user. This image can seen as a color map: the possible different colors of the pixels of this image will be the colors we used to render the mesh of different objects. So, just by reading the color of the pixel we have selected in the visualization tool output image, program knows on which mesh we are going to perform the measurements.

The other output of the triangle-based algorithm is a depth buffer. This depth buffer can be combined with the depth buffer the voxel-based algorithm to study the spatial differences between the two different representations of the same iso-surface.

Another interface between the measure tool and the visualization tool is possible as well. This communication can established through a binary shell. A binary shell is a set of bits, one per cell in the discrete data field. The binary sh provides us with an efficient mechanism to select cells in the voxel space. Usually, the binary shell is used to mark the ce in the voxel space that are crossed by an iso-surface. The usage of a binary shell is one of the main optimizations in o visualization tool. Normally, the binary shell is calculated through comparing each voxel of the cell with the iso-value: if voxels are above or below the iso-value, the associated bit will be 0; if there are voxels above and below, the bit will be Although it is possible to compute the binary shell for a given data set and a given iso-value, we can also compute it fron triangle mesh. For instance, we can be interested in visualizing only one of the multiple disconnected iso-surfaces that we c have in one data set for a given iso-value. In this case, the visualization tool needs the portion of the binary shell belonging that particular mesh. This binary shell can be computed just by analyzing the intersection of the triangles of the object interest with the cells. In our implementation, marching cubes was the algorithm applied to generate the triangles. In th particular case, every triangle belongs to one and only one voxel cell. So we can generate a binary volume for the object interest just by setting a bit for the voxel cell of every triangle in the mesh of the object.

This approach can be used to do a segmentation of the data at sub-voxel resolution, based on connection between is surfaces. Actually, when we have several iso-surfaces crossing the same voxel, we can't segment the data if we compute binary shell comparing the voxels with the iso-value. But with the help of the triangle meshes, we can generate one bin shell for each non-connected iso-surface. Combining these binary shells, we can know not only the cells crossed by the is surfaces, but also which cells are crossed by which iso-surfaces.

5. VOLUME COMPUTATION

The volume estimation algorithm is a voxel-based algorithm. The current algorithm is very simple, and it was designed testing purposes.

We are interested in the value of the volume enclosed by an iso-surface. The method starts with the computation of th binary volumes: the so-called above binary volume, the below binary volume and the iso binary volume. A bit in the ab binary volume is set to one if the eight voxels of the cell are above the iso-value. This means that that cell is complet inside the iso-surface. A bit equals to one in the below binary volume imply that the eight voxels of the cell are below the i value, so the cell is completely outside the iso-surface. The iso-binary volume is the normal binary shell: a bit set in the binary volume reflects the fact that this cell is partially inside the iso-surface and partially outside. We define the ab volume as the number of bits set in the above binary volume. In a similar way, we define the below volume and the volume as the number of bits in the below and iso binary volumes.

With this information we obtain a first approximation of the volume. Assuming that the volume of each cell is one, volume enclosed by the iso-surface will lie between the number of cells that are completely inside the iso-surface and plus the number of cells that are partially inside the iso-surface. Therefore, the volume is between the above volume um volume) and the above volume plus the iso volume (above volume). This affirmation is error free, and is one of n advantages of the algorithm.

ference between this two values is equal to the iso volume, and is usually too big. We can get much better results just lividing the cells that are set in the iso binary volume. With a re-sampling function, we can get a finer grid inside these nd we can apply the same algorithm.

urrent implementation, we subdivide each cell selected in the iso binary volume in 31 * 31 * 31 cells, so for each cell ne iso binary volume, we get 32 by 32 by 32 new voxels. We have chosen this number because it gives a very good tween accuracy and performance for computers with 32 bit registers.

urrent implementation of the volume estimation algorithm, the re-sampling function used to oversample the data was ear interpolation function. Higher order interpolators can be used if they do not introduce overshooting. Overshooting if, when applying the reconstruction filter to compute the value of the continuous field inside a cell, we can get values he cell that are above or below all the eight voxels of the cell. If this is the case, we can have eight voxels greater then -value, but it is possible that this cell is crossed by the iso-surface. The cubic-spline interpolation function is an e of such kind of re-sampling filters that present overshooting.

st results we have obtained applying the algorithm on different medical data sets show that the difference between the m and the maximum value of the volume enclosed by the iso-surfaces are typically between 1 -5 % of the minimum But a more exhaustive test has to be done, including a test comparing our results with another approaches. Table 1 some volume measurements performed with the volume estimation algorithm.

Data set	Iso-value	Min volume	Max volume	Difference	Error (%)
MR brain	62	944687.12	955833.62	11146.5	1.18
MR brain	94	340603.62	348851.46	8247.84	2.42
MR brain	125	61246.67	63156.44	1909.77	3.12
CT head	300	2648969.50	2664086.25	15116.75	0.57
CT head	400	2313747.75	2324807.50	11059.75	0.48
CT head	500	2164199.50	2173077.50	8878	0.41

Table 1. Results of the estimation algorithm.

6. PATH LENGTH COMPUTATION

llows us to have an estimation of the minimum path length between two points over an iso-surface. The selection of oints can be done in the iso-surface volume-rendering tool. Using the triangle mesh, we select two triangles. The ecognizes whether these two triangles belong to the same object, i.e., the same iso-surface. If this is the case, then it ssible to find a minimum length path over the iso-surface. Once we have selected the two triangles in the mesh to the object we are visualizing, the program builds a connected graph using points in the polygon mesh. In a first ation, only vertices of the triangles were used. In this case, the nodes of the graph are the vertices of the triangles, s are the edges of the triangles. The accuracy obtained with this approach was not satisfactory, so we added extra ese extra points were the center points of the triangles and the middle points of the edges of the triangles. With this ach, much better results are achieved. Figure 2 shows the two different graphs for the same triangle mesh, in this plane iso-surface.

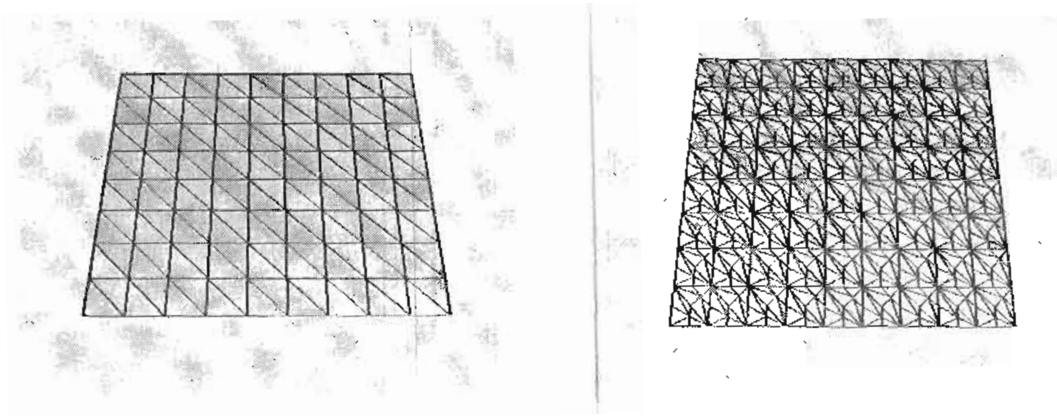

Figure 2. Two different graphs extracted from the same triangle mesh.

Once the program has created the connected graph, we apply the Dijkstra algorithm [9] to find a set made out of nodes of the graph that gives us the minimum length path over the triangle mesh. The length of the minimum path founded in the polygon mesh will be an approximation of the minimum distance between the two points over the iso-surface selected in the iso-surface volume-rendering tool. In Figure 3, two different paths are shown in order to remark the differences between the paths belonging to different graphs.

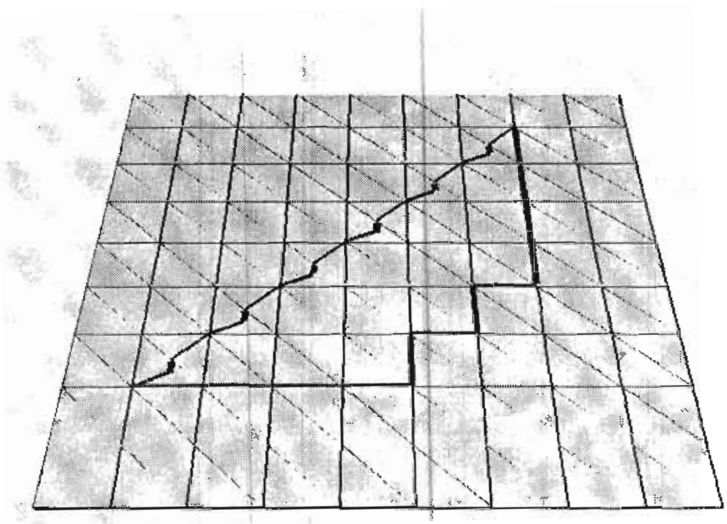

Figure 3. Two paths obtained with two different graphs.

There are several points to discuss about the accuracy of this method. First of all, we are interested in calculating the distance between two points over the surface of the real objects: the heart of a patient, the colon, a bone, etc. But we are computing lengths over a polygon mesh. This algorithm is independent of the method used to extract the polygon mesh, so we will not discuss here how good is the generated iso-surface. A detailed description of different iso-surface extraction algorithms and some topological considerations can be found in a paper from van Gelder and Wilhelms [8]. Once we have a polygon mesh, we want to compute the distance over that mesh between two points. In a plane, the shortest distance between two points is the length of the straight line that joins the two points.

To test the algorithm, we used the following approach. First, we generated a triangle mesh of a plane square. We compute the length between two vertices belonging to this mesh using our algorithm. For this particular case, we know the exact value of the length: because we have a plane, the minimum path between two points of this plane is a straight line, so the minimum distance between these two points is the length of the line that joins them. We can compare both values to calculate the error we make

The two different kinds of graphs were tested. First, we generated a graph from the mesh. Then, we computed the between the center of the square and points inscribed in a circumference finding the minimum path. We also comp real distance between the two points. Figure 5 is a graph of the error in the length estimation versus the angle betw path and the bottom edge of the plane. As we can observe, we have an anisotropy in the error: for some angles, the much higher than for other ones. This is due to the fact that, when applying marching cubes to a data set that present curvature, we get a very regular mesh. In this case, we obtain also a very uniform graph (see Figure 4). This uniformit reason for the anisotropy in the error.

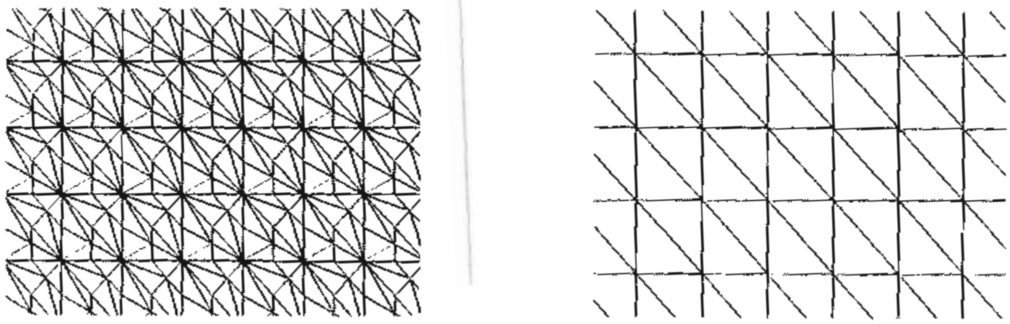

Figure 4. Uniformity in the graphs extracted from a plane mesh.

In a real situation, we usually have non-uniform meshes. So we will take the average error in the Figure 5 as an estimat the error we have in normal cases. In the case of simple graphs, i.e., graphs made only with vertices of triangles, the av error is 16.3 %, while in the graph made taking more points from the polygon mesh (the centers of the triangles an middle edge points), the average error is 5.9 %.

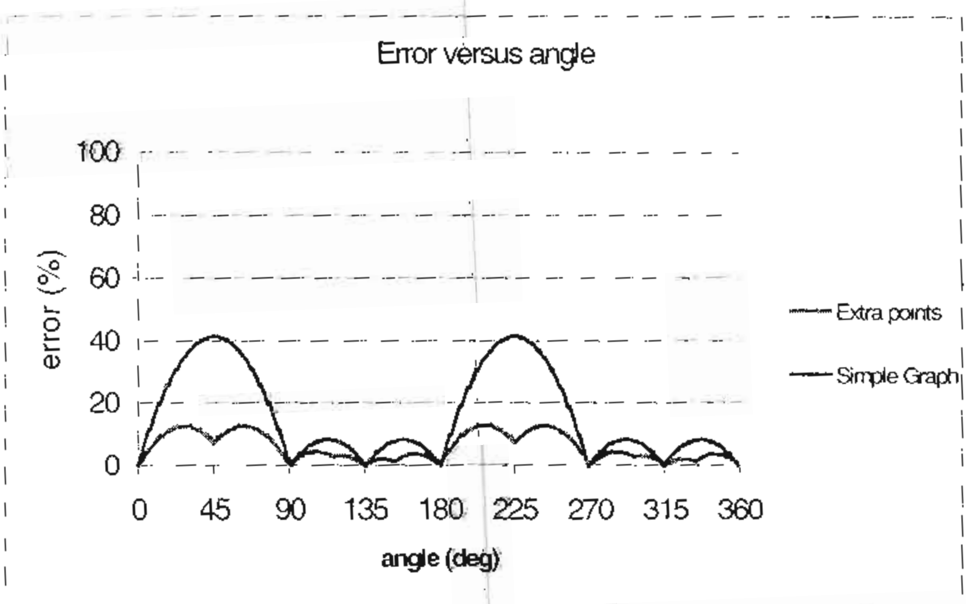

Figure 5. Length error versus angle in two graphs of a uniform triangle mesh.

Once we have a set of three-dimensional points describing the minimum length path, we can use this as input for an algorithm based on deformable models in order to minimize the real length. In our algorithm, we have a very impo limitation: we have to create a graph based on points belonging to the mesh. We can overcome this limitation in fu algorithms based on active models or snakes [10], and modeling the snake with forces based on the gradient o continuous field.

Figure 6 (a) is the output image we get with the render algorithm for a data set of a head. The image in Figure 6 (b) show two paths we obtain between two points belonging to the skull. In Figure 6 (b), the rendering was done in the triangle s using openGl, to show the differences in the quality of the two render techniques: iso-surface volume rendering and tri mesh rendering.

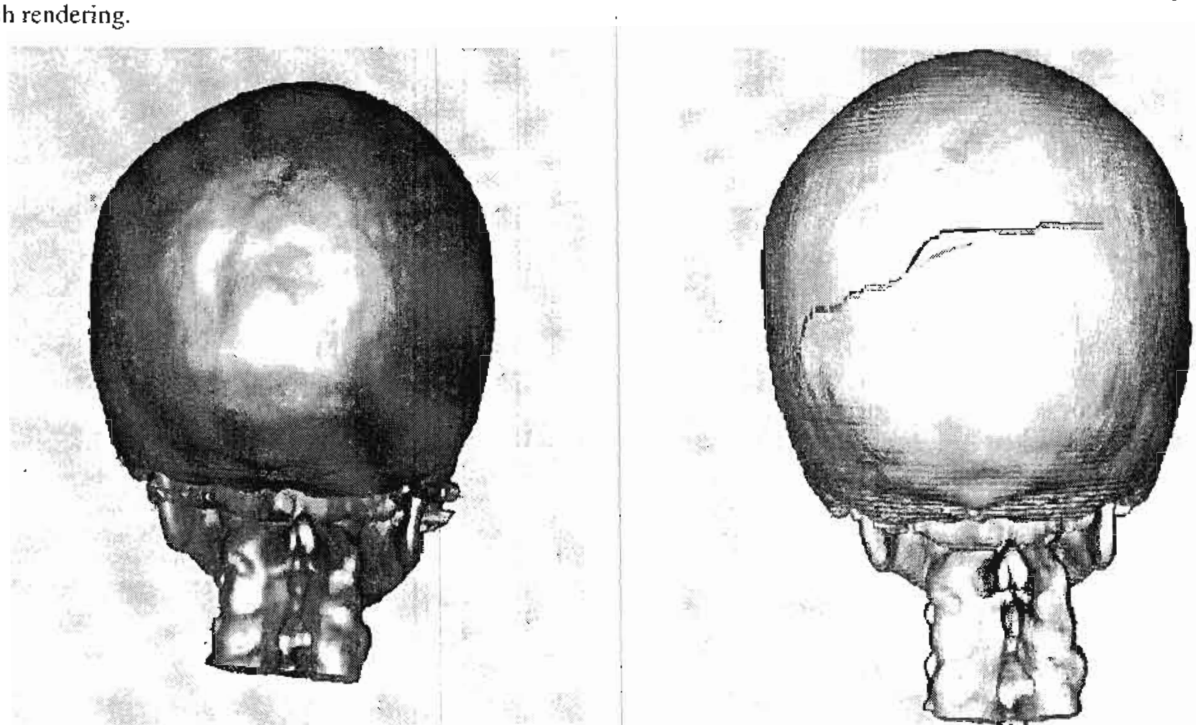

Figure 6. (a) Skull rendered with the visualization tool and (b) two paths over the skull computed over two different graphs.

Although some improvements in the minimum distance estimation algorithm will be implemented in the future, we hav algorithm that provides us with a path between two points. Maybe this path is not optimal yet for distance calculations, h can be used to track automatically certain regions of the data set. Just as an example, in Figure 7 we show two paths obta with our algorithm in a colon data set. Once we have the connected three-dimensional points, the camera can be locate that points and we can render the interior of the colon.

Figure 7. Two paths over the colon surface.

7. DISCUSSION AND FURTHER RESEARCH

In this paper, we have described our methods to combine an efficient visualization algorithm and several measurement acquisition algorithms. The visualization process is done in the voxel space, whereas the measurements are performed in the triangle space. With this approach, fast rendering in a low cost workstation and geometrical information extraction can be achieved at once

Both the visualization and the measurement process are based on the concept of an iso-surface. We render iso-surfaces, and we extract information from them. We can apply the same measurement algorithms for iso-surfaces representing different regions of the human body

The three basic parts of the software (iso-surface rendering, extraction of triangle meshes and measurements acquisition), are independent of each other. They communicate through a well-defined interface. Keeping this interface unchanged, we can modify one part of the software without changing the rest of the application.

8. REFERENCES

1. M. Bosma (2000), Iso-surface volume rendering – Speed and accuracy for medical applications, PhD Thesis, University of Twente, ISBN 90-365-1397-5

2. M.Bosma, J.Smit, and S. Lobregt (1998), "Iso-surface Volume Rendering", proc. SPIE-MI , vol 3335, Pages 10-19

3. Prasun Dastidar, Juhani Maenpää , Tomi Heinonen , Tapio Kuoppala , Milko Van Meer , Reijo Punnonen, and Erkki Laasonen (2000), "Magnetic resonance imaging based volume estimation of ovarian tumors: use of a segmentation and 3D reformation software", Computers in Biology and Medicine, Volume 30, Issue 6, Pages 329-340

4. C. Montani, R. Scateni, and R. Scopigno (1994), "A modified look-up table for implicit disambiguation of Marching Cubes", The Visual Computer, 10(6):353-355

5. Lorensen WE, Cline HE (1987) "Marching cubes: a high-resolution 3d surface construction algorithm", Computer Graphics, 21(4), Pages 163-169

6. P. Cignoni, F. Ganovelli, C. Montani, R. Scopigno (2000) "Reconstruction of topologically correct and adaptive trilinear iso-surfaces" , Computers & Graphics 24 ,Pages 399-418

7. Laurence P. Clarke, Robert P. Velthuizen, Matt Clark, Jorge Gaviria, Larry Hall, Dmitry Goldgof, Reed Murtagh, S. Phuphanich, Steven Brem (1998) "MRI measurement of brain tumor response: comparison of visual metric and automatic segmentation", Magnetic Resonance Imaging, Vol. 16, No. 3, Pages 271-279.

8. Allen van Gelder, Jane Wilhelms (1994) "Topological considerations in iso-s+-urface generation", UCSC-CRL-94-31

9. Dijkstra, E. W. (1959) "A Note on Two Problems in Connection with Graphs", Numeriche Mathematik, 1, Pages 269-271.

10. M. Kass, A. Witkin and D. Terzopoulos (1988) "Snakes: Active contour models", International Journal of Computer Vision, 1(4), Pages 321-331.