THE UNIVERSITY OF CALGARY

HIDDEN OBJECT RECONSTRUCTION FROM ACOUSTIC SLICES

BY

Emad N. Attia

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

December, 1996

Abstract

A method for reconstructing 3D models from cross-sectional seismic signals is developed. This method could be used in visualizing underground objects. The new algorithm starts by preprocessing the sonar images, first via thresholding then segmenting, contour finding, point sampling and finally triangulating. The main objective of this algorithm was to produce the smoothest possible model for branching bodies taking into consideration that the slices might be far apart in reality. Consequently a novel branching/interpolation technique was used that proved superior to existing methods and at the same time did not need involvement of the user at any stage. Seismic simulations have been used to test the algorithm and we succeeded in reconstructing the tomb of Tut Ankh Amen (An Egyptian Pharaoh) from simulated slices.

Acknowledgments

Special thanks are due to the following people: Thomas Simms and Tarek Al-baho who asked our lab for a solution to the 3D problem they are facing, Dr. Robert Stewart and Dr. John Bancroft of the Geology and Geophysics Department who contributed to this thesis by answering my numerous questions about seismic surveys and by allowing me to use their GX2 seismic simulation software and Dr. Saul Greenberg for adding to my knowledge concerning research methodologies. My deepest gratitude to my supervisor Professor Jim Parker who helped me to choose the topic and provided me with all the means to finish my thesis in such a short time. Thanks for the freedom you gave me.

Dedication

To Lily, with all my love.

Contents

Approval Page ii

Abstract iii

Acknowledgments iv

Dedication v

Table of contents vi

List of figures ix

Introduction 1

1.1 Motivation. 2

1.2 Goal. 2

1.3 Thesis structure. 3

Previous work 5

2.1 Reconstruction from sonar images. 6

2.2 3-D reconstruction from cross-sectional contours. 7

2.2.1 Optimal triangulation. 8

2.2.2 Heuristic triangulation. 11

2.2.3 Heuristic branching. 14

2.2.4 Delaunay triangulation. 16
2.2.5 G-Octree reconstruction from a series of G-Quad trees. 21
2.2.6 Syntactic/semantic approach. 23
2.3 Available software packages. 24
2.4 Evaluation of the state of art. 25
2.5 Criteria for the solution. 26
Seismic signals **27**
3.1 Different geometries. 28
3.2 Seismic data processing. 31
Preprocessing **35**
4.1 Introduction. 35
4.1.1 Thresholding. 36
4.1.2 Segmenting. 38
4.1.3 Contour finding. 40
4.1.4 Contour tracing. 42
4.1.5 Matching points. 43
4.1.6 Perspective projection of the resulting object(s). 43
4.1.7 Limitations of the primitive solution. 44
4.2 Results of the primitive solution. 45
A 3-D reconstruction solution **48**
5.1 Area threshold. 48
5.2 Contour mapping. 50
5.3 Triangulation. 52
5.4 Branching. 56
5.5 The algorithm. 60

5.6 CThead algorithm. 63
5.6.1 Removing the area feature. 63
5.6.2 Reconstructing a skull. 64
5.7 Interpolation. 72
5.8 Vertical Seismic Profiling (VSP). 78
Simulated slices **82**
6.1 GX2 Simulator and the SU seismic data processor. 82
6.2 Results of reconstructing 3-D seismic slices. 86
6.3 Tut Ankh Amen's Tomb. 90
Conclusions **102**
7.1 Primary goal and major contributions. 103
7.2 Future research avenues. 103
7.3 Final word. 104
Bibliography **105**

List of Figures

2.1 VSP arrangement. 7
2.2 Parallel arrangement. 7
2.3 Triangulation of 3 contours in 3D space. [Keppel 1975] 8
2.4 Partitioning Vertices Into Convex Sets And Concave Sets. [Keppel 1975] 10
2.5 One of the triangulations proposed for reconstruction. [Keppel 1975] 10
2.6 Each Pij represent a matrix that triangulates 2 convex sets. [Keppel 1975] 11
2.7 Original contours. [Christiansen 1978] 12
2.8 A step in the shortest diagonal algorithm. [Christiansen 1978] 12
2.9 The final triangulation between the 2 contours. [Christiansen 1978] 12
2.10 One step in the Ganapathy algorithm. [Ganapthy 1982] 14
2.11 Branching slices. [Christiansen 1978] 14
2.12 The result of the Christiansen algorithm. [Christiansen 1978] 15
2.13 A difficult case of branching slices. [Christiansen 1978] 15
2.14 Voronoi diagram and Delaunay triangulation. [Boissonnat 1988] 17
2.15 Three types of tetrahedra T1, T12 and T2. [Boissonnat 1988] 18
2.16 Projection of a triangle in one plane onto another plane. [Boissonnat 1988] 19
2.17 Result of the Delaunay triangulation of a branching body. 20

2.18 Representing the gray scale of each pixel in binary format. [Mao 1987] 21
2.19 The resulting image after one step of G_Quad tree construction. [Mao 1987] 22
2.20 The G_Quad tree after 1 step. [Mao 1987] 22
2.21 A 3D volume represented by some sort of voxels. [Mao 1987] 23
3.1 A zero offset arrangement. [Parker 1996b] 28
3.2 Common source arrangement. [Parker 1996b] 29
3.3 Parallel seismic lines with 5 pairs of source/receiver combinations. [Parker 1996b] 29
3.4 VSP arrangement. [Parker 1996b] 30
3.5 VSP arrangement with more than one slice. [Parker 1996b] 30
3.6 NMO geometry. 32
3.7 NMO correction. 33
3.8 Kirchhoff Migration. 34
4.1 A CT slice through the upper jaw and the lower part of the skull. 36
4.2 A threshold CT scan (Threshold value = 13). 37
4.3 Segmented image (PV = Pixel Value) 39
4.4 Contours of the segmented CT slice. 40
4.5 The use of different sampling steps. 42
4.6 matching points in a primitive way. 43
4.7 Branching slices. 44
4.8 Slices containing circles with different diameters. 46
4.9 Shaded reconstruction of the previous 3 slices. 47
4.10 The reconstructed mesh. 47
5.1 Noisy regions resulting from thresholding operation. Area threshold is 0. 49
5.2 Using an area threshold of 30 pixels removed all the noisy regions. 49
5.3 Circular artificial slice. 54
5.4 Another circular artificial slice at a far center of mass. 55
5.5 Correctly triangulated surface. 55
5.6 Correct triangulation. Shaded. 56

5.7 replacing the slice in figure 5.4 with this slice. 57
5.8 A shaded branching object (Top view). 58
5.9 Branching from 1 circle to 2 circles using the interpolation algorithm. 58
5.10 The triangular mesh of the object in figure 5.9. 59
5.11 Delaunay triangulation of a branching body. 60
5.12 Slice number 20 65
5.13 Slice number 40. Notice the grooves at the position of the eyes. 65
5.14 Slice number 60. 66
5.15 Slice number 80. Notice the noise at the top of the image. 66
5.16 slice number 100. 67
5.17 Hybrid algorithm (similarity threshold=0.8, area threshold=40). 68
5.18 Hybrid algorithm (similarity threshold=0.96, area threshold=40). 68
5.19 Improved Hybrid algorithm (similarity threshold=0.8, area threshold=40). 69
5.20 Improved Hybrid algorithm (similarity threshold=0.96, area threshold=80). 69
5.21 Improved Hybrid algorithm (similarity threshold=0.96, area threshold=40). 70
5.22 A Noisy CT slice. 71
5.23 The following slice in the CT scans series. 71
5.24 Irregular shape to be triangulated with a circular contour. 73
5.25 Without using interpolation, the model is not smooth. 74
5.26 Using interpolation. 74
5.27 Delaunay triangulation of the above two slices. 75
5.28 Irregular contours used to test interpolation while branching. 76
5.29 Without using interpolation, the model is not smooth enough. 76
5.30 Using interpolation enhanced the model smoothness. 77
5.31 The VSP triangulation method. 79
5.32 Radial cross-sections triangulated using the method described. 80
5.33 Delaunay triangulation of the two radial slices. 80
5.34 A Branching example. The angle between the two cross-sections is 30 degrees.... 81

5.35 Delaunay triangulation of the two radial slices while branching.....81
6.1 A GX2 trapezium model. 83
6.2 GX2 Zero offset traces produced for the trapezium model. 84
6.3 An SU gray scale image representing the traces produced. 85
6.4 A depth migrated image equivalent to the trapezium model. 86
6.5 3D Trapezium model. 87
6.6 Noisy signals from the trapezium model. 88
6.7 Migrated section. 88
6.8 Reconstructed model using a series of the section shown above. 89
6.9 Smoothed migrated section. 89
6.10 Reconstructed model from the smoothed section. 90
6.11 Different views for Tut's tomb. 91
6.12 Slice one. A slice through the entrance of the tomb at depth of 3 meters. 92
6.13 The bright spot represent the upper surface of the cavity in slice number 1. 92
6.14 Slice 2. Still at the entrance of the tomb but now at a depth of 3.75 meters. 93
6.15 Again the bright spot represent the upper surface of the tomb entrance. 93
6.16 Slice 3. The entrance is at a depth of 4.5 meters. 94
6.17 The corresponding migrated image for slice 3. 94
6.18 Slice 4. Tut's tomb.....95
6.19 Migration of slice 4. 95
6.20 Slice 5. Tut's tomb. 96
6.21 Migration of the signals produced by slice 5. 96
6.22 Slices 6 and 7. The sarcophagus chamber. 97
6.23 Migration of slices 6 and 7. 97
6.24 Slice 8. Tut's tomb. 98
6.25 Depth migrated image of slice number 8. 98
6.26 Slice 9. The annex to the left and the other compartment to the right. 99
6.27 A migrated image of slice 9. 99

6.28 Tut's tomb produced by 3DVIEWNIX. 100
6.29 A reconstruction of the tomb of Tut Ankh Amen. 101

Chapter 1 Introduction

"All the common propositions that result
from experience of the world are synthetic."
-- Immanuel Kant.

The way bats see is very interesting and shows how animals get accommodated to their environments and/or disabilities. Bats have a very weak sense of visual perception. By night, they send sound waves and they use their hearing sense to receive the echoes (reflections) of those waves. They might not receive any waves and they will conclude that there are no obstacles in their way. On the other hand, if they received the reflected waves shortly after sending them, then they will deduce that an obstacle is very close to them and they will take another route. The same idea is used in seismic surveys. Low frequency sound waves (10-100 Hz) are sent through the surface of the earth and reflected waves may show if there is anything of interest under the ground or not.

It is very clear from the above examples that our understanding of the world is

completely dependant on how we perceive it, not just by eyes but with all our senses, and this differs from one creature to another. The process from perception to understanding could be considered as a synthetic process, because each creature will analyze the input signals and produce a series of conclusions about the surrounding environment, and it is up to the creature trying to understand to make it accurate (realistic) or otherwise useful.

1.1 Motivation.

Visualizing underground objects has many applications in geology and archaeology. For example, if an archaeologist is searching for an underground object in a certain piece of land then he/she will have to dig and see for himself / herself. An easier way of doing that would be the use of seismic soundings, and from the signals of these soundings geophysicists will be able to know whether there is an underground object or not. An even better way is to visualize the underground object by reconstructing a 3-D model of those objects from the sonar slices. The purpose of this thesis is to explain a method for implementing such a solution. The process involves processing the sonar images and triangulating[1] the points obtained from those images.

1.2 Goal.

The goal of this thesis is to develop a computer system that is capable of reading cross-sectional sonar slices (simulated) and produce a 3-D model of the hidden underground

1. By triangulating we mean the process of connecting the sampled points in each slice to each other using a series of triangles thus constructing a mesh which represent the original 3-D shape.

object. Care must be taken such that the reconstructed model is as close as possible to the actual object. Since this is very hard to measure, because of the fact that the hidden object itself is unknown to us, sonar images of known objects have been developed so that the developed system could be efficiently tested.

The two criteria that must be taken into consideration are:

- A realistic object reconstruction.
- Fast reconstruction without user intervention.

More detailed requirements for satisfying those two criteria will be explained in the following chapter after a review of some of the most used algorithms and software packages for 3-D object reconstruction.

1.3 Thesis structure.

A survey of 3-D reconstruction techniques is detailed in Chapter 2. For each of the reviewed algorithms a detailed analysis of the algorithm itself is presented and the major advantages or pitfalls are mentioned. Also, two software packages are reviewed in this chapter. In Chapter 3, an introduction to seismic signals, seismic surveys and seismic data processing is given. This introduction is important so that the reader can have an appreciation of the long procedure needed to produce the seismic slices.

The preprocessing stage together with a primitive solution are presented in Chapter 4. The preprocessing involves many computer vision techniques to produce an image adequate for reconstruction. Chapter 5 explains in detail the novel 3-D reconstruction technique from seismic slices. The technique is tested using a series of CT scans (most widely used application for 3-D reconstruction from slices) of a human head. In Chapter

6, the algorithm is applied to simulated seismic slices (clean and noisy) and our system was used to reconstruct the tomb of the Egyptian pharaoh Tut Ankh Amen from those simulated slices. Chapter 7 concludes the thesis by evaluating the presented solution and suggesting further avenues of research.

Chapter 2 Previous work

Only in quiet waters do things mirror themselves undistorted.
Only in a quiet mind is adequate perception of the world.
-- Hans Margolius

Much work has been done previously in the area of 3-D reconstruction from slices and this work was mostly related to medical applications due to the advances in CT scans and NMR technologies. Most of the papers, such as [Keppel 1975] and [Ganapathy 1982], ignored the problem of having a branching body[1], while others, such as [Christiansen 1978] and [Boissonnat 1988], explained how to deal with branching bodies but their algorithms had several limitations as mentioned below in this chapter. Some software packages such as IMOD and 3DVIEWNIX[2] will be reviewed in the last

1. A branching body (or model) is when we have one contour at one slice and then 2 or more contours at the consecutive slice. Thus, a good reconstructing algorithm will connect this one contour to the other contours on the other slice yielding a branching model.
2. A state of the art 3-D reconstruction software package, for medical purposes, developed at the Radiology department in the University of Pennsylvania.

subsection. But first the sole application for 3-D reconstruction from sonar slices will be presented.

2.1 Reconstruction from sonar images.

Reconstruction from sonar slices have been dealt with before in an archaeological excavation in Israel [Witten 1995]. The excavators used the method of geophysical diffraction tomography originally presented in [Devaney 1984] to produce parallel planar slices and radial slices which were reconstructed using the EarthVision software package developed by Dynamic Graphics Inc.

In [Witten 1995] the VSP (Vertical Seismic Profiling - explained below in Section 3.1) system used an array of 32 water-coupled microphones with a spacing of 20 cm. These microphones were inserted inside a borehole. The sources were put radially from the borehole. Each line contained 32 geophone with a spacing of 30 cm (see figure 2.2). The obtained slices are not parallel to each other.

A similar arrangement was done to produce parallel cross-sections but the microphones and the geophones were put at the same line with a microphone at the midpoint between two geophones (see figure 2.2). A hammer is used to produce the sound wave and the geophones, which are connected to a data acquisition system that feeds a 486 PC with data, records the incoming signals within a pre-defined time window corresponding to a specific set of depths.

The produced cross-sections were fed into the EarthVision software which allowed the excavators to view the surfaces under the earth's surface [Witten 1995]. This application for 3-D reconstruction from acoustic slices was the only one found in the literature. In

the following subsection a detailed review of the different 3-D reconstruction algorithms is presented.

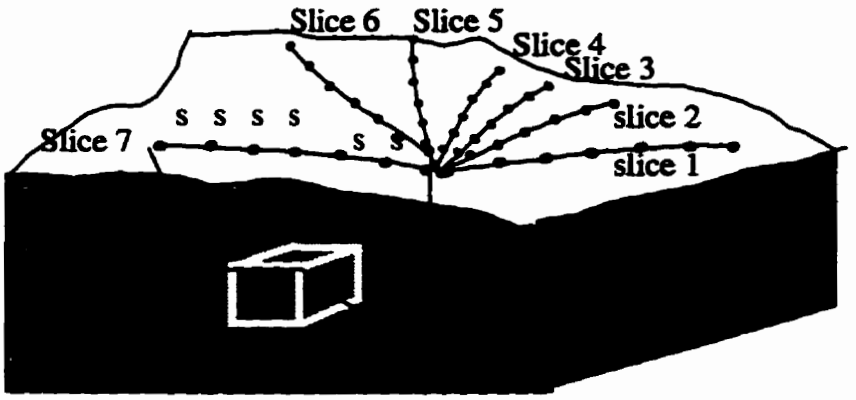

Figure 2.1: VSP arrangement.

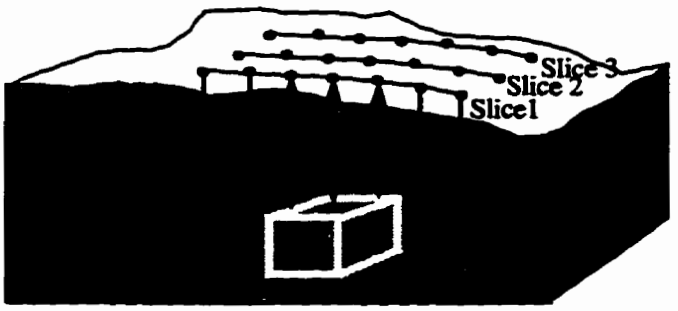

Figure 2.2: Parallel arrangement.

2.2 3-D reconstruction from cross-sectional contours.

A preprocessing stage is needed to produce contours for the objects in the image slices. After preprocessing a reconstruction stage should be performed The purpose of the reconstruction phase is to construct a surface between these contours. In figure 2.3, the three contours A, B and C are connected using triangles. The vertices of these triangles

are points taken on each of the three contours. Notice that as the number of points on the contours increases, and therefore so does the number of triangles, the reconstruction is better.

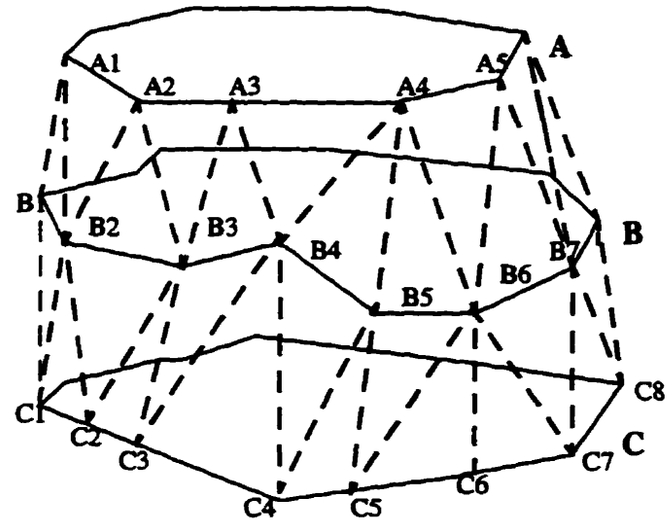

Figure 2.3: Triangulation of 3 contours in 3-D space. [Keppel 1975]

Some researchers developed optimal triangulation algorithms for 3-D reconstruction from planar cross-sections but, unfortunately, the algorithms are very slow. This led other researchers to investigate the possibility of finding a heuristic solution for the triangulation problem which will result in a near-to-optimal solution in a much shorter time while some others totally ignored the triangulation methodology and provided some other solutions such as G-Octree generation. The following subsections provide a survey for the different approaches to the problem of 3-D reconstruction from serial cross-sections.

2.2.1 Optimal triangulation.

E. Keppel is one of the pioneers in the field of 3-D reconstruction from planar cross

sections. He proposed an optimal solution for the problem in the mid seventies. The proposed procedure goes as follows: First, some contour points are defined on the boundaries of each contour. The number of these points depends on the accuracy level needed. That is, the more points used the better the triangulation process [Keppel 1975]. Second, the contour points are partitioned into concave sets and convex sets. In figure 2.4, which is borrowed from [Keppel 1975], the concave sets are {A3, A4, A5, A6, A7} and {A12, A14, A17, A18, A20} and the convex sets are {A12, A13, A14}, {A14, A15, A16, A17} and {A18, A19, A20}.

Keppel presented his view of an optimal algorithm for purely convex contours and he proposed that, in the concave case, the algorithm has a dual. For figure 2.5, which is borrowed from [Keppel 1975], Keppel stated that "The triangulation which maximizes the volume of the polyhedron A1 A2 A3,..., Am B1 B2 B3,....., Bn gives the optimal approximation of the surface provided by a pair of closed convex contour lines." [Keppel 1975]. In other words, maximizing the volume of the polyhedron will produce a surface which is the closest possible to the real world object.

In order to satisfy this condition of maximum volume of the reconstructed model a graph shown in figure 2.6, which is borrowed from [Keppel 1975], is proposed. This graph represents all the combinations of triangulation between any two points on the contours (i.e. all the solutions to the optimization problem). Each path in this graph from A1 B1 to Am Bn is considered a solution. A cost function is used to maximize the volume reconstructed and thus the path in the graph that gives maximum volume is selected to be the optimal triangulation.

The process is very costly since the process of partitioning the contour points into convex and concave sets is not a trivial task and the process of constructing the graph

and finding the optimal path is again not trivial. Also, this process is repeated for each set that was produced in the partitioning process.

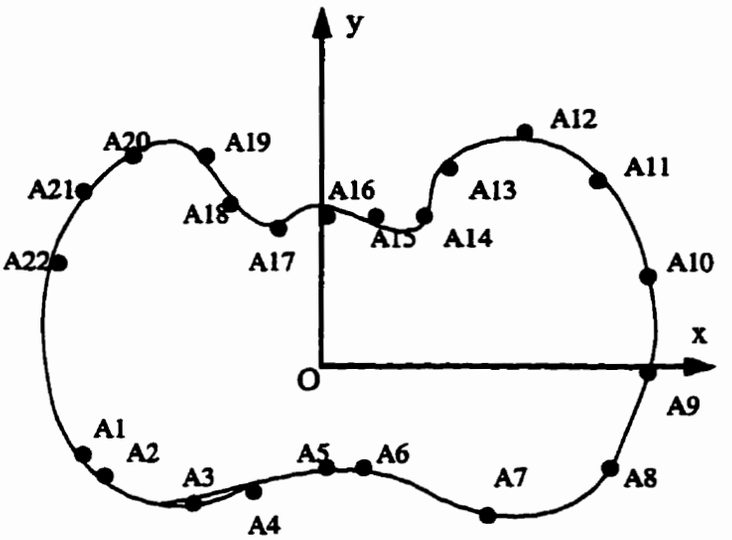

Figure 2.4: Partitioning Vertices Into Convex Sets And Concave Sets. [Keppel 1975]

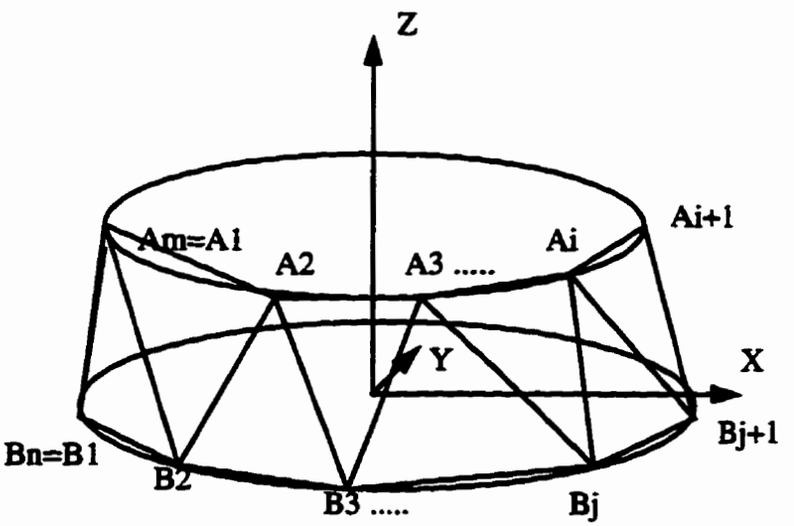

Figure 2.5: One of the triangulations proposed for reconstruction and the polyhedron represented by it. [Keppel 1975]

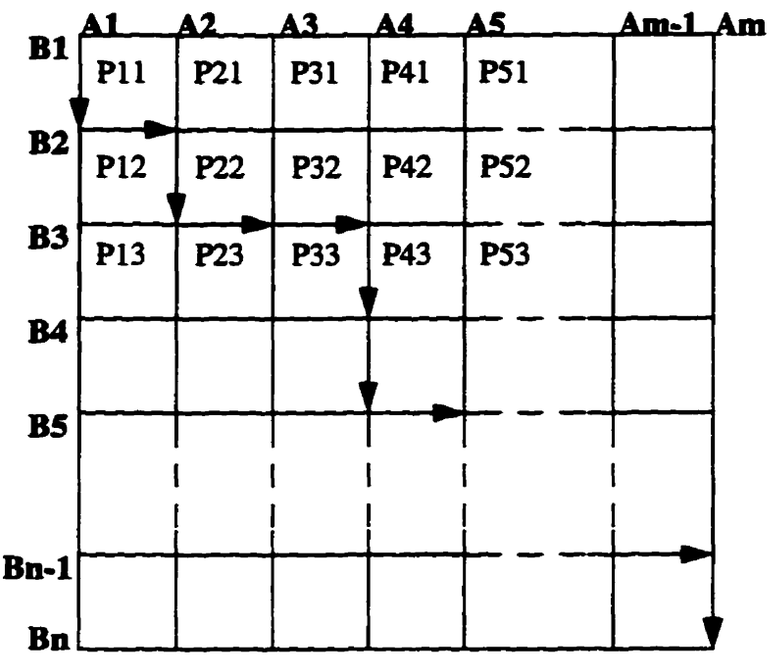

Figure 2.6: Each Pij represent a matrix that triangulates 2 convex sets. [Keppel 1975]

2.2.2 Heuristic triangulation.

Christiansen presented a heuristic triangulation algorithm which is called the *shortest diagonal method* [Christiansen 1978]. In figure 2.8, line segment 1t-1b is initially constructed. Then there are two choices to make a triangle: 1t-1b-2t or 1t-1b-2b. The line 1t-2b is selected because it is shorter then 1b-2t. Thus, we have a triangle with the smaller surface area and, intuitively, this will produce the best, near optimal reconstruction (figure 2.9). Figures 2.7, 2.8 and 2.9 are borrowed from [Christiansen 1978].

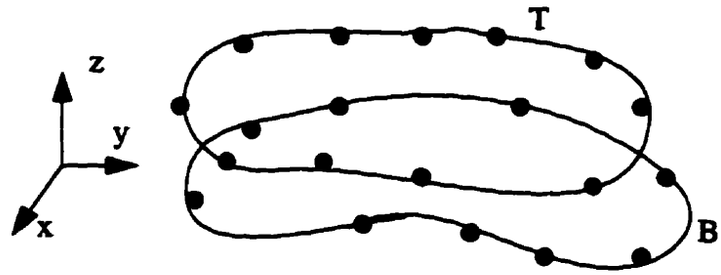

Figure 2.7: Original contours. [Christiansen 1978]

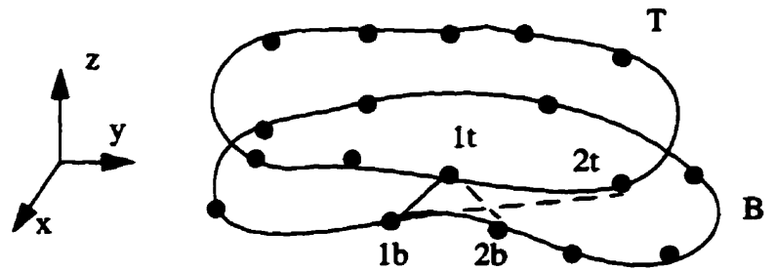

Figure 2.8: A step in the shortest diagonal algorithm. [Christiansen 1978]

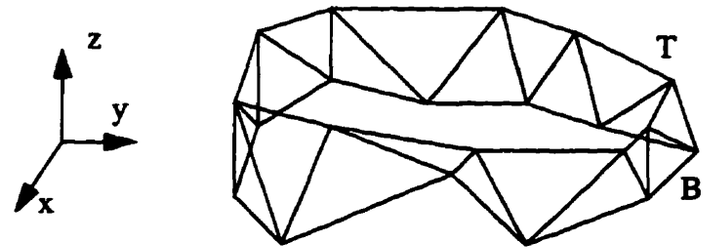

Figure 2.9: The final triangulation between the 2 contours. [Christiansen 1978]

In [Ekoule 1991], the authors proved that this method of connecting the smaller edges along the contours are not adequate when the two contours are different in shape (i.e. when one contour is convex and the other has many concavities). The authors proposed another solution to the problem by transforming concave contours into their convex hulls and then associating points to each other by minimizing a certain distance measure. After that a heuristic method of connecting the remaining points is used. This method depends

on the shortest diagonal method explained by [Christiansen 1978]. The reconstructed models are now better but the contour transformation process is considered a huge overhead to the triangulation process.

Another heuristic algorithm was presented by Ganapathy and Dennehy in 1982. They defined φ_α to be the length of the contour segment included in a triangle divided by the perimeter of the contour on which this segment lies. Consequently, if we have contours h and v with m and n as number of sampled points respectively then we get

$$\Phi_h = \sum_{i=1}^{m} \varphi_{h_i} = 1 \qquad (2.1)$$

$$\Phi_v = \sum_{j=1}^{n} \varphi_{v_j} = 1 \qquad (2.2)$$

In this algorithm we try to connect the points along the contours to minimize the absolute difference between these two values Φ_h and Φ_v. When an acceptable surface is reached the difference between these two values will be 0 because we would have passed through all the points on the 2 contours. See figure 2.10, which is borrowed from [Ganapathy 1982], for an illustration of one step in the algorithm.

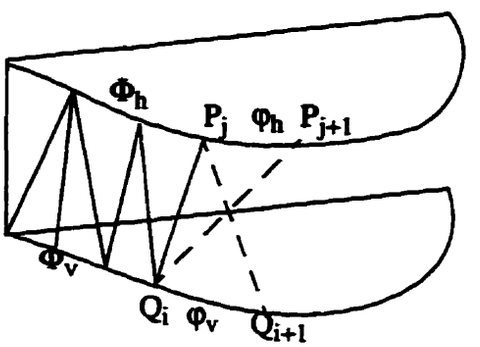

Figure 2.10: One step in the Ganapathy algorithm. [Ganapathy 1982]

2.2.3 Heuristic branching.

Christiansen presented a solution to the branching slices problem [Christiansen 1978]. The solution is summarized in the following steps (see figures 2.11 and 2.12 which are borrowed from [Christiansen 1978]):

- Introduce a new node midway between the closest nodes on the branches. The *Z* coordinate of the new node is the average of the two *Z* coordinates of the 2 slices.
- Renumber the nodes as shown in figure 2.12. Notice that the new node and its neighbors are numbered twice.
- Triangulate as usual using the shortest diagonal approach explained above. [Christiansen 78].

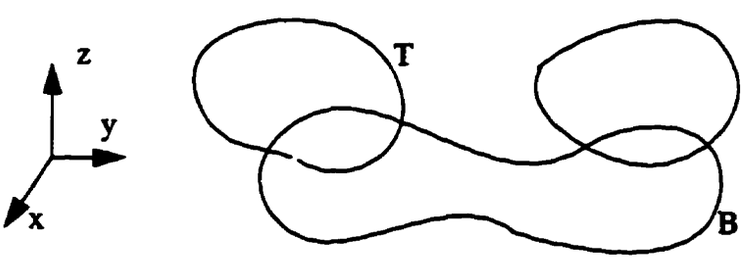

Figure 2.11: Branching slices.[Christansen 1978]

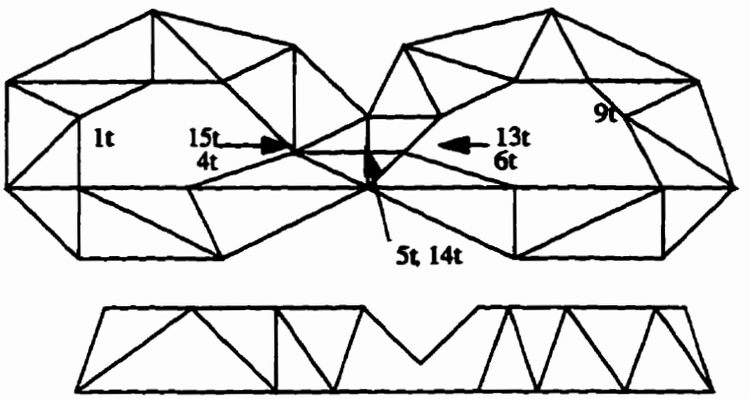

Figure 2.12: The result of the Christiansen algorithm. [Christiansen 1978]

Unfortunately, branching slices which have the shape shown in figure 2.13, where B_1 is branching to T_1 and T_2 cannot be recovered using the above algorithm. This is because no single intermediate node will help in this case; we need more than one node. Christiansen proposed that the only way to select more than one intermediate node is to make the process interactive so that the user will be able to select as many nodes as he wants. Figure 2.13 is borrowed from [Christiansen 1978].

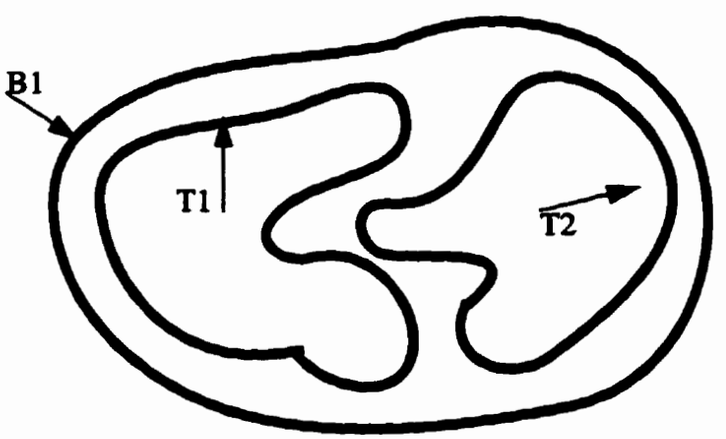

Figure 2.13: A difficult case of branching slices which requires user intervention.[Christansen 1978]

In [Meyers 1992], the authors proposed a solution to the above mentioned limitation of the branching algorithm. They proposed that the interior (or the "canyon" as they referred to it in their paper) between contours T1 and T2 (figure 2.13) could be triangulated on the plane of the slice T. No more points are added between the two slices. Then the outer parts of the contours T1 and T2 are triangulated as usual with contour B. The authors failed to find an automatic method for detecting canyons between branching contours. The method they used was that of constructing the convex hull of the two contours (T1 and T2 in our example). As soon as a pair of points, one on each contour, is found to be connected in the convex hull, then this pair is a candidate for the entrance of the canyon. This method will not work when the two branching contours are curved (i.e. circles) [Meyers 1992].

Another branching algorithm was presented in [Ekoule 1991] and it is dependant on the triangulation algorithm mentioned in the previous subsection. If a contour in a source slice is to be connected to more than one contour in the destination slice, then an intermediate contour is created by interpolation. The intermediate contour is simply the union of all the contours in the destination slice where the outer contour pieces are the only ones kept in the final interpolated contour. This intermediate contour is then triangulated with the source contour and with each of the contours on the destination slice [Ekoule 1991]. The branching point is assumed to be at the middle between the source and the destination slices and this might not be the case in reality. Again, the process of transforming all the contours to their convex hulls is very costly and would make the whole process very slow.

2.2.4 Delaunay triangulation.

In [Boissonnat 1988], a Delaunay triangulation algorithm is presented for reconstructing

3-D objects from cross-sections. A *Voronoi diagram* is a sequence of convex polyhedra covering a d-dimensional space E where each polyhedron consists of all the points of E which are nearest to a point in a pre-defined set of points M. The dual of the Voronoi diagram is the Delaunay Trinagulation (DT). See figure 2.14 for an example borrowed from [Boissonnat 1988].

The paper lists a series of propositions and we will give here the results of that paper. For detailed proofs refer to the original paper [Boissonnat 1988]. The first proposition is that the contours of each cross-section will be contained in the final DT if and only if, in each plane, the contours are contained in the 2-D DT of that plane. Therefore, computing the 2-D DT of the contours of each plane is the first step in the process of 3-D reconstruction.

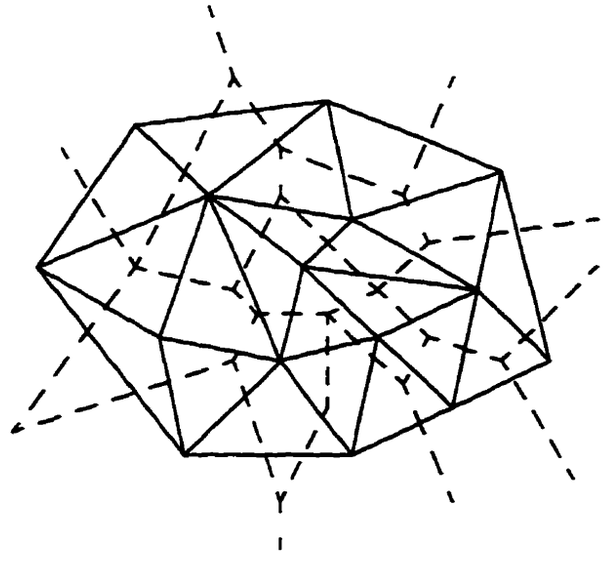

Figure 2.14: Dots represent points in set M, dashed lines are the Voronoi diagram and the solid lines are the Delaunay triangulation of M. [Boissonnat 1988]

Faces and points could be connected between the two DT of the two contours. Three types of tetrahedra are defined. The first is T1, where the face of the tetrahedron is on the P1 plane, the second is T2, where the face of tetrahedron is on the P2 plane and the third type is the T12 which is a tetrahedron with one edge on each plane (We will assume that this is a special case of a tetrahedron). See figure 2.15 for an illustration of the different types borrowed from [Boissonnat 1988].

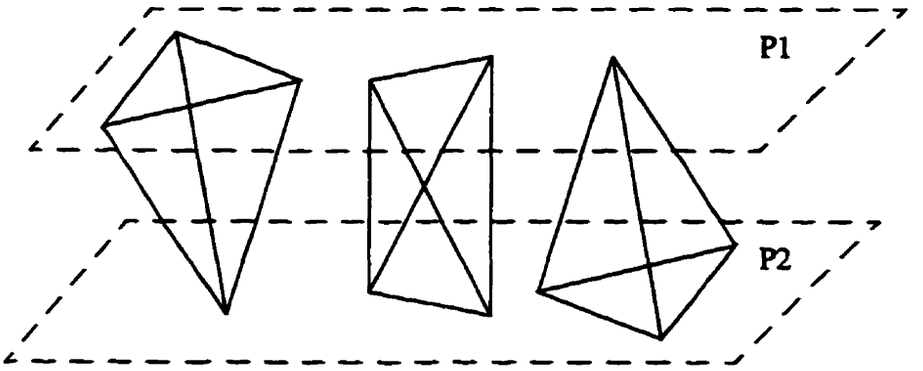

Figure 2.15: Three types of tetrahedra T1, T12 and T2. [Boissonnat 1988]

Another result presented in the paper is that whenever we project (perpendicularly) the triangles (and the Voronoi diagram) of one plane onto the other we can know what edges are connected to what faces/edges in the final 3-D DT. Suppose we have 2 triangles ABC in plane 2 and abc in plane 1. When we project both triangles and Voronoi diagrams onto each other we get the image shown in figure 2.16, which is borrowed from [Boissonnat 1988], where V_{ij} is the Voronoi dual of the edge ij. The nearest point on plane 2 to the center of the circle circumscribing the points of p(a)p(b)p(c) (abc after projection) is the point B. Thus we have tetrahedron abcB, of type T1, in the final 3-D DT. Similarly, the closest point on plane 1 to the projection of ABC (of plane 2) on that plane is the point a. Consequently, we have the tetrahedron aABC, of type T2, in the final output. Moreover, whenever two edges of the Voronoi diagram intersect, we have a

tetrahedron of type T12. Notice that the graph in the same figure represent the two intersecting Voronoi diagrams and the nodes are the tetrahedra of the final 3-D DT. Some of the produced edges and/or faces are outside the contours of the object to be reconstructed. Thus, tetrahedra which contain those edges and/or faces will be discarded. This is the same as removing the nodes which correspond to these tetrahedra from the graph.

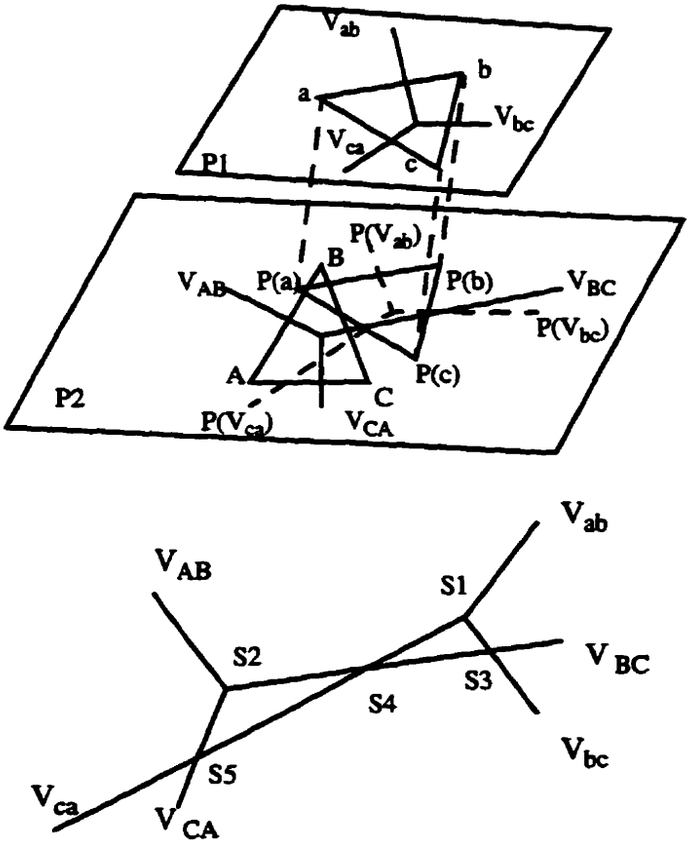

Figure 2.16: 2 Planes having a triangle each and the projection of the one in P1 on P2. The corresponding graph is also shown. [Boissonnat 1988]

Boissonnat claimed that keeping horizontal surfaces is adequate in some applications. This is done by triangulating the points of the contours at the same slice (a direct result

of applying the algorithms mentioned above). We believe that this is not correct since the reconstructed model will be misleading as seen in the example shown in figure 2.17. The same effect happens when a branching body is reconstructed using the same algorithm. Figure 2.17 has been produced using a software package available from the Los Alamos National Laboratory. Also, it is clear from the same figure that the branching point is assumed to be at the lower slice and this might not be the case in reality.

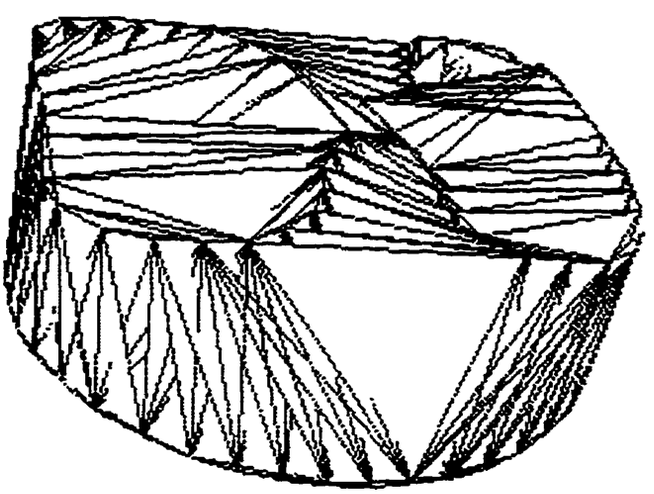

Figure 2.17: The horizontal surface is kept when a irregular shape (at the top) is triangulated with a circle (at the bottom). This is not acceptable in some applications.

Also, Boissonnat claimed that radial cross-sections could be reconstructed in a similar manner as the parallel ones. The only difference is that instead of perpendicularly projecting planes onto each other, the planes will be rotated until they overlap and then the same algorithm could be used. This is true but it will be very costly since the process of rotating the slices will not be a trivial one. On the other hand, the process will be useful in handling radial slices. An alternative fast algorithm is explained below in Section 5.8.

2.2.5 G-Octree reconstruction from a series of G-Quad trees.

In 1987, Mao et al proposed an algorithm for reconstructing 3-D images from 2-D cross-sections represented by G_Quad trees, a variation of the Quad tree where each node contains the pixel value of a corresponding region in an image. The image is represented as a series of binary numbers that correspond to the gray levels of the pixels as shown in figure 2.18 which is borrowed from [Mao 1987]. An example of constructing a G_Quad tree from an image is shown in figures 2.19 and 2.20 which are borrowed from [Mao 1987]. The upper right four pixels have a common 2 most significant bits (00) and thus the corresponding node in the G_Quad tree has (00). The same goes for the lower right four pixels and the lower left four pixels. On the other hand, the upper right four pixels have different 2 most significant bits. This leads to the expansion of the corresponding node to four more nodes at the following level each representing a different value. The process of reconstructing the G_Quad tree is repeated until the desired accuracy is reached.

00001	10110	00110	00110
00010	11110	00111	00110
00010	00011	00111	00110
00010	00011	00110	00010

Figure 2.18: An image representing the gray scale of each pixel in a binary format [Mao 1987]

001	110	110	110
010	110	111	110
010	011	111	110
010	011	110	010

Figure 2.19: The resulting image after one step of G_Quad tree construction. [Mao 1987]

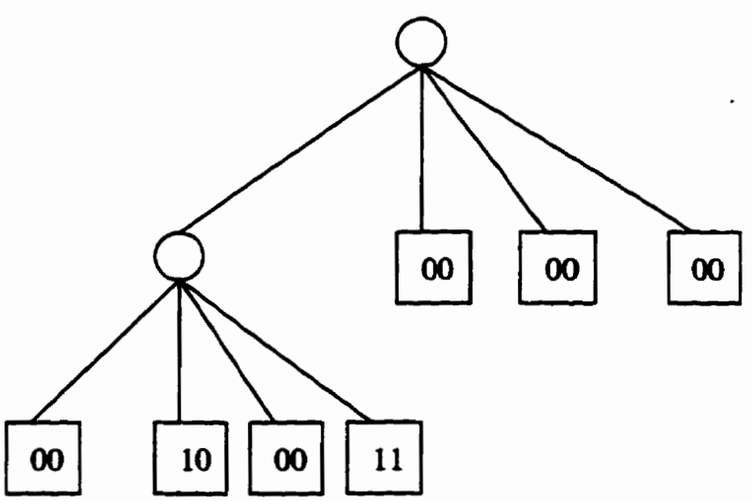

Figure 2.20: The G_Quad tree after 1 step. [Mao 1987]

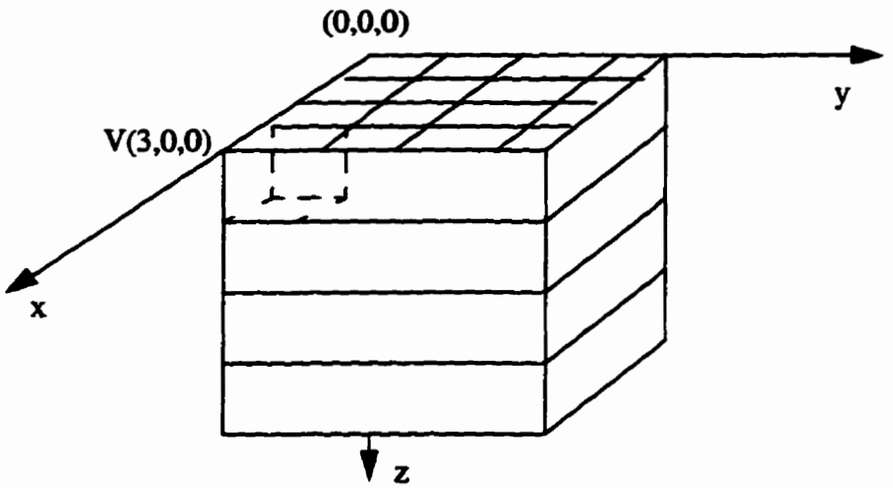

Figure 2.21: A 3-D volume represented by some sort of voxels. [Mao 1987]

In the 3-D case, the same process is performed to produce G_Octrees. Instead of looking at neighboring 4 pixels, the neighboring 8 pixels are examined and the Octree is reconstructed. Each leaf node in the produced Octree will represent the color of a voxel in the 3-D space as shown in figure 2.21 which is borrowed from [Mao 1987]. Most of the software packages available for 3-D reconstruction employ volume rendering as described in this section. See figure 6.28, in Chapter 6, for an example of a volume rendered 3-D model.

2.2.6 Syntactic/semantic approach.

In [Kehtarnavaz 1988] a 'high level' description of contours was presented where contours of similar substructures are triangulated together. The main steps of this process go as follows:

- Shape retrieval: Each contour is approximated by a polygon as explained in [Pavlidis 1982].

- Syntactic/semantic contour representation: Each polygon is analyzed and a string is produced that describes the primitives in this polygon and their attributes. Primitives might be an arc (quad-arc) within a certain limit of perimeter (attribute) or a line with a certain length (another attribute).
- Relationship between contours: A graph is constructed relating all feature strings using Levenshtein distance.
- Surface formation: Similar substructures in each pair of consecutive parallel contours is found by searching the graph for minimum distances between the corresponding feature strings. Similar substructures are then triangulated using any of the previously mentioned triangulation algorithms.

The algorithm is very complicated (i.e. very costly) because of the fact that if the contours are close to each other then similar substructures will be triangulated automatically without using the above mentioned algorithm. Otherwise, if the contours are far apart, then the algorithm is not relevant since the contours will vary in shape and the only solution to this problem is to interpolate as mentioned below in Section 5.7. The reconstructed models shown in [Kehtarnavaz 1988] are not sufficient to justify the algorithm. Just one example is given and this example is just a reconstruction from 3 contours.

2.3 Available software packages.

IMOD and 3DVIEWNIX are 2 software packages used to reconstruct 3-D objects from 2-D slices of medical images (e.g. CT scans). In both programs, the user selects a value for the threshold for all the images (i.e. all the 2-D cross-sections will have the same threshold). This has the advantage that the user will be able to select whatever objects he/ she wants to be taken into consideration in the reconstruction process. In IMOD, the user

can select, with free hand, the objects he/she are interested in. This has another advantage that the user can process contours that cannot be selected with one threshold for the whole image. Volume rendering is used to produce the 3-D models in the two packages. This is very similar to the algorithm described in Section 2.2.5.

In both programs the process of reconstruction is very much delayed because of the involvement of the user. Moreover, if we ignored this delay, the process still takes a long time (especially when the 3DVIEWNIX is used). In Chapter 6, actual time comparisons are presented.

2.4 Evaluation of the state of art.

Due to the limitations of the different branching algorithms mentioned above (mainly the algorithms of [Christiansen 1978], [Boissonnat 1988], [Ekoule 1991] and [Meyers 1992]) a new method should be developed to account for the defects. In all of the algorithms the point of branching is assumed at a certain point between the two slices or at one of the two slices and this might not be correct (realistic). Also, the lack of interpolation between slices in most of the techniques will produce 3-D models that are not smooth.

Note that the available software packages for 3-D reconstruction make the user interactively select the objects in the slices that must be taken into consideration in the 3-D reconstruction. This is usually done by either allowing the user to select, using a mouse, the desired objects, or to select a threshold value which is adequate according to him/her. Therefore there is a need for an automatic system were the user should not intervene altogether or at least does not intervene for each model reconstruction (i.e. set

some parameters for a class of images to be used as input to the reconstruction process).

2.5 Criteria for the solution.

To satisfy the 2 criteria mentioned in Section 1.2 we need to develop a solution with a new branching/interpolating technique so that the produced 3-D model is as realistic as possible. Also, the speed of reconstruction must be taken into consideration because the process of producing the slices themselves take a long time. In addition to that an automatic system (i.e. where the user should not intervene) is desired. To summarize, the main criteria for the solution are:

- The production of realistic 3-D objects obtained from a branching/interpolating algorithm.
- Fast solution compared to existing solutions.
- Automatic solution where users will not be involved.

Chapter 3 Seismic signals

A study of seismic signals is of great importance for the development of an appropriate solution for reconstructing 3-D models from cross-sectional seismic slices. The following is a basic introduction to seismic signals, seismic surveys and seismic data processing.

To produce a seismic signal, a seismic source and a geophone are needed. The source and the geophone (sometimes called receiver) are fixed on the surface of the earth (or inside it). And a sound wave (between 10 Hz and 100 Hz) is emitted from the source and wave travels through the earth until it reaches a surface (an interface between two media with two different velocities of sound) and gets reflected back to the surface of the earth. The receiver will then record the reflected signal. This process will produce a two direction time map for the receiver. In other words, a peak signal will appear after the time needed for the wave to go down and then up after reflection. The following two sections will explain the different geometries for sources and receivers and then the

sequence of data processing needed to produce a cross-sectional representation of the underground surfaces from the two direction time maps.

3.1 Different geometries.

Different arrangements of sources and receivers are possible; in this section we will discuss several different arrangements. The first one is the *zero-offset* arrangement, and is shown in figure 3.1. In the Zero offset arrangement, the sources and the receivers coincide; that is, they are virtually at the same point on the surface of the earth. It is used generally in simulations and as a model of the process called stacking. The signals are already stacked and they only need to be migrated[1] to produce the 2-D cross-section of the objects surveyed under the surface of the earth.

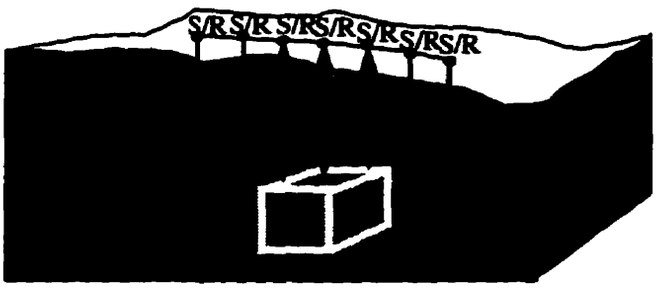

Figure 3.1: A zero offset arrangement. S/R means a source/receiver pair at the same point on the surface of the earth. [Parker 1996b]

The second arrangement is the common source in which we have one source and a spread of receivers over the area to be surveyed. The receivers are equally spaced. An

1. Migration is the process of producing a depth image from a zero offset (stacked) representation. A migrated image is a representation of the underground surfaces. It is explained in detail in the following section.

illustration of this arrangement is shown in figure 3.2.

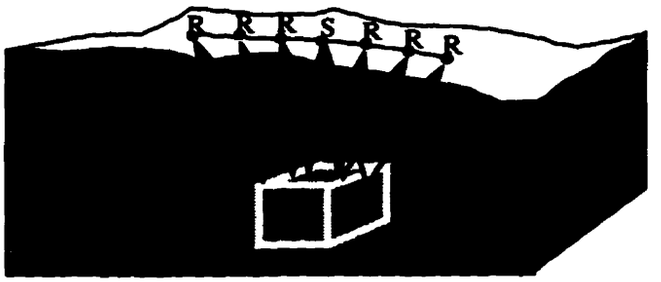

Figure 3.2: Common source arrangement. [Parker 1996b]

The two arrangements mentioned above produce parallel planar cross-sections when the lines on sources and receivers are arranged in parallel over the surface of the earth as shown in figure 3.3.

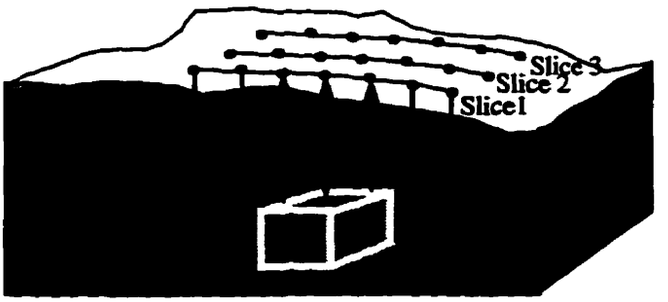

Figure 3.3: Parallel seismic lines with 5 pairs of source/receiver combinations. Top view over the surface of the earth. [Parker 1996b]

After migrating the produced signals from this parallel arrangement, any of the algorithms mentioned above in Chapter 2 could be used to reconstruct the 3-D model after appropriate preprocessing. A problem arises with the third arrangement to be discussed in this section which is the Vertical Seismic Profiling (VSP). The source-receiver arrangement is shown in figure 3.4.

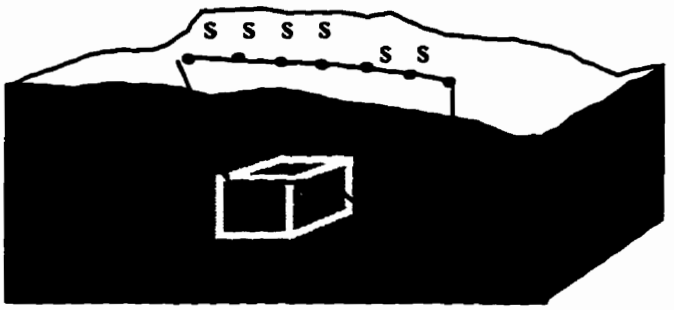

Figure 3.4: VSP arrangement. [Parker 1996b]

In this arrangement the sources are equally spaced at the surface of the earth and the receivers are arranged over a borehole line inside the earth. The receivers are also equally spaced. Again, after stacking and migration, a cross-sectional slice will be produced representing the underground surfaces which exist. Due to the availability of one line of receivers (one borehole inside the earth), the produced cross-sections in a VSP survey are radial as shown in figure 3.5.

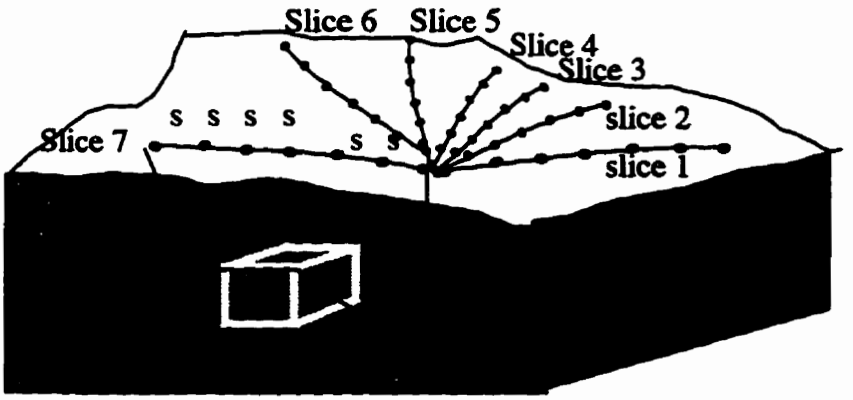

Figure 3.5: VSP arrangement with more than one slice. [Parker 1996b]

In this figure, each dashed line has a series of equally spaced sources. The borehole has

the equally spaced receivers. The produced slices are now separated by a certain angle and not by a distance as in the previous two arrangements. Reconstructing from radial cross-sections, that may result from a VSP arrangement, was not investigated before and consequently we developed an algorithm to solve this problem in Section 5.8 below.

3.2 Seismic data processing.

To produce a cross-sectional slice of underground objects from sonar signals a series of data processing operations should be performed. The operations are performed in that sequence:

1 - Deconvolution.

2 - Normal-moveout.

3 - Stacking.

4 - Migration.

The result of the migration process is a cross-sectional slice of the underground objects. In this section each of these processes will be discussed in details. The first operation is deconvolution, which adds to the temporal resolution of the input signals [Yilmaz 1987]. The recorded soundings have peaks wherever there is a difference in the velocity of sound in the underground media. This can only happen when two different kinds of media interface. Thus, the recorded seismogram could be modeled as a convolution between the earth's impulse response and the seismic wavelet. This wavelet is a combination of source signature, geophone response, recording filter, etc. What we want is the earth's impulse response representing the true reflections. So, the seismic wavelet is compressed to a spike so that the earth's impulse response is produced [Yilmaz 1987]. This process is called deconvolution and it could be performed in a manner similar to the

one performed in digital image processing [Gonzalez 1992]. Inverse filtering or Weiner filtering could be used in this process.

The purpose of normal-moveout is to produce an approximation of a zero offset arrangement where the sources and the receivers coincide at the same point on the surface of the earth. Assume that we have the arrangement shown in figure 3.6 below. S is the source, R is the receiver and M is the mid-point between them. D is the point where the ray from S is reflected at the interface beneath the ground. Let $t(x)$ to be equal to the time taken from S to D and then to R and let $t(0)$ to be double the time through MD. Now the travel time equation is as follows according to the Pythagorean theorem.

$$t^2(x) = t^2(0) + x^2/v^2 \quad , \qquad (3.1)$$

where x is the distance (offset) between the source and the receiver and v is the velocity of the medium above the interface. This equation represent a hyperbola in the plane of 2-way travel time versus offset (distance in x direction).

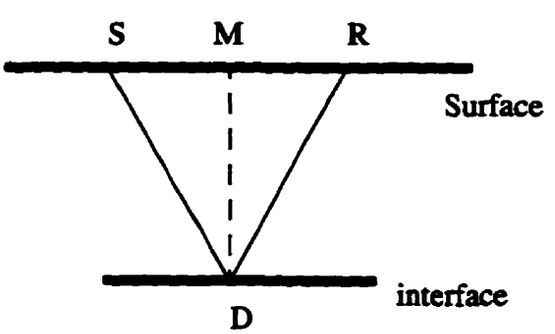

Figure 3.6: NMO geometry.

The normal moveout correction is given by

$$\Delta t_{NMO} = t(x) - t(0) \qquad (3.2)$$

This NMO correction is equivalent to the correction shown in figure 3.7. In this figure, the point A is transferred according to the above equation to point A' on a straight line parallel to the x-axis. This means that if we are given a seismic trace of a horizontal layer beneath the ground produced by the geometry shown above in figure 3.6, we will get a hyperbola in that trace. NMO correction will transform this hyperbola into a straight line as shown in figure 3.7 and this straight line representation is the one expected when a zero offset survey is performed (as a matter of fact, that might not be possible in practice). We might add here that when dipping (tilting) surfaces exist, the equation for the NMO correction will differ, since the angle of dipping will be taken into consideration, but the effect will be the same [Yilmaz 1987].

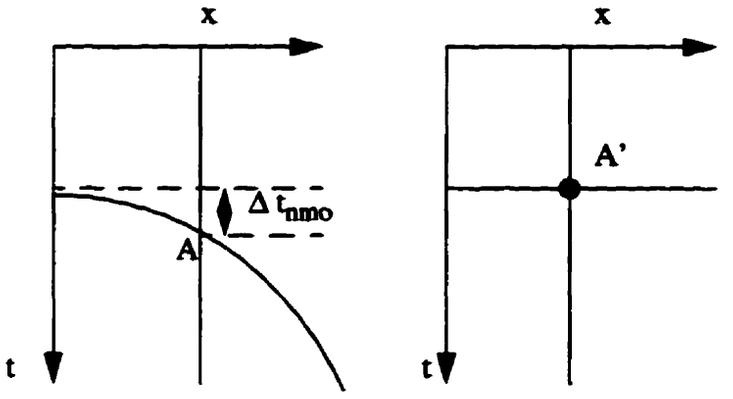

Figure 3.7: NMO correction.

The process of stacking is simply adding all the traces at zero-offset to reduce random

and coherent noise. The final step in the seismic data processing is migration. If all the surfaces we have are horizontal (i.e. parallel to the level of the earth), then we might not need migration. The goal of the migration process is to produce an image similar to the cross-section of the underground surfaces by relocating dipping surfaces and removing any kind of diffractions that might occur during the survey process [Yilmaz 1987]. There are many algorithms for migration which could be used for different outputs. What we need here is a migration with an output image in depth. To perform depth migration some information must be known, such as the location of each source and geophone and the velocity of seismic wave propagation through the medium above the reflector.

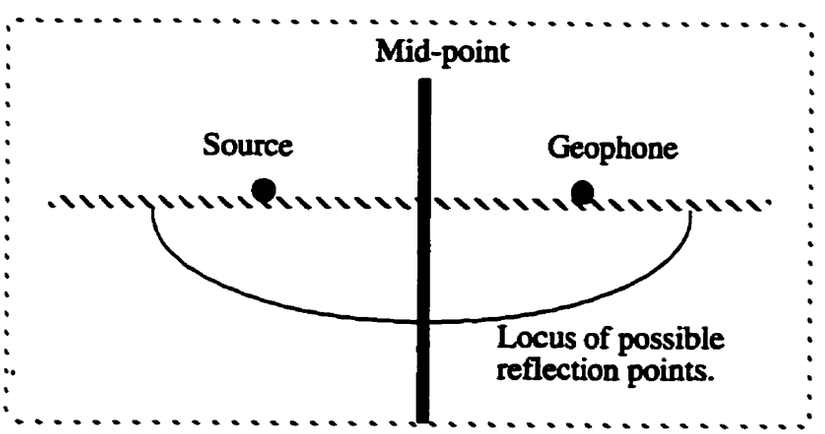

Figure 3.8: Kirchhoff Migration.

In figure 3.8 it is shown that for every geophone receiving energy, at a specific time, from a certain source there is a collection of possible points under the ground where the surface might exist. This collection of points have the form of half an ellipse. For each source / geophone pair it is one and only one point on the surface which produces the reflection at the geophone. Taking into consideration more and more pairs of sources and geophones, we get a collection of such ellipses which if summed together, the resultant image will represent the underground surfaces. This process is called Kirchhoff migration [Dohr 1981].

Chapter 4 Preprocessing

Preprocessing is needed to produce a set of data that represent the different contours on a certain slice from the original cross-section. Many techniques of computer vision, such as thresholding, segmentation and contour tracing, are used to produce the required set of points needed for the following processing stage (triangulation). In this chapter the preprocessing stage is explained in detail and a primitive triangulation procedure is presented and evaluated.

4.1 Introduction.

Input slices are 255 gray scale images. The slices are parallel in 3-D space and the Z parameter of each slice is input to the program. *The main purpose of the preprocessing stage is to produce, for each contour in each slice, a series of points that represent those contours so that triangulating could be performed using the sample points in each contour for each two consecutive slices.* Slices are thresholded, then segmented and

sampled points for each object in each slice are produced. The sampled points in each pair of slices are matched and the matched points are connected to form the 3-D mesh which is then projected perspectively on the 2-D plane. The process is described in the sub-sections below and an image from a CAT scan sequence of a human skull is used to illustrate the different steps in the procedure (figure 4.1).

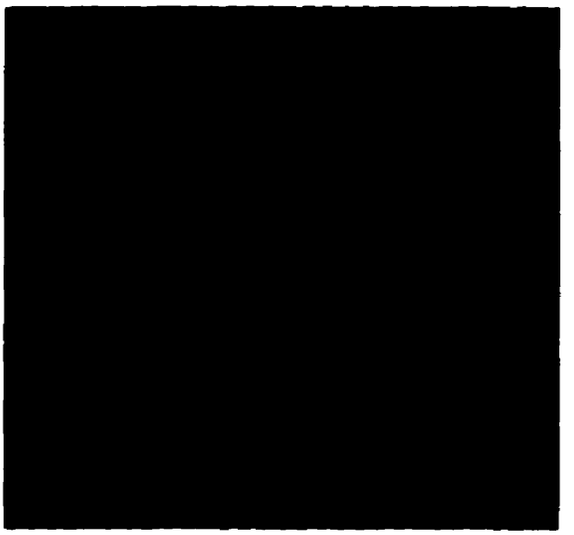

Figure 4.1: A CT slice through the upper jaw and the lower part of the skull.

4.1.1 Thresholding.

An interactive thresholding technique is used here where the user selects a certain threshold value to process the image. The input to this module is a gray level image. The algorithm starts with scanning the image row-wise (or column-wise) and each pixel is compared to the threshold value (say 128 - chosen by the user). If the value of the pixel is less than 128, the output pixel value is 255. If the value of the pixel under investigation is greater than 128, the output pixel value is 0. Thus, a bi-level image is the output of this process. A pixel value of 0 indicates the presence of an object at this point

and the a pixel value of 255 indicates the presence of background.

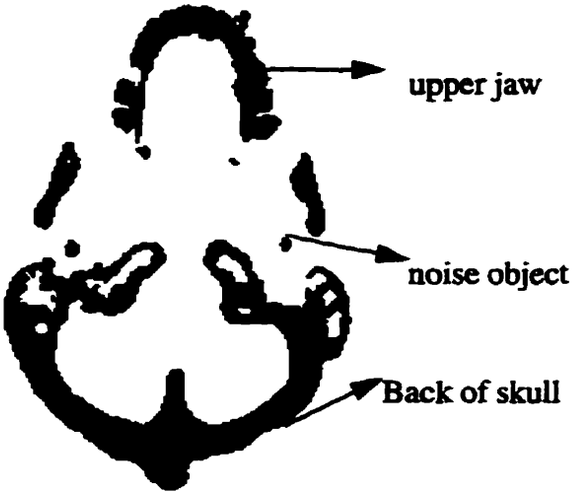

Figure 4.2: A threshold CT scan (Threshold value = 13).

Applying that algorithm to the image in the previous section, the output will be the bi-level image shown here in figure 4.2 (threshold used is 13). Notice how some noisy objects are produced from this process. This will be prevented when the new algorithm presented in Chapter 5 is used.

In all of the available 3-D reconstruction software packages the user has to select a threshold value for each image (and sometimes it is more limited to one threshold value for all the slices). To automate this process a large set of the available thresholding algorithms were tested for the CT scan image and simulated seismic slices. The following is a list of all the thresholding algorithms tested:

- Thresholding using edge pixels [Weszka 1974].
- Iterative selection thresholding explained in [Ridler 1978] and [Thrussel 1979].

- Method of grey level histogram presented in [Otsu 1978].
- Entropy methods presented in [Pun 1981], [Kapur 1985] and [Johanssen 1982].
- Fuzzy sets methods presented in [Huang 1995] and [Yager 1979].
- The mean method were 50% of the pixels in the finally thresholded image are black and the other 50% of the pixels is white.
- The two peaks method were two peaks in the histogram of an image are located and a low point between those two peaks is selected to be the threshold value.

All of the above algorithms produced a different threshold for each slice (grey level image). They are classified as single threshold algorithms as compared to regional threshold algorithms which take more processing time (not used here because of the requirement that our system should be fast). Each of the produced images, for each of the above algorithms, was compared subjectively to an image where the user selected the threshold value himself / herself. Also, the whole preprocessing stage is completed for each of the above algorithms and then the produced 3-D model was compared to a model produced by interactively selecting a threshold. The result of these subjective comparisons yielded the result that the entropy method for thresholding as described in [Kapur 1985] is the best compared to others and it could be used for thresholding seismic slices. This method was employed by our system.

4.1.2 Segmenting.

Each thresholded slice is processed to produce a map which gives the locations of all the different objects in the slice. The segmentation procedure starts with scanning the image and stopping when a pixel value of 0 (an object) is encountered. This pixel location is passed to the marking function which marks (or labels) all the 8-neighbors of that pixel with a value other than 0 (object) and 255 (background). The marking function proceeds to mark all the neighboring pixels until no more neighboring unmarked pixels exist. The

process is repeated for all the objects in the image slice. The final output is an image with 255 as background pixel values and a different pixel value for all the pixels in each object (figure 4.3).

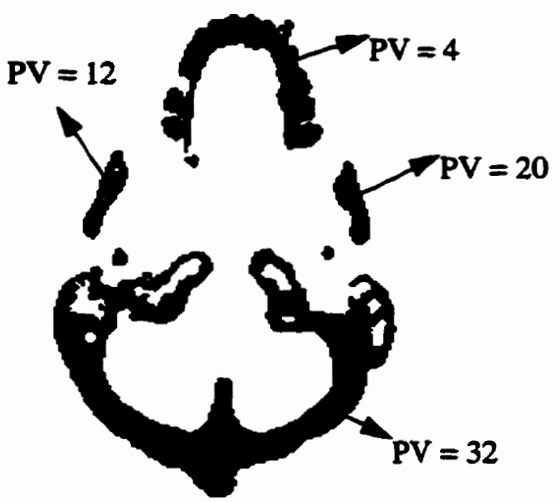

Figure 4.3: Segmented image (PV = Pixel Value)

The algorithm to do this process is explained in [Parker 1994]. The image is raster scanned until a pixel value of 0 is encountered. That pixel is called the seed and its x and y coordinates are called the iseed and the jseed.

```
function mark_8 (IMAGE x, int value, int iseed, int jseed) {
x->data [iseed][jseed] = (unsigned char) value;
/* Mark this pixel with a value other than 0 */
/* loop for all neighboring pixels */
for (i=-1; i<=1; i++)
```

```
    for (j=-1; j<=1; j++) {
     n  = i+iseed; m=j+jseed; /* update seed value */
     if (range (x,n,m)==0)  continue;
     /* if pixel is not within image boundaries go on */
     if (x->data[n][m] ==0)  /* if that pixel belongs to the same object */
       mark_8 (x,value,n,m);  /* mark it by calling the function recursively. */
    }
  } /* function mark_8 */
```

4.1.3 Contour finding.

For each object in the slice, its contour is found. Contours are found by scanning the whole image for values other than 255 (i.e. we are looking for object pixels) and if this pixel has a neighbor which is a background (255) then it is recorded as a contour pixel, otherwise it is not a contour pixel. When we look for neighboring background pixels, the algorithm examines the up, down, right and left pixels only so that the produced contour is 4-connected with the background (figure 4.4).

Figure 4.4: Contours of the segmented CT slice.

The algorithm is as follows:

```
/* Assumption: no more than 60 regions. Region numbers are multiples of 4 */

 for (i=0; i<no_of_rows; i++)
    for (j=0; j<no_of_columns; j++) {
       if (image->data[i][j] != 255) {
        /* if not a background pixel - an object pixel */
        val = image->data[i][j];  /* Get the pixel value for that object */
        for (n=-1; n<=1; n++) /* loop for all surrounding pixels */
          for (m=-1; m<=1; m++) {
            if (n*m) continue;  /* 4-connected to background */
            if (range (image,i+n,j+m)==0) continue;
            /* Check if pixel is within image boundaries */
            /* If the neigboring pixel is a background pixel then that pixel under
             investigation is a boundary pixel ---- Mark with val + 1 */
             if (image->data[i+n][j+m]==255) image->data[i][j] = val+1;
        }
       } /* If image data is 255 */
    } /* for j=0; */

 /* Scan the image, if a value is found which does not give a 0 remainder when it
is divided by  4, then mark this value with val otherwise erase this value with a
white pixel 255 */
 for (i=0; i<no_of_rows; i++)
  for (j=0; j<no_of_columns; j++)
    if ((image->data[i][j] % 4) == 0) image->data[i][j] = 255;
```

```
else if ((image->data[i][j] % 4) == 1) image->data[i][j] --;
```

4.1.4 Contour tracing.

Each contour is traced using the chain coding algorithm explained in [Freeman 1961] and sample points are taken at constant steps. The sampling rate is constant for all the contours in all the slices. For example, if the sampling rate is 7, then the algorithm records a pixel every 7 pixels along the contour under investigation in a clockwise direction. These points are the definition of the contour in 3-D space and at the end straight lines will be drawn between these points for each contour. The following is the result of applying the chain algorithm to the image of CT scan. Notice how the points are very close to each other when the sample step is 3 (figure 4.5(a)). Of course, using a sample step more than 3, say 8 (see figure 4.5(b)), will produce a coarser representation of the contour and consequently this will affect the quality (smoothness) of the produced triangular mesh since the number of line segments (triangle bases) will be fewer.

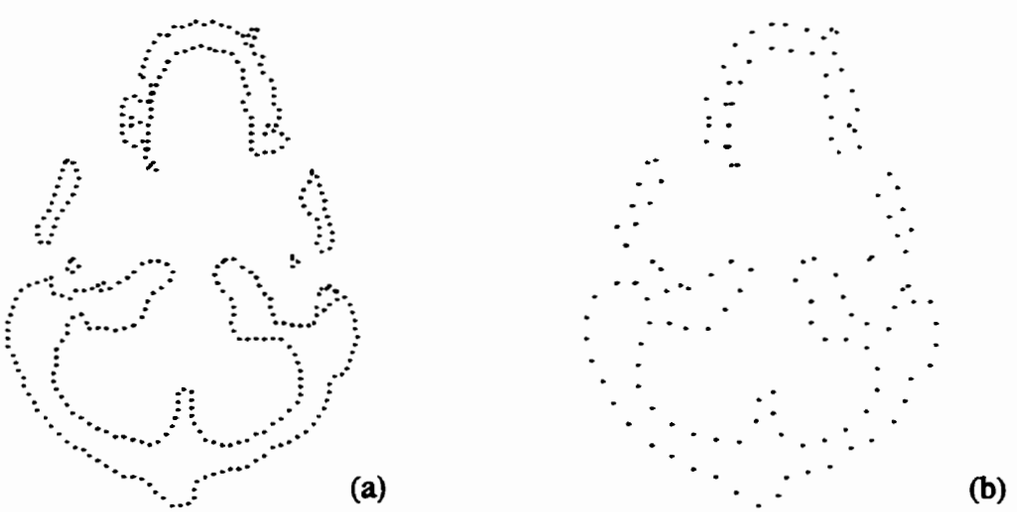

Figure 4.5: (a) Sample step equal to 3. (b) Using a sample step equal to 8 produces sparse sample points.

4.1.5 Matching points.

For each two slices, 1 and 2, the sampled points are matched. Each point in slice 1 is compared to all the points in slice 2 within a window of a constant size. Using a window of search is intended to limit the search space and it is justified by the fact that points which should be connected are close enough to each other in 3-D space. Euclidean distances are calculated and the point in slice 2, which has a minimum Euclidean distance to the point under investigation in slice 1, is the corresponding point. Euclidean distances are calculated according to the following equation.

$$Dist = \sqrt{\langle x_1 - x_2 \rangle^2 + \langle y_1 - y_2 \rangle^2 + \langle z_1 - z_2 \rangle^2} \qquad (4.1)$$

4.1.6 Perspective projection of the resulting object(s).

The finally matched points are then connected as shown in figure 4.6. Notice how the dashed lines form the final approximating polygon. The final three-dimensional mesh is projected perspectively on the 2-D plane to produce an image of the reconstructed object from a certain viewing angle.

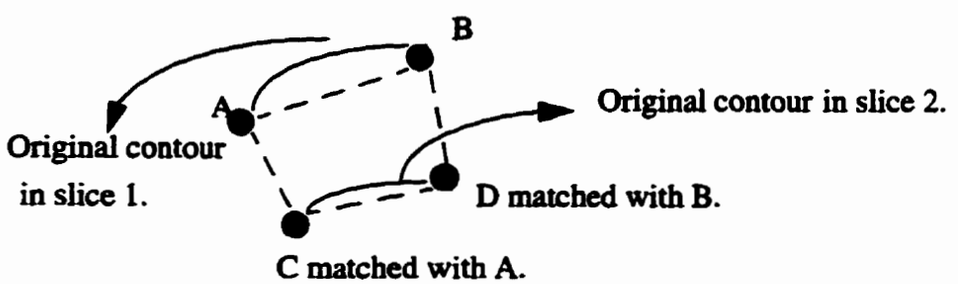

Figure 4.6: matching points in a primitive way.

4.1.7 Limitations of the primitive solution.

The problem with the primitive algorithm is in the handling of branching slices. Notice, in figure 4.7, that the sampled points on the left hand side of T_1 will be connected to the sampled points on B_1. On the other hand, the points on the right hand side of T_1 will not be connected to any of the points in B_1 because these points are not closer to the points on B_1 than the others at the top and bottom parts of T_1. The same goes for T_2 leaving a hole in the reconstructed model.

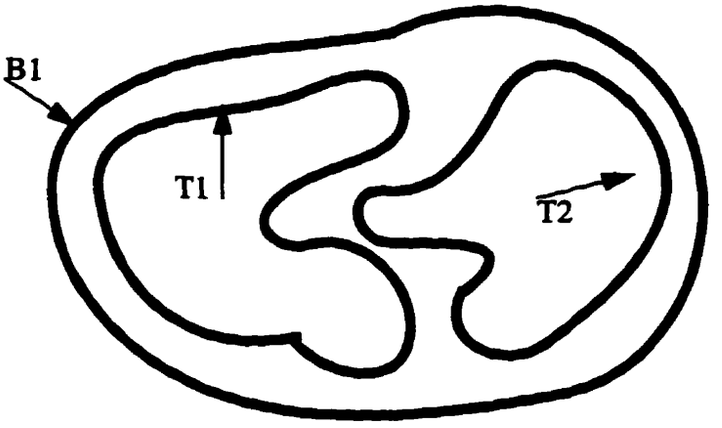

Figure 4.7: Branching slices.

Another problem which has been overlooked in the previous solution is the contour mapping problem. If in slice A we have 3 contours and in slice B we have 4 contours, which contour in slice A should be connected to a contour in slice B. This problem is solved in the following hybrid solution.

4.2 Results of the primitive solution.

The primitive solution is tested using hand-drawn synthetic slices. The slices contain simple figures such as a circle and a line. The slices and their reconstruction are shown below (figures 4.8, 4.9 and 4.10). In those figures, the gird lines are displayed so that the relative centers of masses, marked with crosses, and the differences in diameter could be identified visually. The spacing between the lines is 0.5 inch on paper. Notice that in figure 4.10, the reconstruction of the 3 circles is not a cone as it is expected because of the difference in the center of masses of the 3 circles.

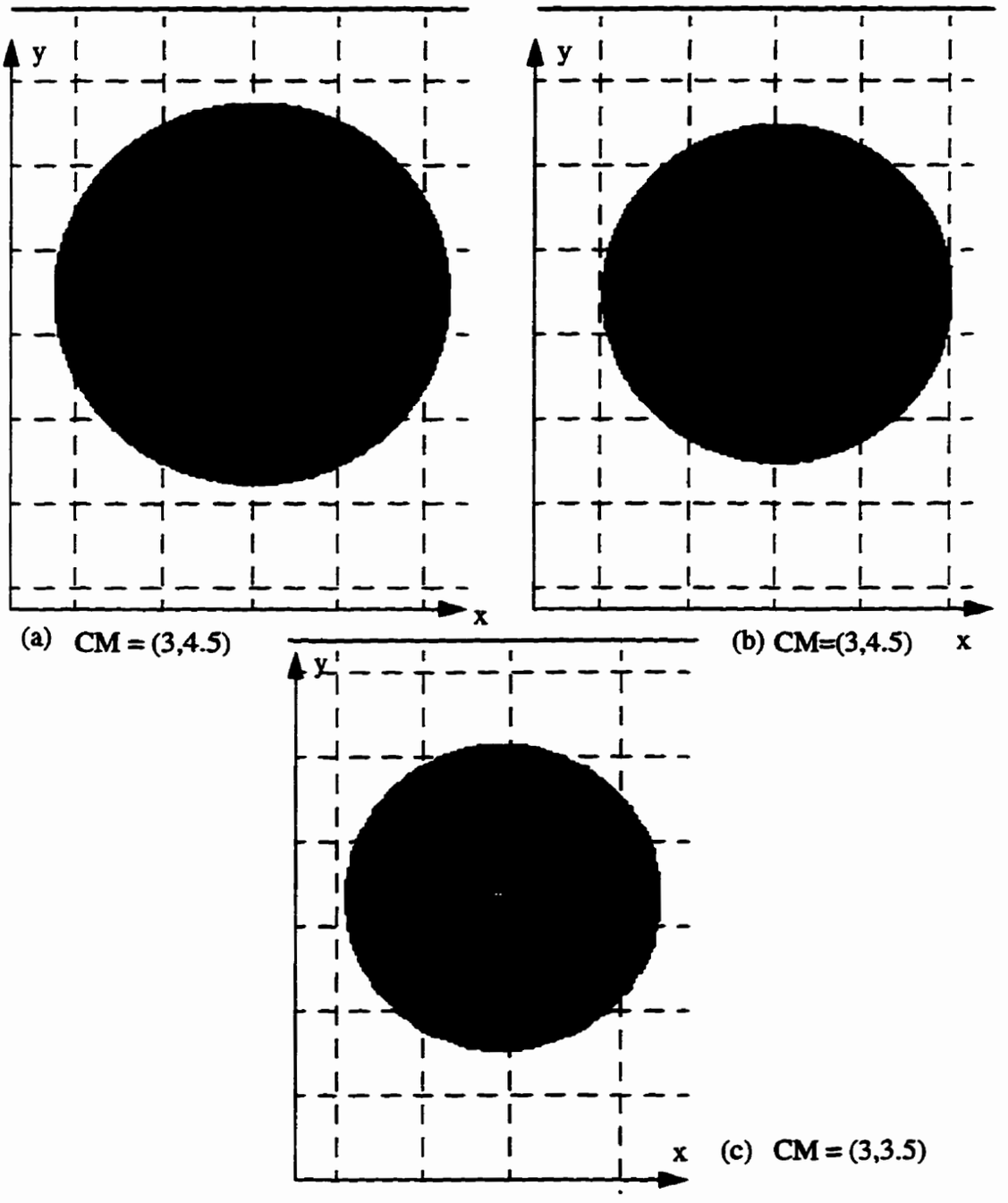

Figure 4.8: (a) First slice at depth value of 2. Notice the thin line at the top of the image. (b) Second slice at depth value of 30. This circle has a smaller diameter. (c) Third slice at depth value of 60. Again the diameter is smaller.

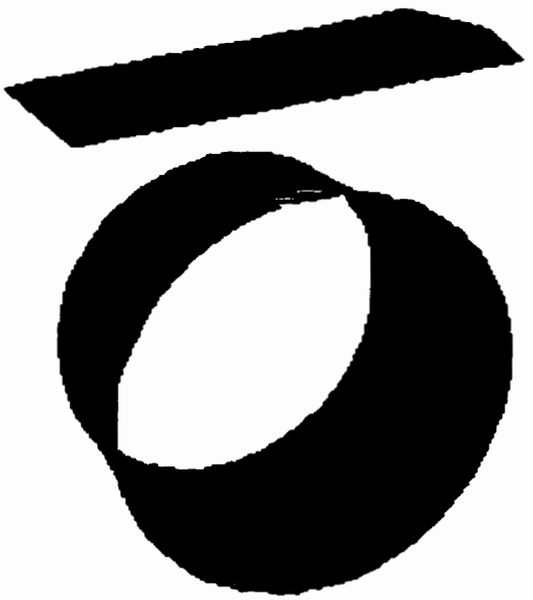

Figure 4.9: Shaded reconstruction of the previous 3 slices. Notice how the lines in each image were reconstructed as a plane in 3-D space.

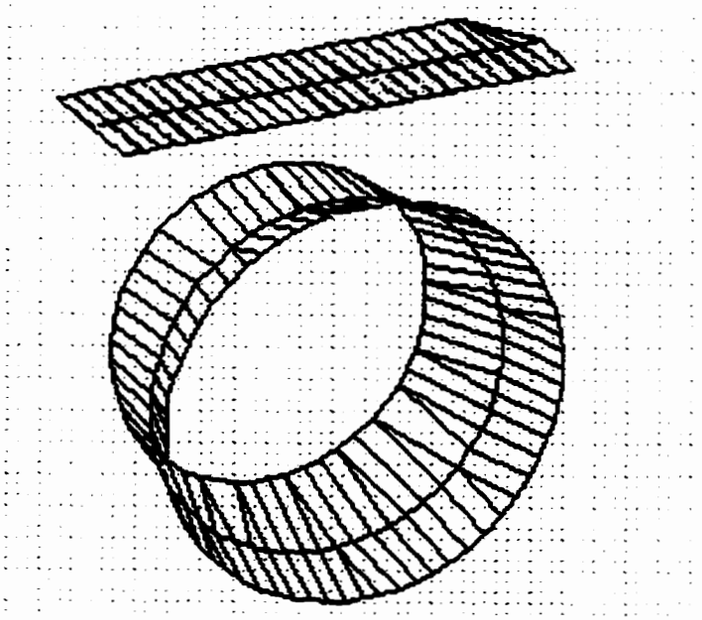

Figure 4.10: The reconstructed mesh.

Chapter 5 A 3-D reconstruction solution

The newly developed algorithm is explained in detail in this chapter. The only technique which was borrowed from elsewhere is the Ganapathy triangulation method although the initial step in that algorithm was modified in our implementation to enhance the quality of the produced image. This modification is explained in Section 5.3. Also, the contour mapping algorithm is very similar to the one presented in [Ekoule 1991] but different parameters are used here for the reasons suggested in Section 5.2.

5.1 Area threshold.

After thresholding, some black regions (where a black region is considered an object) will result because of noise in the original image. Therefore an area threshold is introduced so that regions which have areas less than the area thresholded will be discarded and they will not be processed further. That is, they will not be considered for

contour finding, tracing and triangulation. Consider the CT image of the lower jaw shown above in Chapter 4. If this image is thresholded, the result may contain some noisy regions as shown in figure 5.1. If an area threshold is used, these noisy regions will be eliminated in the segmentation process. Thus, a clear image is obtained. Figure 5.2 shows the same image when an area threshold of 30 pixels is used. Notice how the noisy regions are now eliminated.

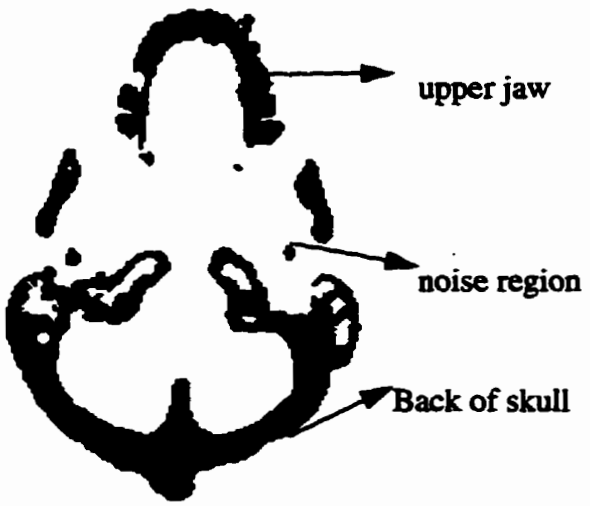

Figure 5.1: Noisy regions resulting from thresholding operation. Area threshold is 0.

Figure 5.2: Using an area threshold of 30 pixels removed all the noisy regions.

5.2 Contour mapping.

The contours of every two slices are mapped into each other in order that triangulation should take place between the contours of the same object. A feature vector is filled for every contour. The features used so far are the area of the region that this contour encloses and the center of mass of the contour. A similarity measure is used and it is:

$$Similarity = \frac{a \bullet b}{(a \bullet a + b \bullet b - a \bullet b)} \quad , \qquad (5.1)$$

where a is the feature vector for the source contour and b is the feature vector for the destination contour [Parker 94]. Notice that the similarity measure ranges from 0 to 1. If the value of the similarity measure is above a certain threshold, say 0.8, the corresponding location in a boolean matrix is set. For example, if we have 4 contours in the source slice and 5 contours in the destination slice, the mapping matrix might look as the one shown below. A value of one at any location means that the two contours at this row and column are similar and they should be connected. Therefore, in this matrix, we have 4 situations.

- Source 1 is branching to Dest 1 and Dest 2
- Dest 3 is branching to Source 2 and Source 3.
- Source 4 is connected to Dest 4.
- Dest 5 is the beginning of an object. (conversely, if any contour in the source slice is not related to any contour in the destination slice, it would have been the end of an object).

	Source 1	Source 2	Source 3	Source 4
Dest 1	1	0	0	0
Dest 2	1	0	0	0
Dest 3	0	1	1	0
Dest 4	0	0	0	1
Dest 5	0	0	0	0

Having a high similarity threshold will prevent different contours from being connected (i.e. the mapping matrix will have less 1's than required) and having a low one will allow more contours to be connected together yielding a more connected model (i.e. the mapping matrix will have more 1's than required). So, care should be taken while choosing a similarity threshold for a certain class of images.

Other contour mapping algorithms have been mentioned in the literature but they did not seem adequate to the application at hand. In [Ekoule 1991], a similar mapping matrix was presented but the similarities between the contours where in terms of how much they are intersecting when they are projected onto each other. This seemed to be inaccurate because of the fact that non-intersecting small contours might be connected to each other in reality and using this mapping procedure will prevent that from happening. Also in [Meyers 1992], the authors suggested that the best way to map contours onto each other is to find the best-fitting ellipse for each contour and then comparing how far the different ellipses, on consecutive slices, are from each other in terms of the orientation of the main axis, the center of mass, the major axis and the minor axis. This is unnecessarily complicated for our application since the orientation of the contours is assumed to be the same. Thus, no need for comparing orientation angles between

contours. Also, the process seems to be very slow compared to ours.

5.3 Triangulation.

A variation of the Ganapthy algorithm described in Section 2.2.2 was used. Instead of minimizing the absolute difference between the ratios of parts of the 2 contours already examined to the perimeter of the contour itself, we minimized the absolute difference between the ratios of number of sampled points already examined to the number of sample points for each contour. Notice that the two algorithms are equivalent since the number of sample points already examined is equivalent to the perimeter of the contour already examined and at the same time the total number of sampled points is equivalent to the total perimeter of the contour.

The algorithm starts with choosing the sampled points (P1 and P2) with minimum x value in both contours. Then two triangles, P1 - P1+1 - P2 and P1 - P2 - P2+1 are examined. The one which yields the smaller absolute difference between the ratios of the number of sampled points examined to the total number of sampled points for each contour, is the one which is chosen for triangulation. The algorithm proceeds in the same manner until the initial points P1 and P2 are reached again.

In [Ganapthy 1982] the authors mentioned that the first two starting points (one on each contour) to start the triangulation with are the left most sampled points (points with minimum x-axis value) in each contour. This happened to be not accurate enough when the two contours to be triangulated are of totally different shapes. In this case the reconstructed shape looks twisted and the triangles happened to have a large surface area although we are looking for minimum surface area triangles since we should be

connecting the closest points to each other. Thus, an optimal technique was used to account for this inaccuracy[1]. A point X_0 is selected from the source contour and then Euclidean distance is measured from X_0 to Y_j another point in the destination contour. Then the distance between X_1 and Y_{j+1} is measured and added to the previously calculated distance and so on for all the points until we reach the end of one of the two contours. The process is repeated starting from another Y_j at the destination contour. The process keeps on going until we started from all the possible points on the destination contour. The Y_j which yields the minimum total line segment lengths is chosen to be the most correct point to start triangulation from with X_0. This process could be formulated by this formula.

$$Minimize\left(\sum_{j=r,\, i=0}^{n,\, m} ED\,(X_i, Y_j) \right) \qquad \forall \ r \qquad (5.2)$$

Where n and m are the numbers of points in the destination and source contours respectively, *ED* is the Euclidean Distance measure between 2 points in 3-D and r is the initial point in the destination contour. The Y_r which gives the minimum for that formula is the one which should be used to be connected to X_0 and the process continues to connect all the other points using the Ganapthy method.

The complexity of the triangulation process is simply related to the number of points in both contours (assuming no branching) and it is O(n+m) where n and m are the numbers of points in the source and destination contours respectively. If n is equal to m, then the complexity of the triangulation process is O($2n$) which is O(n). This is a characteristic of

1. A very similar technique was used in [Ekoule 1991] to find closest points on two convex contours. The only difference is that in [Ekoule 1991], the authors kept all closest pairs for later use in the triangulation process while we keep the first pair only. Also, in [Ekoule 1991], the authors did not mention how they did measure the distance between points.

all the heuristic triangulation algorithms because all the sampled points are handled once during the process of triangulation. The choice of the first pair of points, one on each contour, is a little bit more involved. Since, for each point in the source contour, we are testing it against all the points in the destination contour, then the complexity of that algorithm is O(nm). If n is equal to m, then it is O(n^2). Thus, we can say that the overall complexity of the triangulation process is O(n^2).

The Christiansen triangulation algorithm would not have worked for the following example. If we have two contours with distant center of masses, as shown in figures 5.3, 5.4, 5.5 and 5.6, all the points at the first contour will be connected to one and only one point in the second contour. This point is the *closest* to all the points in the first contour and it is the point at the bottom of the second contour. Therefore, a heuristic approach for triangulation which depends on a distance measure, say Euclidean, will never work for distant center of masses and unfortunately this situation can easily occur in many applications.

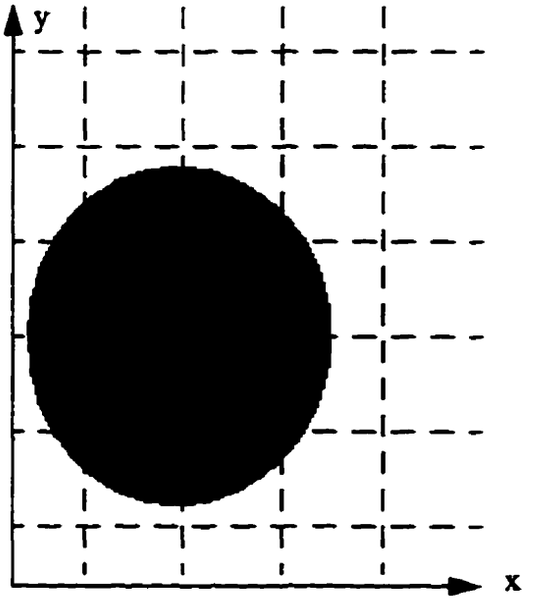

Figure 5.3: Circular artificial slice.

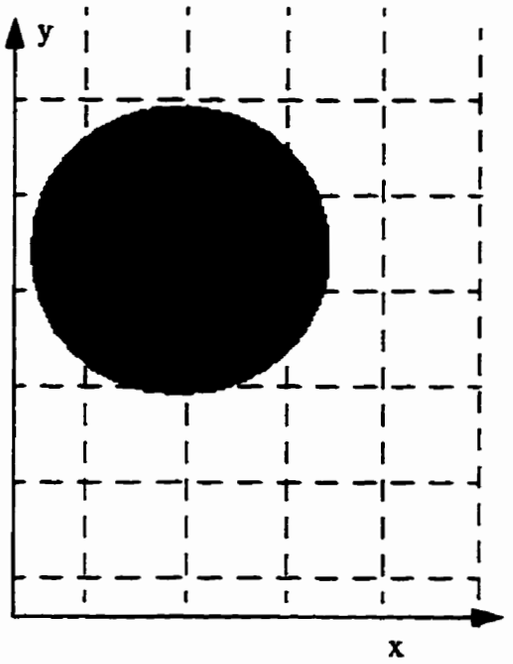

Figure 5.4: Another circular artificial slice at a far center of mass.

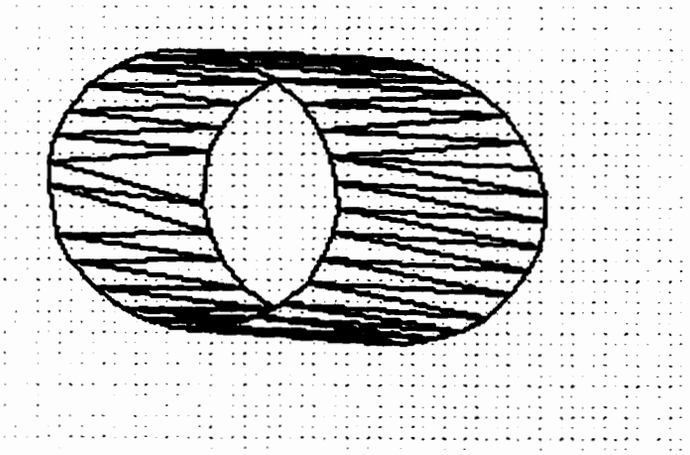

Figure 5.5: Correctly triangulated surface.

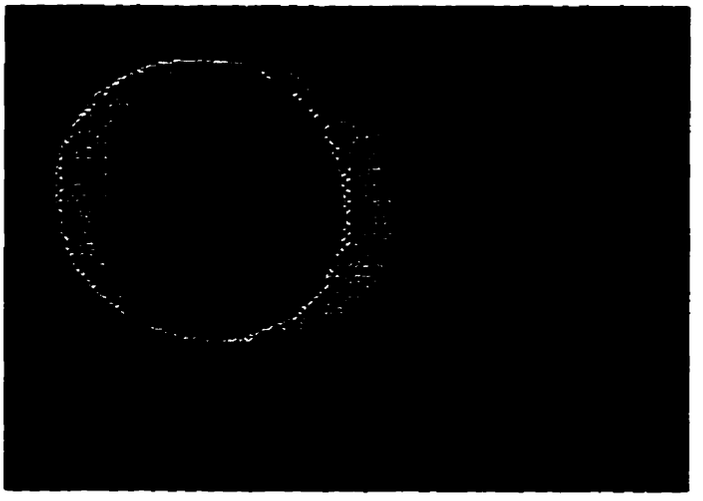

Figure 5.6: Correct triangulation. Shaded.

5.4 Branching.

Because of the inability of the branching method described in [Christiansen 1978] to handle difficult branching cases we developed a very simple alternative method which depends totally on the contour mapping algorithm described above. When we find a 1 at any location in the contour mapping matrix we connect the corresponding contours. So, if contour x, in the source slice, is mapped to contours y and z, in the destination slice, then contour x is triangulated with contour y and then contour x is triangulated again with contour z. The produced triangles will overlap at the point of branching giving a more realistic representation (figures 5.7, 5.8, 5.9 and 5.10). A direct consequence of the above branching method is that some surfaces will be formed inside the object itself. For example, if the inside of figures 5.9 or 5.10 is examined, extra surfaces will be found. This could be considered as an artifact but as long as the use of the developed method is restricted to reconstructing the surfaces of underground cavities then this algorithm is usable.

The complexity of the triangulation process is not changed for a branching body. If we put the algorithm for choosing the first pair of points aside, we can see that the triangulation process, in the branching case, is O(k*n*+*m*). This is when a source slice having n points is triangulated with k contours on the destination slice each containing m points. Again, if *n* is equal to *m*, then the whole process is O((k+1) *n*). Assuming that k is not very large (between 2 and 20), then the algorithm is linear (O(*n*)). Taking the initializing algorithm for choosing the first pair of points into consideration, we can say that the whole triangulation process, including branching, is in the order of $O(n^2)$.

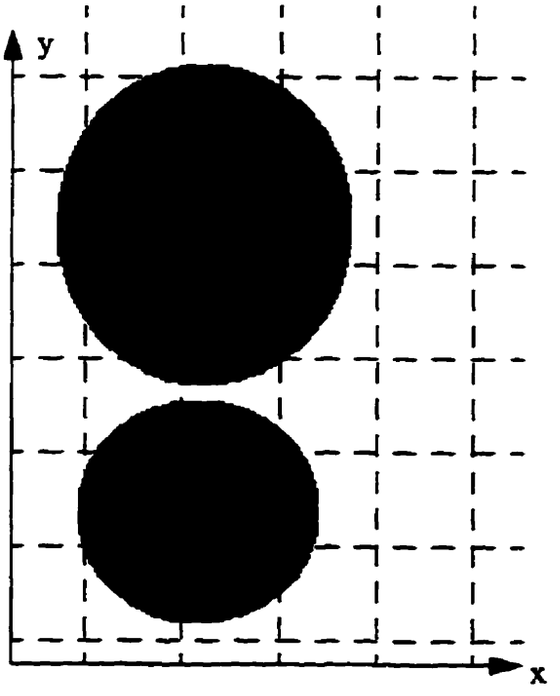

Figure 5.7: Replacing the slice in figure 5.4 with this slice.

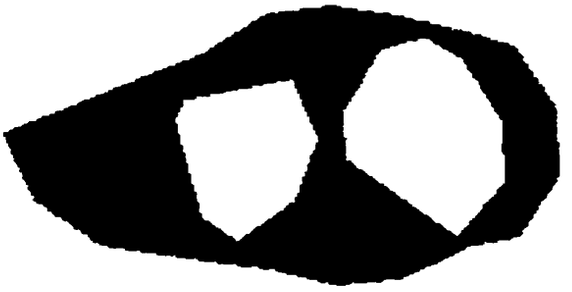

Figure 5.8: A shaded branching object (Top view).

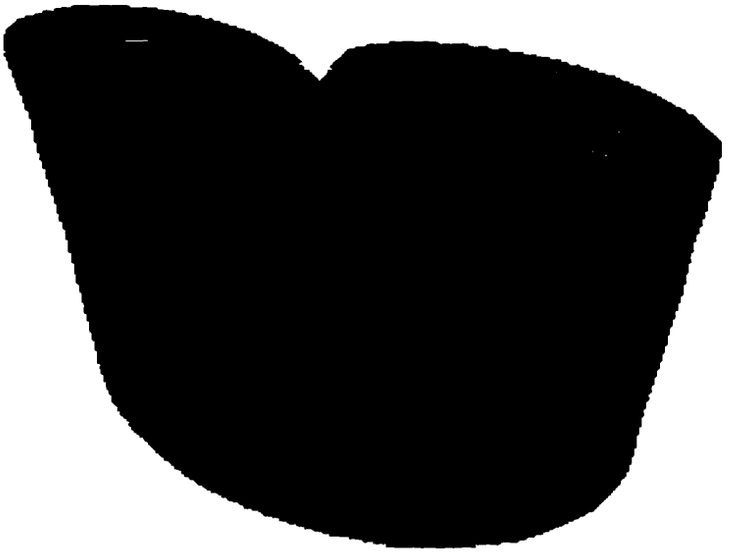

Figure 5.9: Branching from 1 circle to 2 circles using the interpolation algorithm described in Section 5.7 below. Notice how smooth is the reconstructed object compared to the object in figure 5.8.

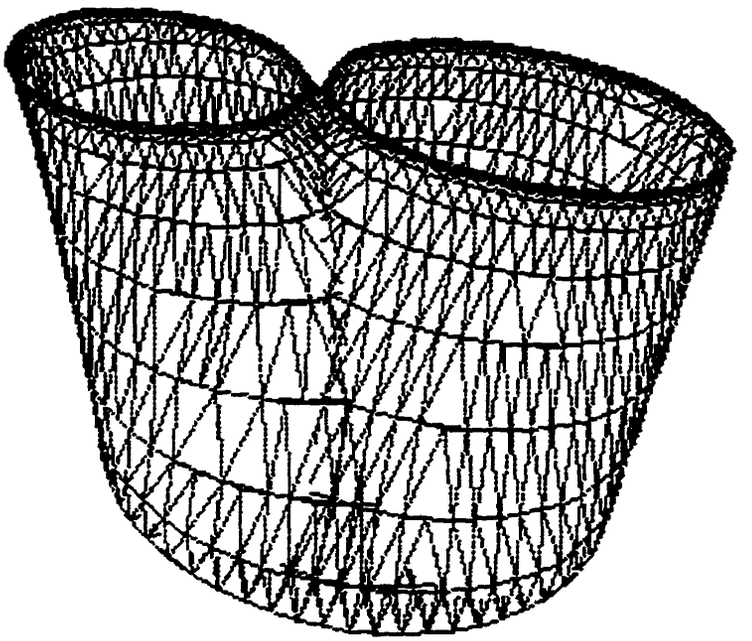

Figure 5.10: The triangular mesh of the object in figure 5.9.

From Section 2.2.3 and Section 4.1.7 it was very clear that the Christiansen method of branching is not adequate when the contours have awkward shapes. Also, it was clear from Section 2.2.3 that Ekoule's solution was very slow and Meyer's solution was not fully automated. Thus, the new algorithm here is, in this case, superior to Christiansen's, Ekoule's and Meyer's. Also, we compared our method to the Delaunay trinagulation method which takes into consideration branching cross-sections. Figure 5.11 shown below is a reconstruction of a branching body (exactly the same as the one shown above in figure 5.10). Notice that in this figure all the points were triangulated even the ones at the same slice (level). This is considered one of the drawbacks of Delaunay triangulation. Notice how the branching point is assumed to be at the lower slice which might not be true in reality. The algorithm used to produce this reconstruction is called Detri[1] .

1. Developed by E.P.Muke at the Los Alamos National Laboratory in California.

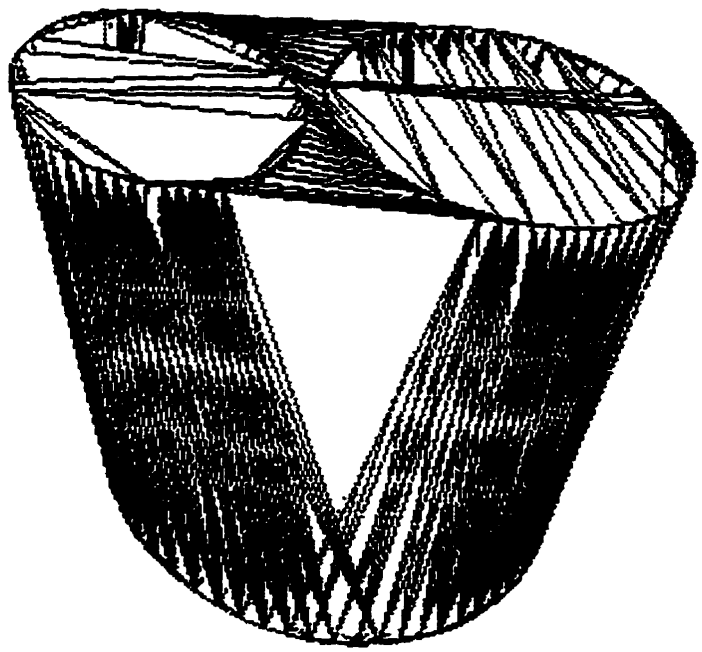

Figure 5.11: Delaunay triangulation of a branching body.

5.5 The algorithm.

The following is an outline of the algorithm used to implement the hybrid algorithm. Read the comments for more detailed explanation.

```
/* Read slices data file where the first integer in the file is the number of slices
and then slice   filename and depth in the Z direction alternate. */
f = fopen(argv[1],"r");
if (f == 0) exit (1);
k = fscanf (f,"%d", &slices); if (k<1) exit (1);
k = fscanf (f,"%s",name); if (k<1) exit (1);
k = fscanf (f,"%d",&Z_pos); if (k<1) exit (1);
```

```
/* Read the image of the first slice */
Image1 = read_image (name);
Threshold (Image1); /* Threshold that image using the Kapur entropy method */
Image1a = (IMAGE) Segment (Image1,&R); /* Find the different regions in
that slice */
Contour (Image1); /* trace the contours for these regions and highlight them*/
Points (Image1,Z_pos,p,&np); /* Produce a list of points on these contours */
i ++;
while (i<slices) {  /* While we can read more slices from the data file */
 /* Read the following slice */
 k = fscanf (f,"%s",name); if (k<1) break;
 k = fscanf (f,"%d",&Z_pos); if (k<1) break;

 if (k<1) break;
 Image2 = read_image (name); /* Read the image of another slice */
 /* Go through the whole process as above to produce a list of points on the
contours of this slice */
 Threshold (Image2);
 Image2a = (IMAGE) Segment (Image2,&R);
 Map_Contours (Image1a,Image2a,Contour_map);
 Contour (Image2);
 Points (Image2,Z_pos,p1,&np1);

 /* Triangulate the two series of points in p and p1 using the method described
above where contours are mapped to each other according to a certain criterion
and the matched contours are triangulated together. */
 Match_points (argv[2],Contour_map,p,np,p1,np1);
```

```
   /* Copy the destination slice (the second one read) to the source slice and
repeat the whole process */
   copy (&Image1a, Image2a,1);
   /* Also copy all the produced points of the destination slice to the arrays of the
source slice */
   np = np1;
   for (v=0; v<np1; v++) {
      p[v][0] = p1[v][0];
      p[v][1] = p1[v][1];
      p[v][2] = p1[v][2];
   }
   i++;  /* increment number of slices read */
  } /* End of while loop */
```

The list of constants/parameters which are used in the algorithm are:

SAMPLE_STEP	Number of pixels between each contour vertex - See CONSTANT_POINTS below. Typical values are between 3 and 15.
MPP_CONTOUR	Maximum points per contour. A maximum of 20000 was used.
MAX_OBJECTS	Maximum objects in a slice - Has to do with the 4's used everywhere in my programs. If it is desired to change this number then all the 4's in my programs should be changed. A value of 60 is used here.
MAX_FEATURES	Number of elements in the feature vector. A vector size of 10 elements is used.

SIMILARITY_THRESHOLD	Threshold above which the contour is accepted. Range from 0 to 1.
AREA_THRESHOLD	To eliminate small objects. Typically between 10 and 1000.
INTERPOLATE_THRESH	This is the maximum Z distance allowed between slices. If exceeded interpolation is used. See Section 5.7 below.
VSP	Either 0 or 1. When VSP = 0, the numbers in the slices data file are depth values in the Z direction for each slice. Otherwise, these figures represent the radial angle between each subsequent slices ---- See Section 5.8 below for a detailed implementation of the VSP algorithm.
CONSTANT_POINTS	This is the constant number of sample points per contour which is used when we use interpolation and it is an alternative of the SAMPLE_STEP. A value of 60 points per contour was used in most of the experiments.

5.6 CThead algorithm.

The following subsections explain how the pre-mentioned algorithm is modified to be used in reconstructing a human skull from a series of CT scans. This was done to evaluate the developed algorithm because of the fact that CT scans are the most widely used input for 3-D reconstruction systems. So testing our system using CT scans seemed appropriate to compare it to other systems.

5.6.1 Removing the area feature.

It was found that when branching is occurring frequently in an object, the area feature

which is used in the contour mapping algorithm will not be of any help but it will disturb the mapping process by mapping unrelated contours to each other. This is because when a contour is branching, say, to two other contours, then the areas of those two contours are in no way similar to the area of the original contour.

5.6.2 Reconstructing a skull.

The algorithm described in Section 5.5 (and modified in the previous subsection) was used to reconstruct a skull from a series of CAT scans. The CT slices where thresholded with a very low threshold value (13). Thus, any pixel value above this threshold was set to 0 (black) and any pixel value below this threshold was set to 255 (white). Therefore, the skull contour shown in slice number 20 was turned into a black object and the black background in the same image was turned into a white background in the thresholded image. The area threshold was 40 (i.e. any region which had an area less than 40 pixels was not considered an object), the sample step was 10 (i.e. choose a contour point every 10 points) and the similarity threshold was chosen to be 0.9. It is desirable to have an inverse relationship between the area threshold and the similarity threshold. This is because as the objects tend to be smaller, the more probable they will be closer to each other and consequently the more closer their center of masses. The opposite case is evident. Figures 5.12, 5.13, 5.14, 5.15 and 5.16 show some slices of a human skull.

Figure 5.12: Slice number 20

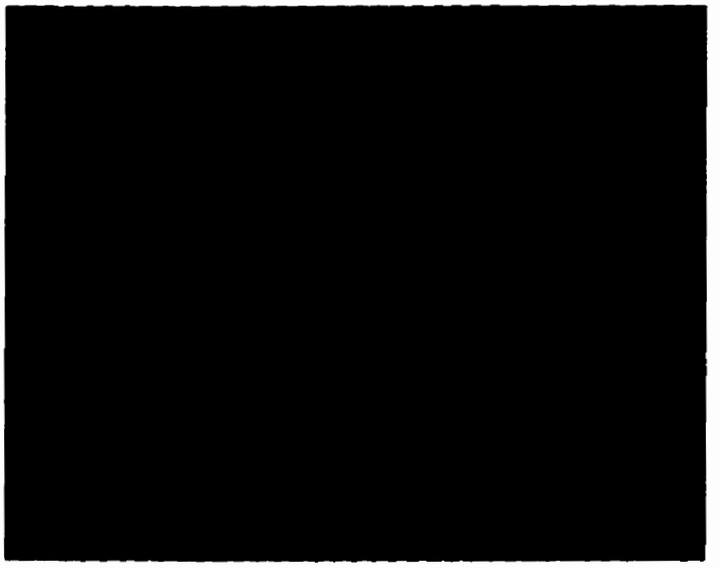

Figure 5.13: Slice number 40. Notice the grooves at the position of the eyes.

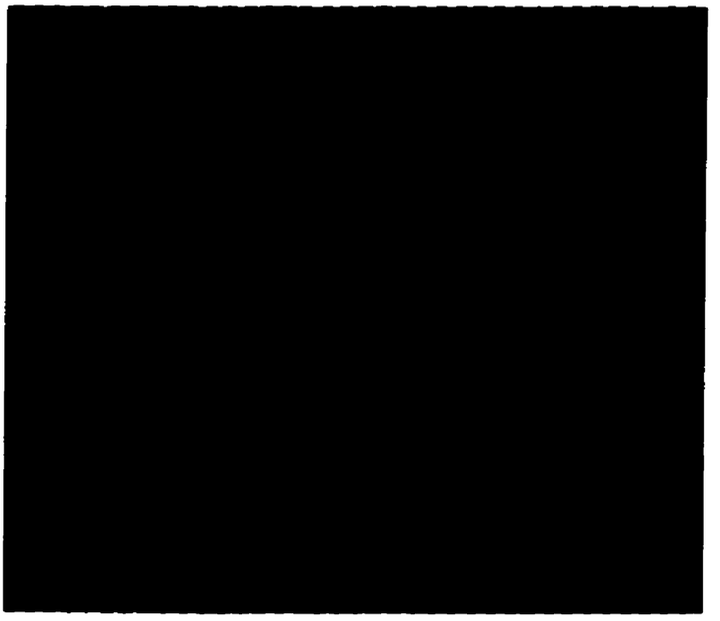

Figure 5.14: Slice number 60.

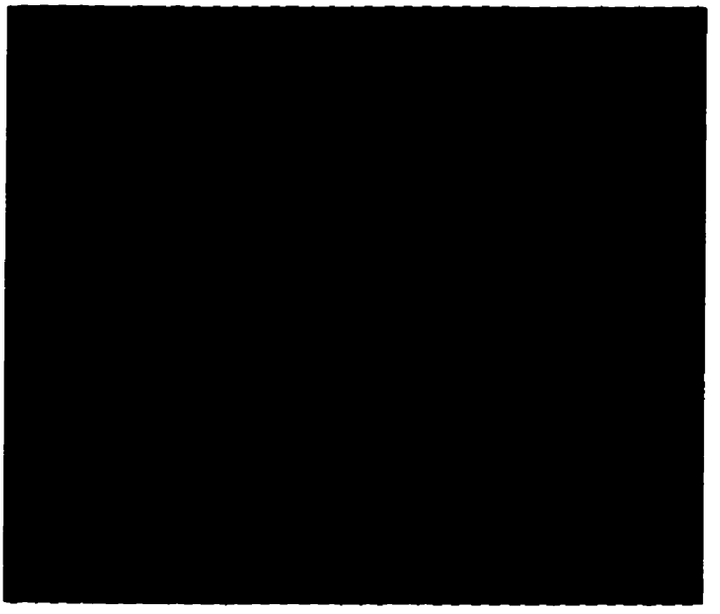

Figure 5.15: Slice number 80. Notice the noise at the top of the image.

Figure 5.16: slice number 100.

The following reconstructed skull images has been rendered using the Open Inventor graphics tool. The points on each contour have been sampled using a sampling rate of 10 pixels such that a sample point (pixel) is chosen every 10 pixels on the contours. The figures show why the improvements of the hybrid algorithm were needed to produce better results in reconstructing the skull.

Figure 5.17: Hybrid algorithm with similarity threshold of 0.8 and area threshold of 40 pixels. Notice how the chin is connected to the spinal because of the low similarity threshold. Also, notice that the top of the skull is not very smooth.

Figure 5.18: Hybrid algorithm with similarity threshold of 0.96 and area threshold of 40 pixels. Notice how the details are clearer.

Figure 5.19: Improved Hybrid algorithm with similarity threshold 0.8 and are threshold of 40 pixels. Notice how the top of the skull is now smoother but because of the low similarity threshold some undesirable triangles exist.

Figure 5.20: Improved Hybrid algorithm with similarity threshold 0.96 and area threshold of 80 pixels. Notice how the high area threshold removed the nose whose area is much smaller than 80 pixels in the slices.

Figure 5.21: Improved Hybrid algorithm with similarity threshold 0.96 and area threshold of 40 pixels. This is the best skull obtained.

The apparent errors at the lower jaw and the bone connecting the lower jaw with the upper part of the skull is due to the fact that the original images are of a very bad quality as seen in figures 5.22 and 5.23. When a constant threshold is used throughout the image, the rays in the CT scans shown passing through air are erroneously considered an object. The only solution to this problem is to make the thresholding process interactive so that the user will be able to detect undesirable objects in the original slices.

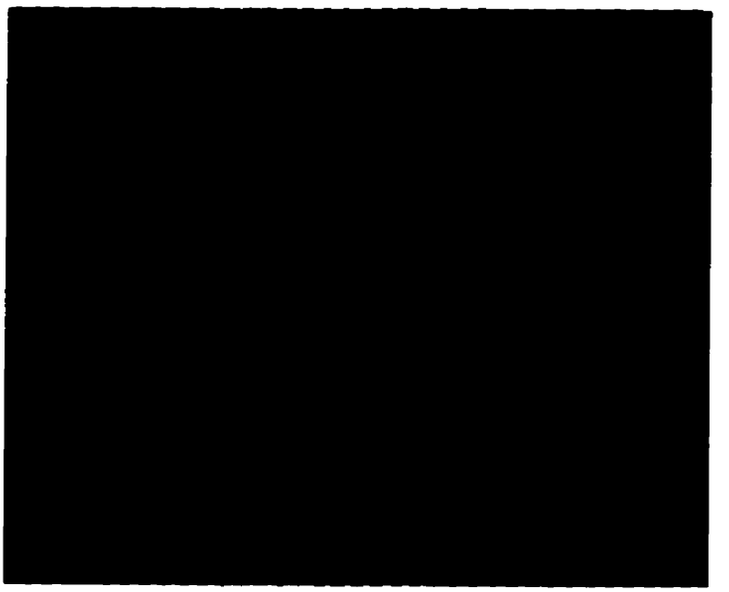

Figure 5.22: Notice the undesirable rays near the lower jaw which are not detectable by an automatic thresholding algorithm.

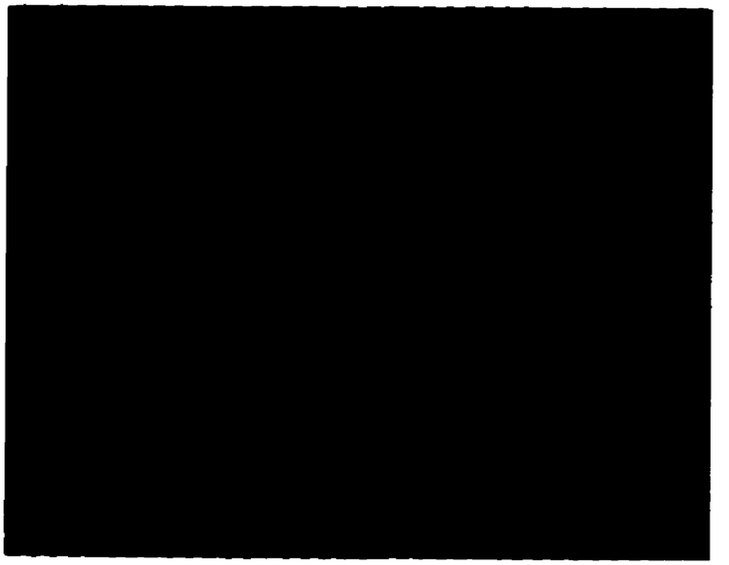

Figure 5.23: The following slice in the CT scans series. The undesirable rays are more evident here.

5.7 Interpolation.

Due to the fact that slices might be distant from each other, an interpolation algorithm is needed. This algorithm is executed whenever the distance between 2 slices is above a certain threshold. The interpolation threshold depends on the spacing of the source/ receiver lines. If the spacing is large then the interpolation threshold should be small enough to account for the missing slices. On the other hand, if the spacing is small compared to the size of the object under investigation then we might not need to interpolate altogether. So in most cases it is advised that a small threshold value be used to enhance the quality of the reconstructed model. This threshold might be equivalent to 25 cms - 50 cms on the surface of the earth since the most common spacing between sources/receivers lines is 1 meter.

The first step in the interpolation algorithm is to locate equal number of points (where the parameter CONSTANT_POINTS, mentioned in Section 5.5 above, contains the value to be used) on each of the two contours to be eventually triangulated. This process is done by measuring the perimeter of each of those contours and the sampling step is now dynamic from one contour to the other and it is equal to the perimeter of the contour divided by CONSTANT_POINTS. The following step is to find the starting point on each contour to start the triangulation with. This could be done using the method described above in Section 5.3 where a certain distance function is minimized. After identifying the two starting points, linear interpolation is performed to produce more slices between the 2 distant contours. The number of new slices is chosen to be

$$NewSlices = \frac{DistanceBetweenSlices}{InterpolationThreshold} \tag{5.3}$$

The finally produced model is smoother than the one produced by just triangulating the two distant contours without interpolation. In the first example shown here, a circle is to be connected to an irregular shape shown in figure 5.24. Figure 5.25 shows the wire frame of the triangulation without interpolation. Notice how the shape of the irregular contour was distorted in the reconstructed figure. On the other hand, notice how the interpolation algorithm enhanced the reconstructed model shown and the produced image is much smoother and the shape of the irregular contour is well preserved in the reconstructed model (figure 5.26).

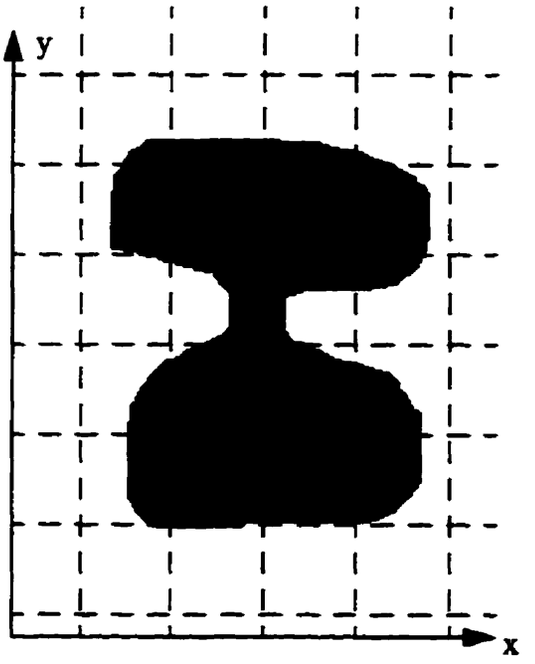

Figure 5.24: Irregular shape to be triangulated with a circular contour.

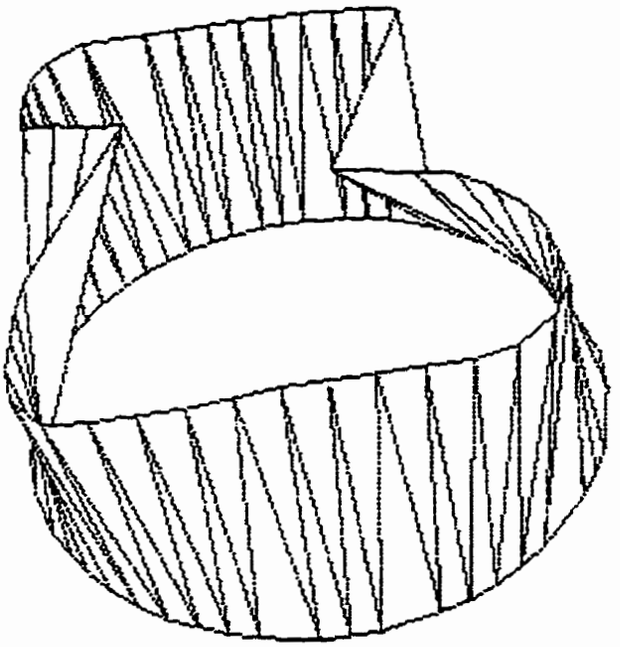

Figure 5.25: Without using interpolation, the model is not smooth and the shape of the irregular contour is distorted.

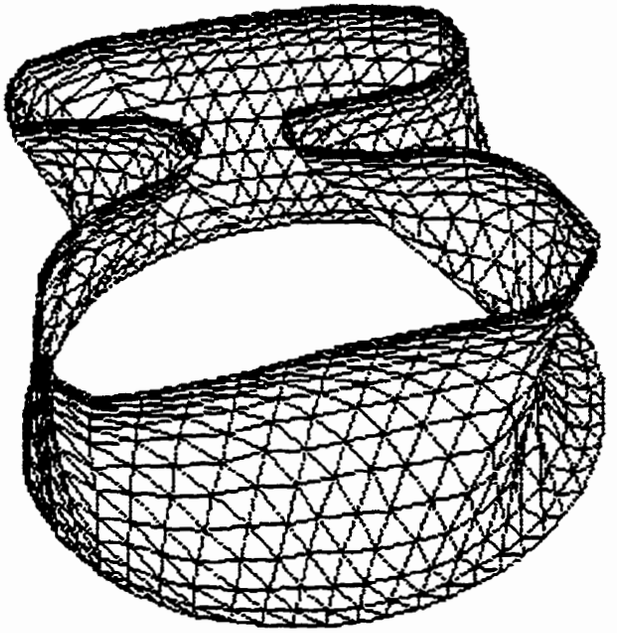

Figure 5.26: Using interpolation. The model is much smoother than the one shown above. The shape of the irregular contour is well preserved.

The interpolation algorithm is very simple and it is easy to recognize the fact that the interpolation process is linear in n (the number of points in a certain contour). But since the algorithm for choosing the first pair of points is used here to find some sort of correspondence between the points, then the complexity of the algorithm is now $O(n^2)$.

When Delaunay triangulation is used to reconstruct that object, the smoothness of the object is not any better as shown in figure 5.27. Notice how the points of the same contour are connected together resulting in a confusing object.

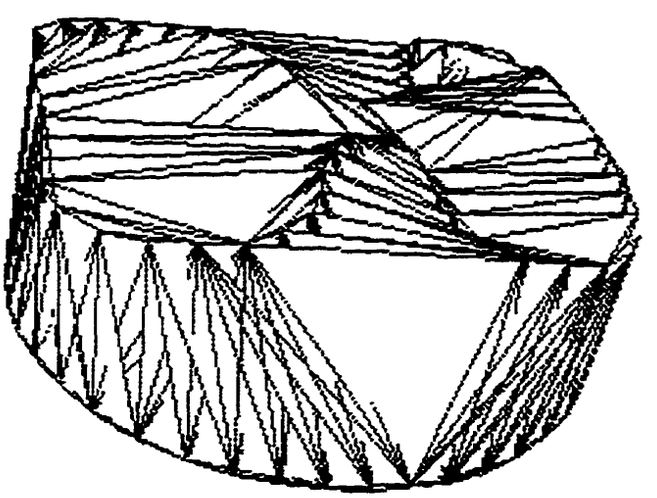

Figure 5.27: Delaunay triangulation of the above two slices.

In the second example shown here, branching with interpolation is performed. Two irregular contours are to be triangulated with a circular contour. The irregular regions are shown in figure 5.28. If interpolation is not used, then the reconstructed model is not smooth and the shape of the irregular contours is not preserved. The opposite occurs when interpolation is used. See figures 5.29 and 5.30.

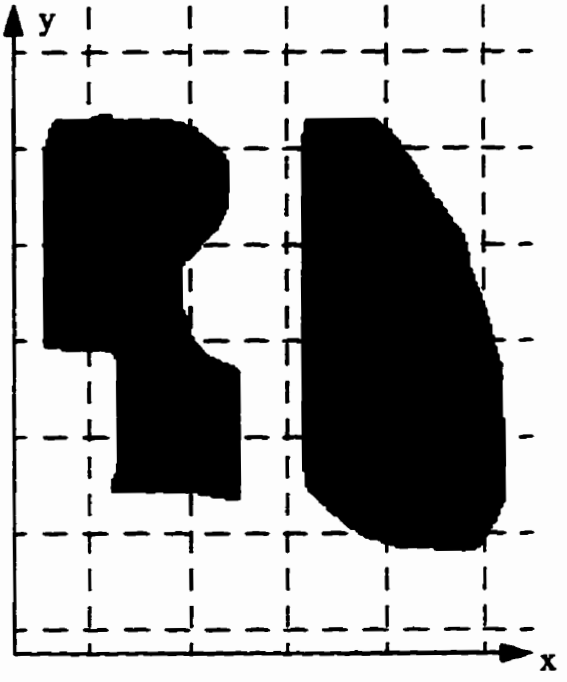

Figure 5.28: Irregular contours used to test interpolation while branching.

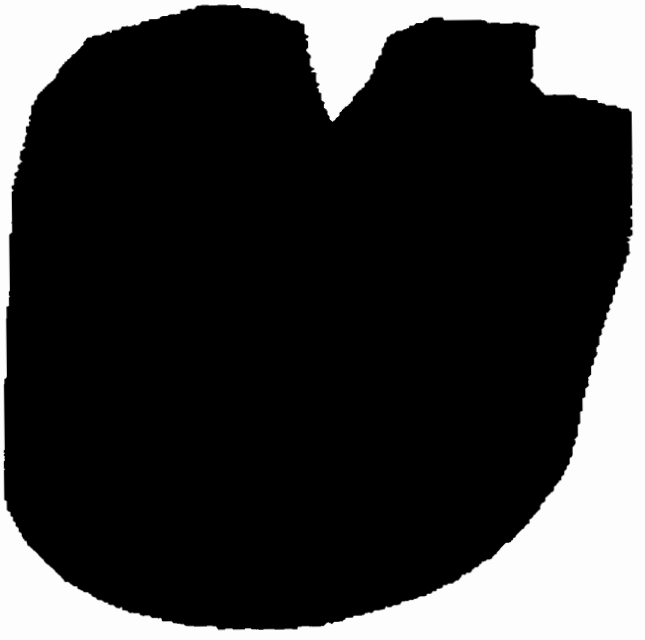

Figure 5.29: Without using interpolation, the model is not smooth enough.

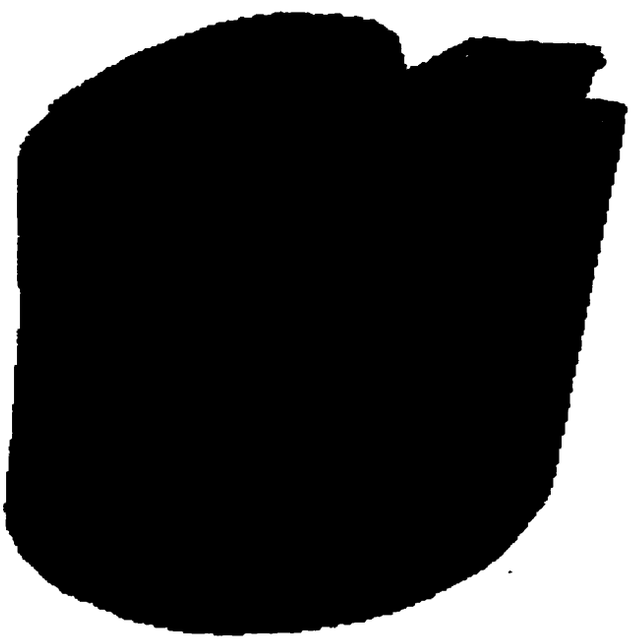

Figure 5.30: Using interpolation enhanced the model smoothness.

The algorithm is straight forward and is shown below.

```
(loop until we have no more matched contours to process) {
/* Get all source points and all destination points for ever pair of contour
combination in the two slices. */
Get_SPDP (Objects++, Objects1, Contour_map, CSP, CDP, SP, DP,
&np_source, &np_destination);

/* If no points were produced then the two contours are not matched and then we
don't triangulate. */
if ((np_source == 0) || (np_destination == 0)) x = -1;
else x = 0;
if (x != -1) {
```

```
/* For interpolation we test if the distance between the 2 slices is greater than a
certain constant and if this condition is satisfied we interpolate. */
if ((DP[2] - SP[2]) > INTERPOLATE_THRESH) {
      new_slices = (DP[2] - SP[2])/INTERPOLATE_THRESH;
      /* Produce as many slices as needed to produce a smooth object. */
      for (slice = 1; slice <= new_slices; slice++) {
       /* Produce new series of points (Linear interpolation between the initial
       source points and the destination points. */
       Interpolate (Source_Points, Destination_Points, New_Dest_Points,
       slice,new_slices);
       Triangulate (Source,New_Dest_Points,out,np_source,np_destination);
       /* copy destination points to source arrays */
       Copy (Destination, &Source);
      } /* for slices */
     }
    else {
      Triangulate (Source_Points, Destination_Points, out, np_source,
      np_destination);
     /* No interpolation */
     }
    }
} /* end loop */
```

5.8 Vertical Seismic Profiling (VSP).

When planar cross-sections are used, the x- and y-coordinates of the points in the finally produced 3-D figure are given by the corresponding coordinates in the cross-section

itself. The z-coordinate is derived from the distance between the cross-sections given in the input data file. Vertical Seismic Profiling is a method used in seismic explorations and it has been described above in Section 3.1. The resulting signals from a VSP survey are a collection of radial cross-sections rather than a series of parallel cross-sections. We developed an algorithm to triangulate these radial cross-sections which is described below.

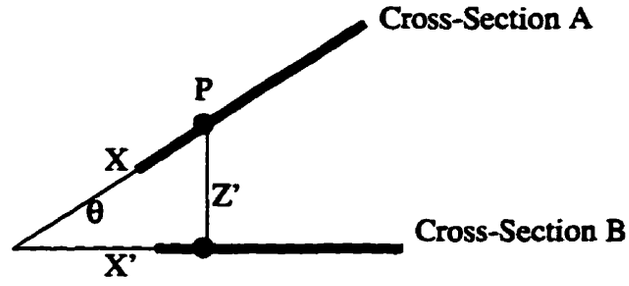

Figure 5.31: The VSP triangulation method.

In figure 5.31, a top view of 2 radial cross-sections is shown. The y-coordinate is given by the corresponding coordinate in the cross-section image itself. θ is provided in the input data file and consequently we can deduce the 2 remaining coordinates X' and Z' using the following equations ($X'=X \cdot \cos\theta$ and $Z'=X \cdot \sin\theta$). Getting the 3 coordinates of all the points required for triangulation solves the problem and any of the algorithms mentioned above could be used to triangulate the points.

Two examples are shown in figures 5.32 and 5.34. The first one shows three circular radial cross-sections with 30 degrees angle between each pair and the second one is an example of a branching cross-section while being surveyed using VSP. The triangulation is smooth enough to identify the objects. Compare these models with the ones produced by Delaunay triangulation in figures 5.33 and 5.35

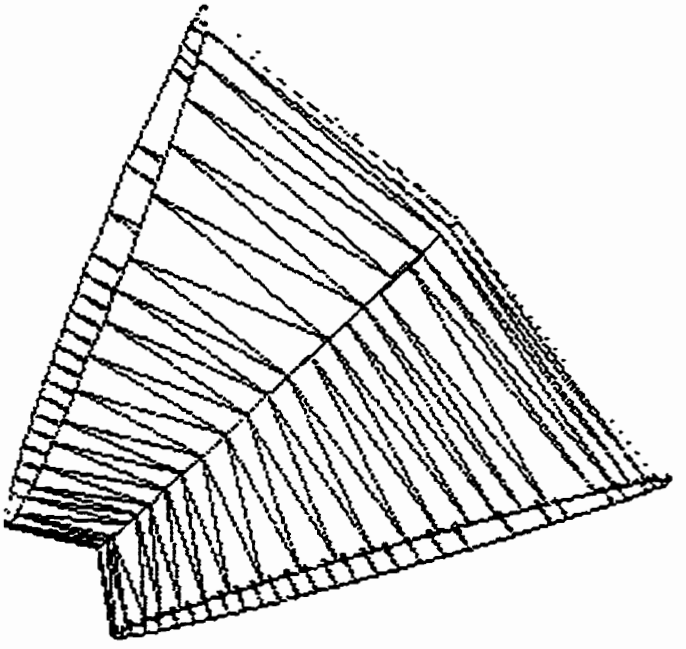

Figure 5.32: Radial cross-sections triangulated using the method described in the text. The angle between each pair of cross-sections is 30 degrees.

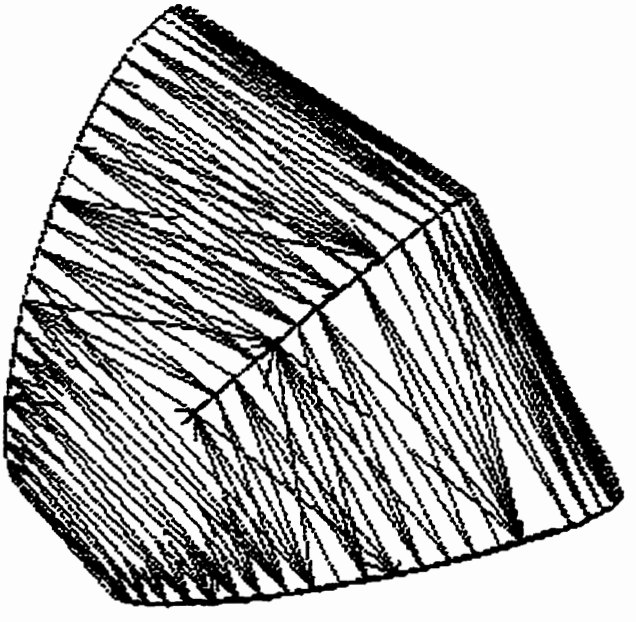

Figure 5.33: Delaunay triangulation of the two radial slices. Very similar to the above figure but here we have more tetrahedra.

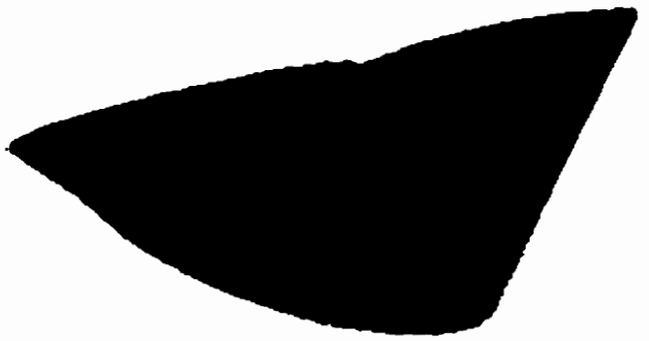

Figure 5.34: A Branching example. The angle between the two cross-sections is 30 degrees.

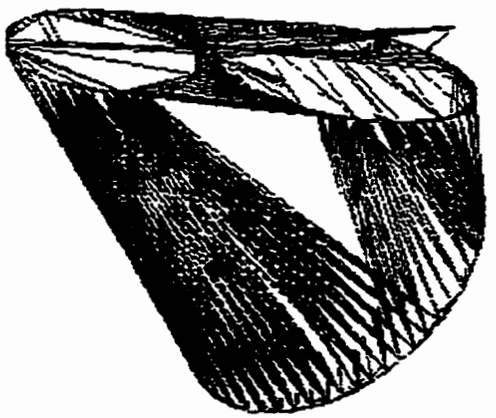

Figure 5.35: Delaunay triangulation of the two radial slices while branching. Notice how the horizontal surface is kept which is inadequate.

Chapter 6 Simulated slices

The following subsections discuss the experiments done using simulated slices. Software packages for simulating seismic surveys and for performing seismic data processing were used.

6.1 GX2 Simulator and the SU seismic data processor.

GX2[1] is a seismic simulation package used to simulate real world cases. Any arrangement of sources and receivers could be implemented and traces could be produced that simulate the real situation. GX2 has been used for that purpose to test the algorithms developed in the previous 2 chapters. The following is a zero-offset survey simulation. In figure 6.1, a trapezium model is defined to be an underground object. The velocity of sound outside the model is 1200 m/s and the velocity of the sound inside the

1. GX2 is a registered trademark of GX Technology, Inc. Houston, Texas, USA.

model is 330 m/s (air). The model's upper surface is at 1 meter depth and its lower level is at 5 meters depth. The model extends for 12 meters horizontally from the 7 meters offset to the 19 meters offset. A zero offset survey was simulated using 101 pairs of sources and receivers. The GX2 produced traces are shown in figure 6.2.

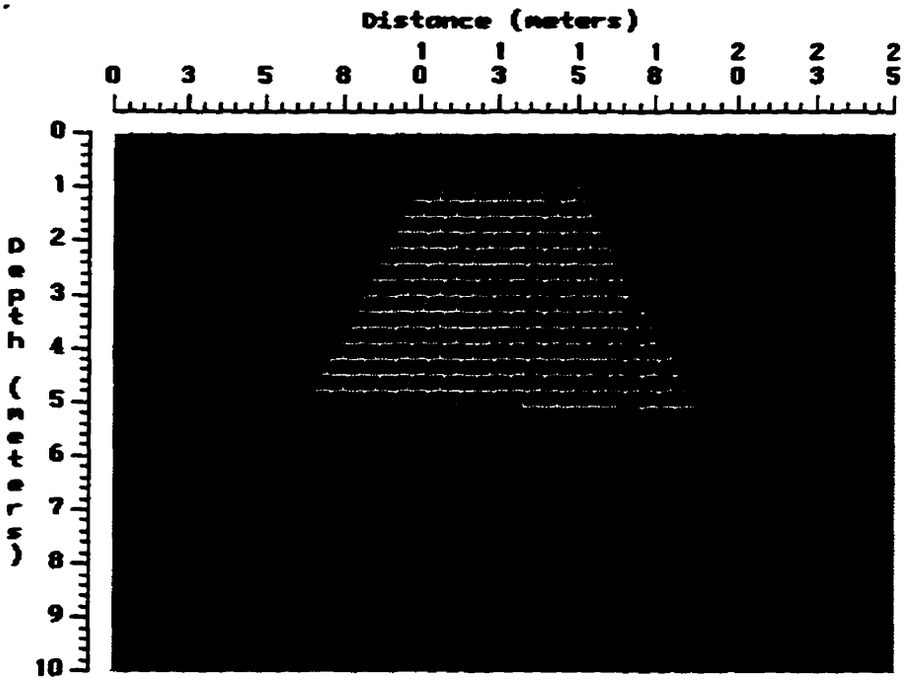

Figure 6.1: A GX2 trapezium model.

In our case here, all the surfaces are reflecting. The vertical axis is a two way travel time, that is the time needed for the sound wave to travel down, get reflected and travel back to the receiver. With simple mathematics the peaks in the traces could be easily justified. When sound travels through a medium with a velocity of 1200 m/s and with an upper surface at 1 meter depth, the upper surface will show reflections at 1/1200 X 2 seconds which is 1.67 millisecond (compare with figure 6.1 - notice that we multiplied by 2 because its a two-way travel). The lower surface will show reflections at a later time

because of the slow velocity of sound through air. The reflections should appear at (1/ 1200 + 4/330) X 2 seconds which is 25.91 milliseconds (again compare with figure 6.1- also notice how the polarity is reversed).

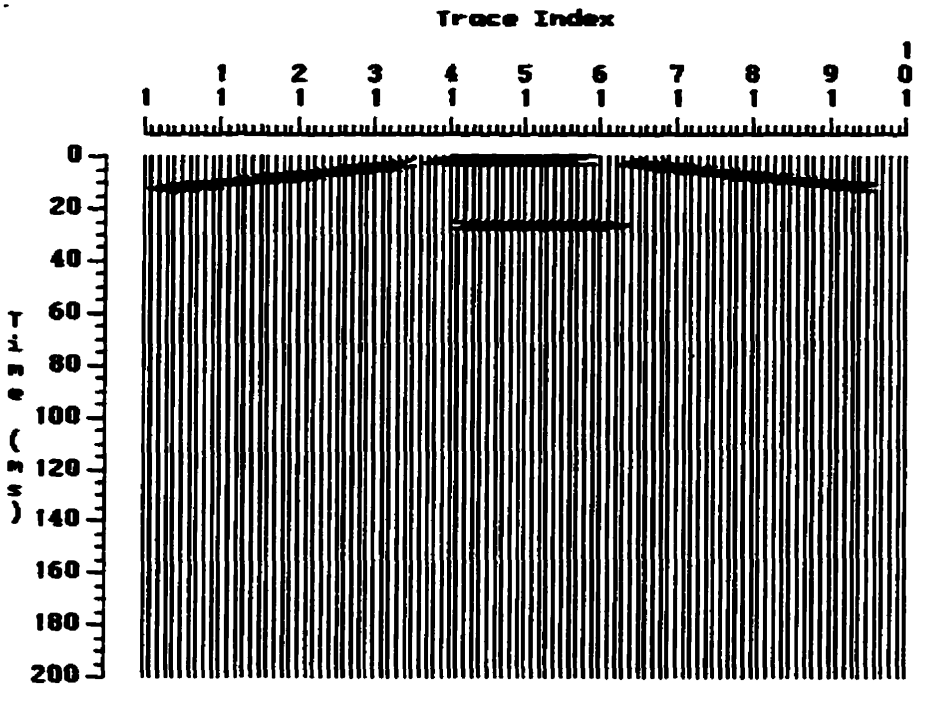

Figure 6.2: GX2 zero-offset traces produced for the trapezium model.

SU[1] is a software package for seismic data processing. It has been used for depth migrating the data coming from GX2. In figure 6.3, the traces of GX2 are imported into SU and the peaks are shown as gray scale shades. Opposite polarities are shown with opposite shades of gray around the mean of 128.

1. Seismic Unix is a seismic data processing package developed at the Colorado School of Mines.

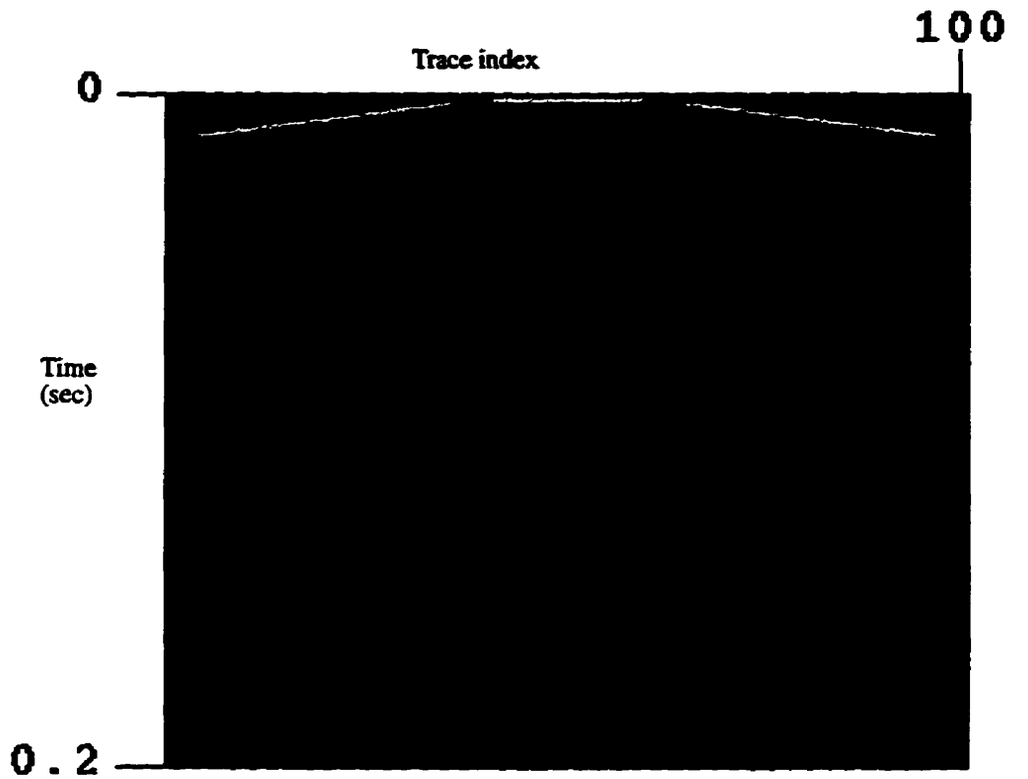

Figure 6.3: An SU gray scale image representing the traces produced. The horizontal axis is pairs of S/R's and the vertical axis is two-way travel times in seconds.

The Kirchhoff depth migration method was performed, using a velocity of 1200 m/s, for the traces shown above to produce the depth model shown in figure 6.4. The velocity model used was not sufficient for the migration program to detect the lower level of the object and this is considered the only drawback of this method. The bright areas in the shown image represent the surfaces of the object in their correct locations under the ground. If we have a series of these images, then a reconstruction of the 3-D model is possible by detecting the bright areas and then triangulating. Detecting bright areas could be done using thresholding.

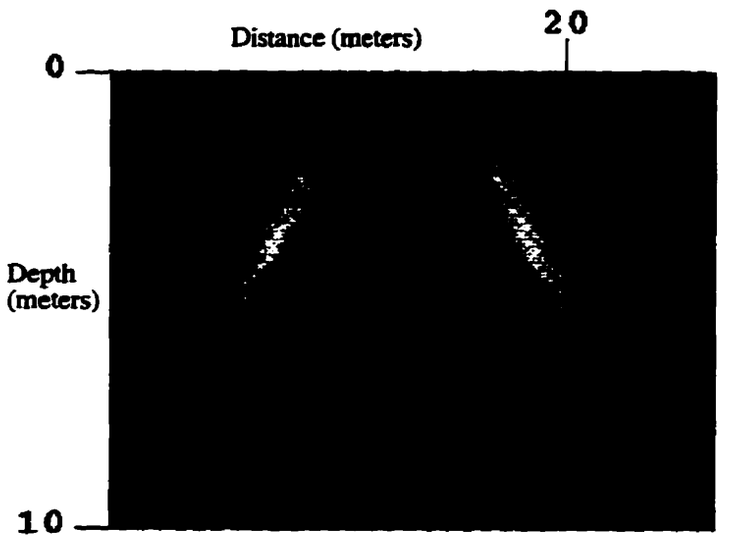

Figure 6.4: A depth migrated image equivalent to the trapezium model shown above. Horizontal axis is meters on the surface of the earth and vertical axis is meters deep in the earth.

6.2 Results of reconstructing 3-D seismic slices.

In the previous section, we mentioned the use of the simulator GX2 and the SU seismic data processing package to produce realistic situations of seismic surveys. Taking duplicates of the image shown in figure 6.4 along the Z-direction and using the hybrid algorithm described above with a THRESHOLD_VLAUE of 140, we produced the 3-D shape shown in figure 6.5. Notice how the model itself is not complete because of the missing lower level (which cannot be recovered) and the discontinuity at the edges of the model. Nevertheless, the model is visually recognizable and the lower level of the model is visually deductible.

Figure 6.5: 3-D Trapezium model.

When noise is introduced to the simulated signals, the produced model is not very clear to identify. Figure 6.6 shows the noisy signals (10 db) produced from a zero offset simulation using the trapezium model. In figure 6.7, the migrated sections are shown. Notice how the bright spots in the migrated section are not as clear as the ones shown above. Reconstructing a series of those sections produced the model shown in figure 6.8 which is not as smooth as required. It is to be noted that the upper surface is the only surface which has been detected using a threshold value of 210. As was shown above, the upper surface is sufficient to identify the underground surface. Another experiment was done by using the same threshold value but the slices were smoothed using a 7X7 window before being processed. The image in figure 6.9 was the result of smoothing the original image. This yielded the model in figure 6.10 which is much smoother than the previous one.

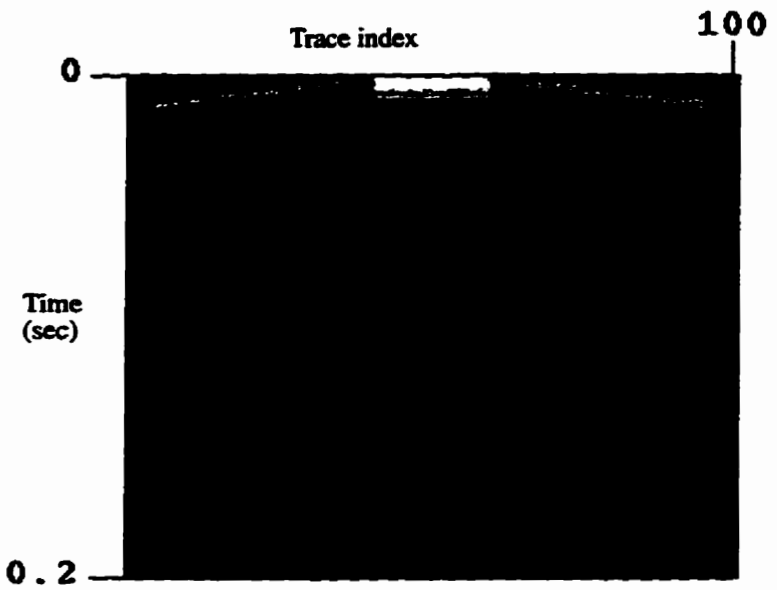

Figure 6.6: Noisy signals from the trapezium model.

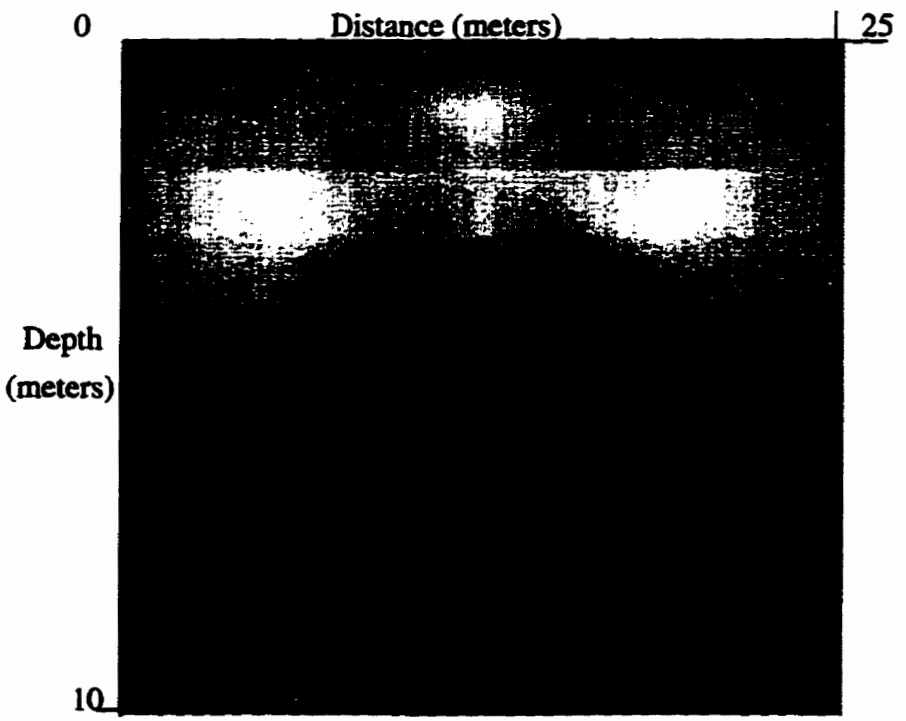

Figure 6.7: Migrated section. The bright spot represents the upper surface of the trapezoidal cavity.

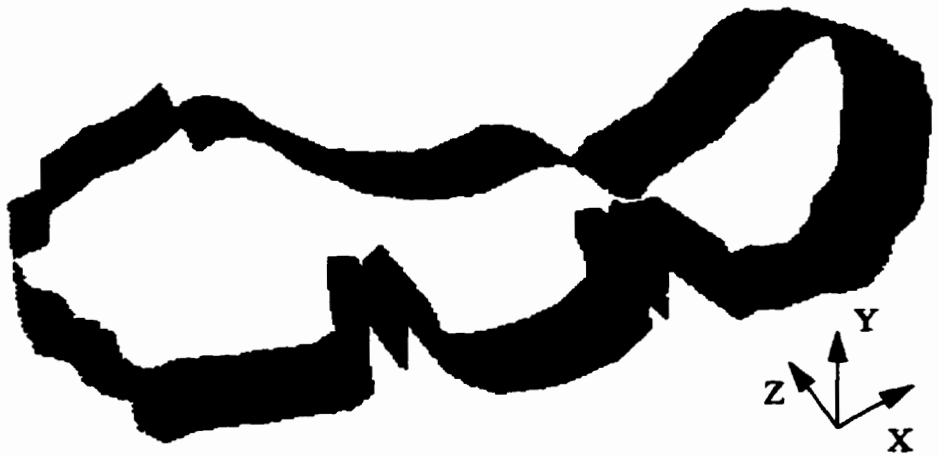

Figure 6.8: Reconstructed model using a series of the section shown above.

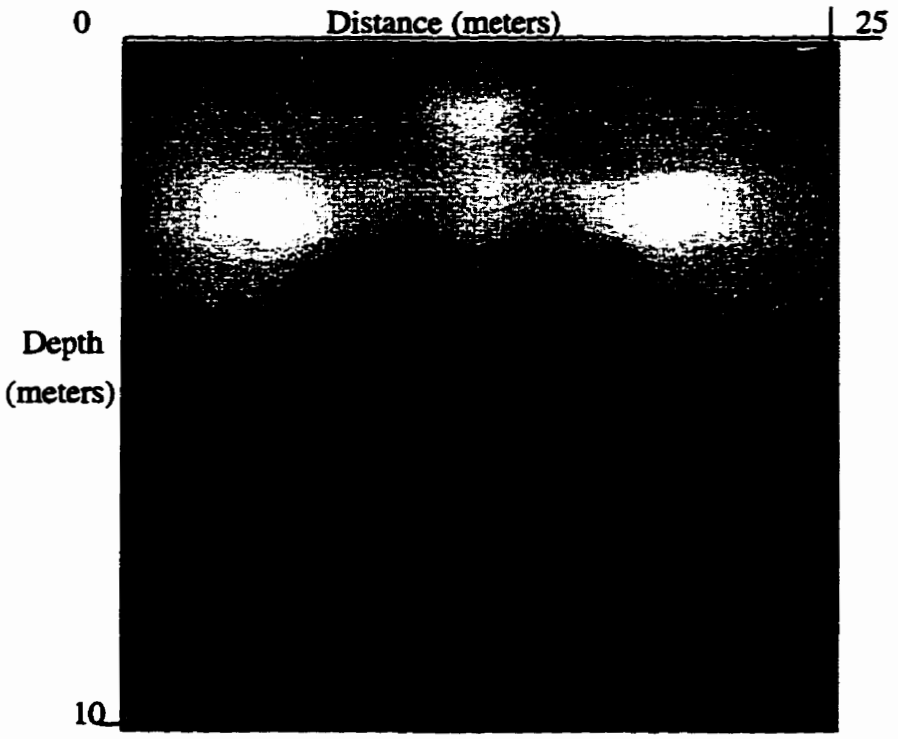

Figure 6.9: Smoothed migrated section.

Figure 6.10: Reconstructed model from the smoothed section.

6.3 Tut Ankh Amen's Tomb.

The following figures show the experiment done to reconstruct a 3-D model from a simulated series of sonar slices of the Tomb of Tut Ankh Amen. Figure 6.11, which is borrowed from [Parker 1996b], show the actual shape of the tomb from a cross-sectional view and a top view. These two figures were used to produce the simulated slices using GX2.

Ten equally spaced slices have been modeled using GX2. In each slice a cross-section taken horizontally in figure 6.11 is modelled using GX2. Both the simulated slices and the corresponding SU Kirchhoff depth migrated slices are shown below. Slices 6 and 7 (falling in the Sarcophagus chamber) were simulated using one slice which is used twice in the sequence. Each GX2 slice is followed by the corresponding migrated slice and in each of those migrated images the horizontal axis is offset above the ground and the vertical axis in depth inside the earth. A zero offset simulation was used to produce

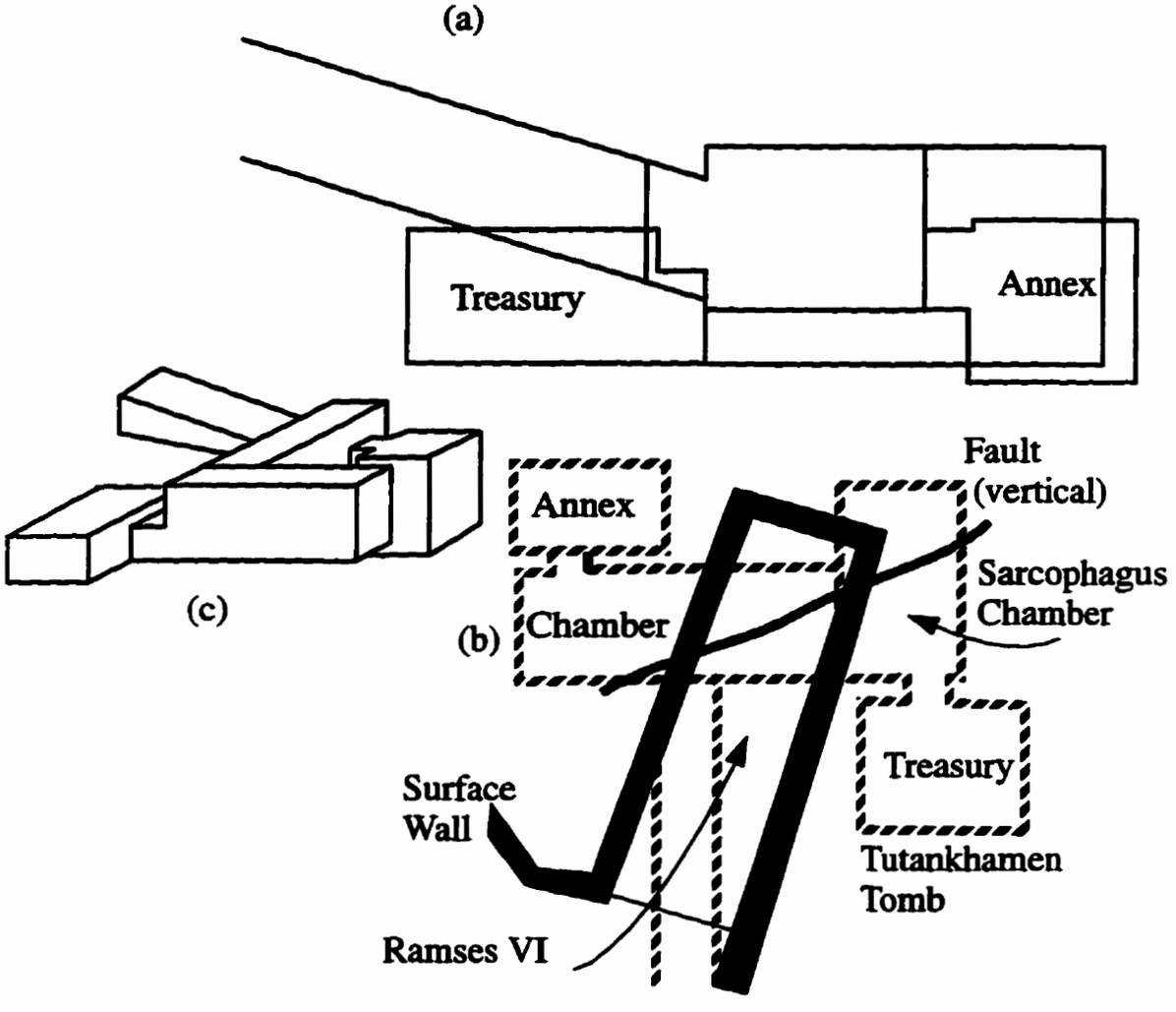

Figure 6.11: (a) Cross-sectional view of the tomb of Tut Ankh Amen. (b) Top view of the tomb; the dashed lines are the tomb walls. (c) A rough 3-D sketch. [Parker 1996b]

ultrasound signals. The spacing of the Sources / Receivers pairs was 25 cms. Thus, we have 101 S/R pair along the 20 meters line (see figures below). Those signals were exported to SU where Kirchhoff migration was performed to produce depth slices. The migrated slices show the cavities as a curved bright spot. This is because no diffractions were used in the model. Real data will have diffractions and the image could actually be better. The original sonar images resulting from the simulated slices are not shown here.

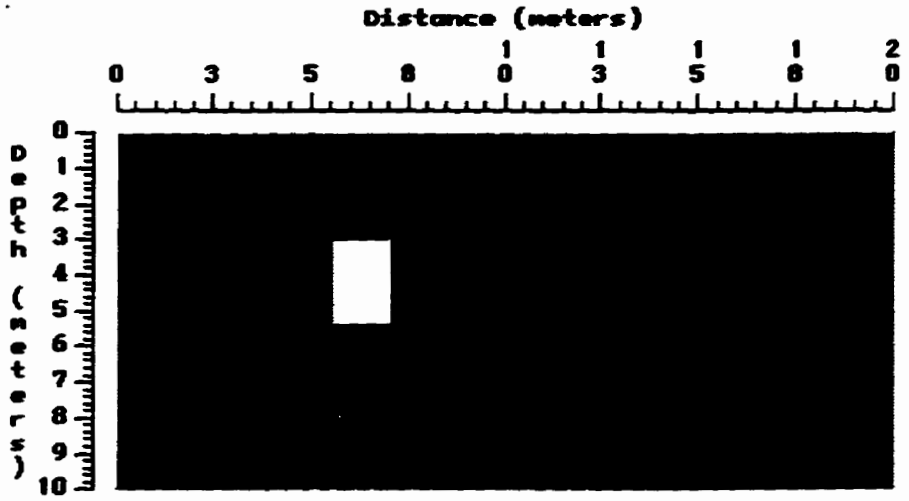

Figure 6.12: Slice one. A slice through the entrance of the tomb at depth of 3 meters.

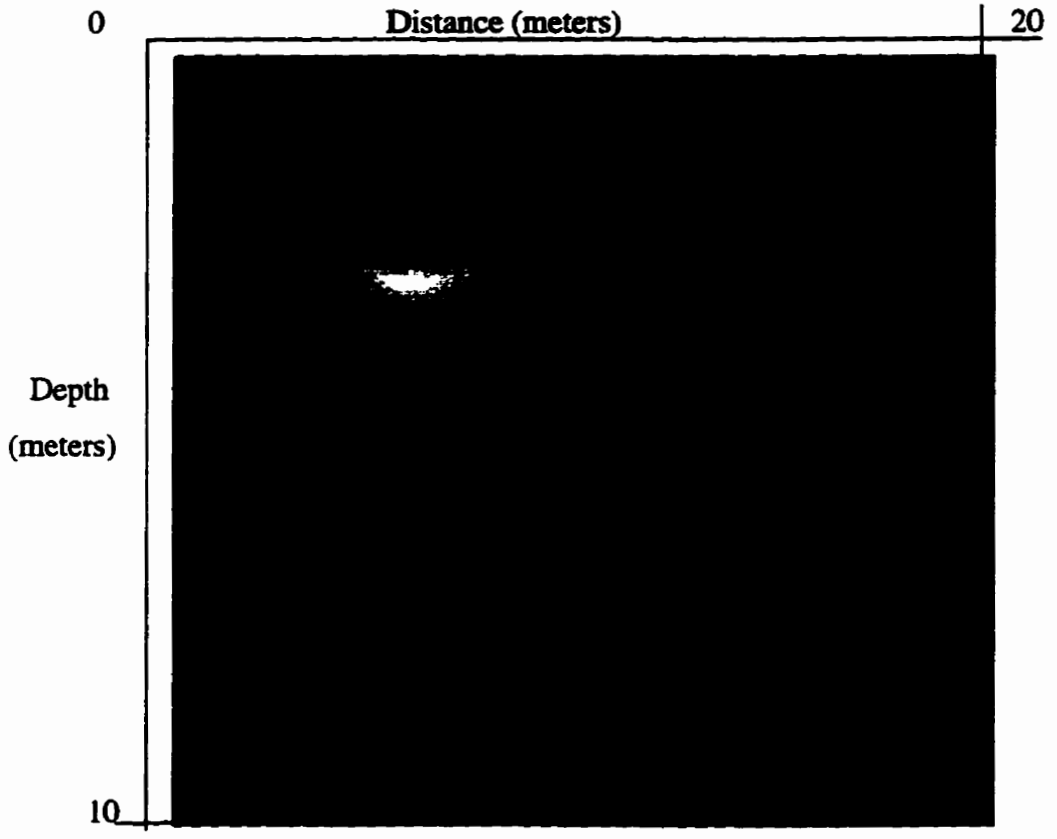

Figure 6.13: The bright spot represents the upper surface of the cavity in slice number 1.

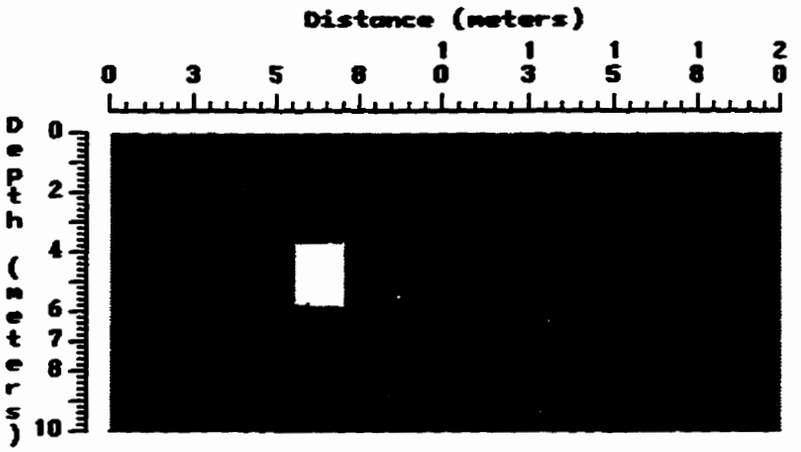

Figure 6.14: Slice 2. Still at the entrance of the tomb but now at a depth of 3.75 meters.

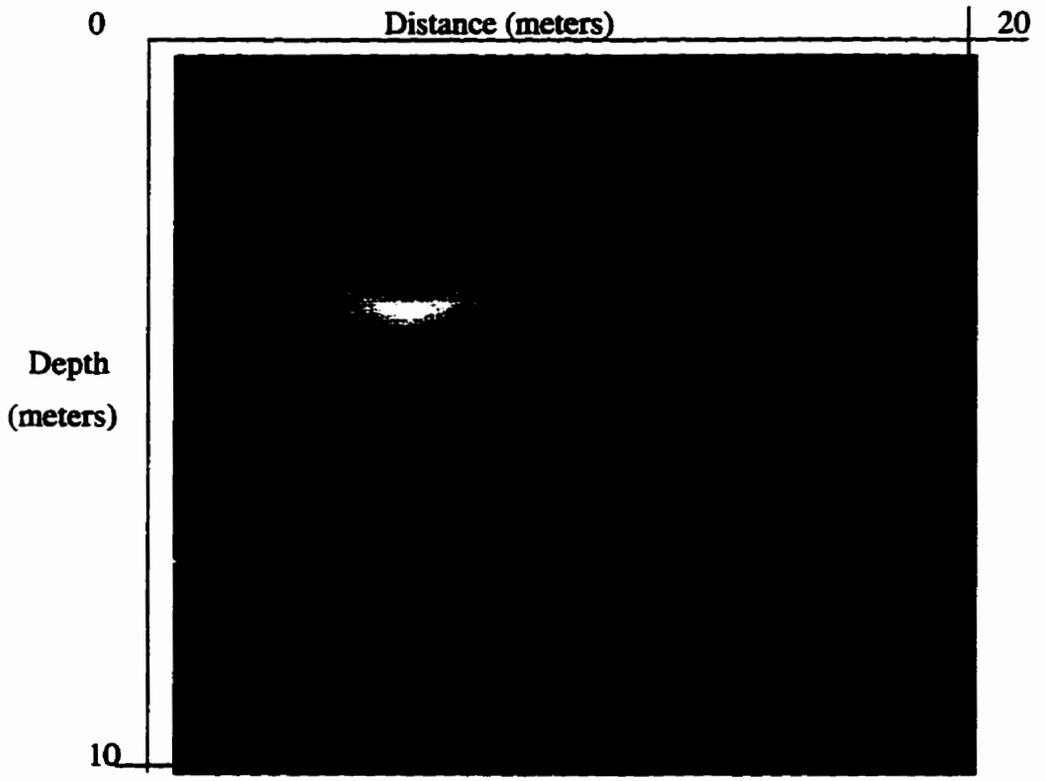

Figure 6.15: Again the bright spot represent the upper surface of the tomb entrance.

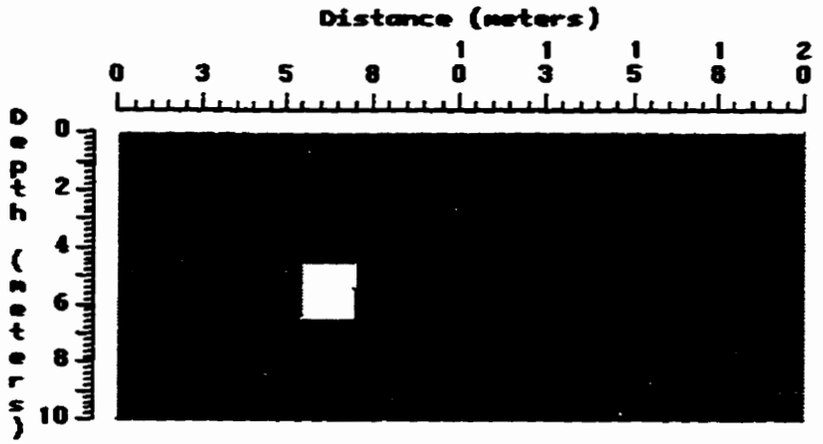

Figure 6.16: Slice 3. The entrance is at a depth of 4.5 meters.

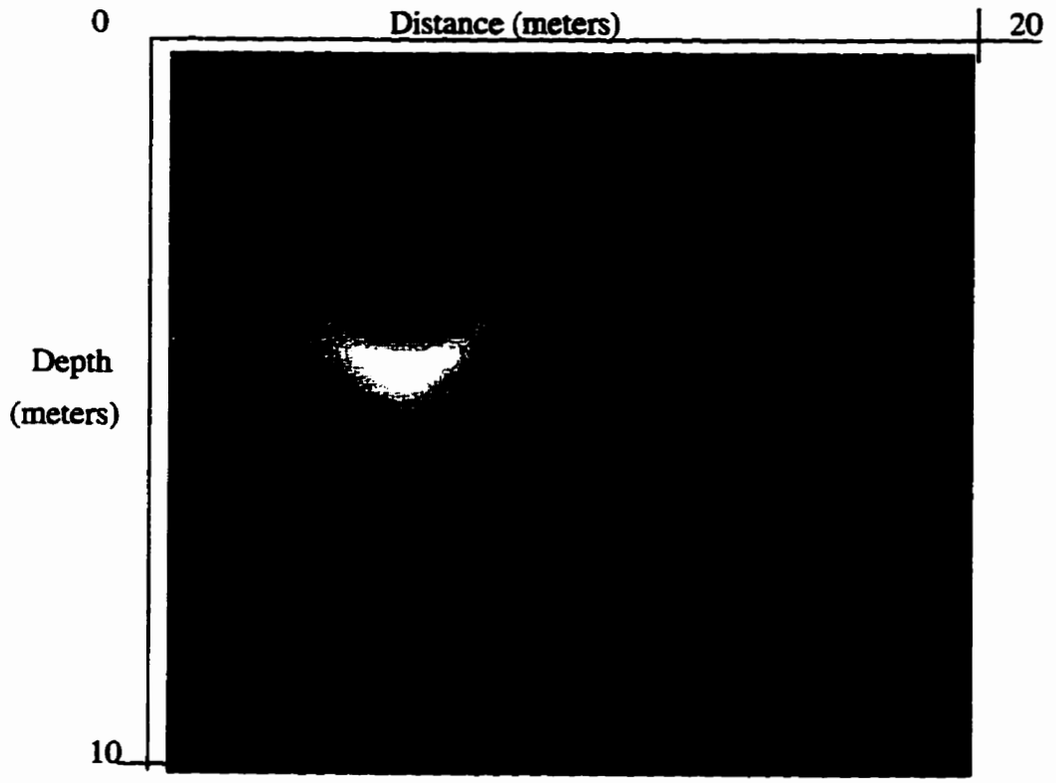

Figure 6.17: The corresponding migrated slice for slice 3.

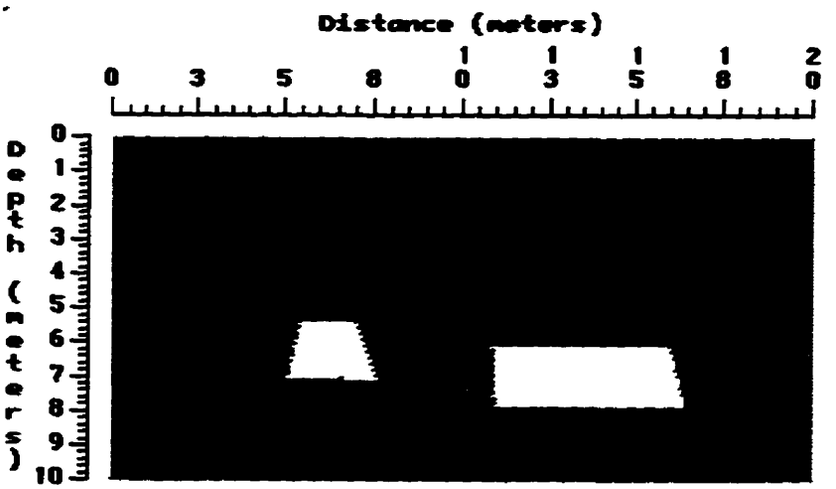

Figure 6.18: Slice 4. The entrance to the left at depth 5.25 meters with the beginning of the treasury at a depth of 6 meters.

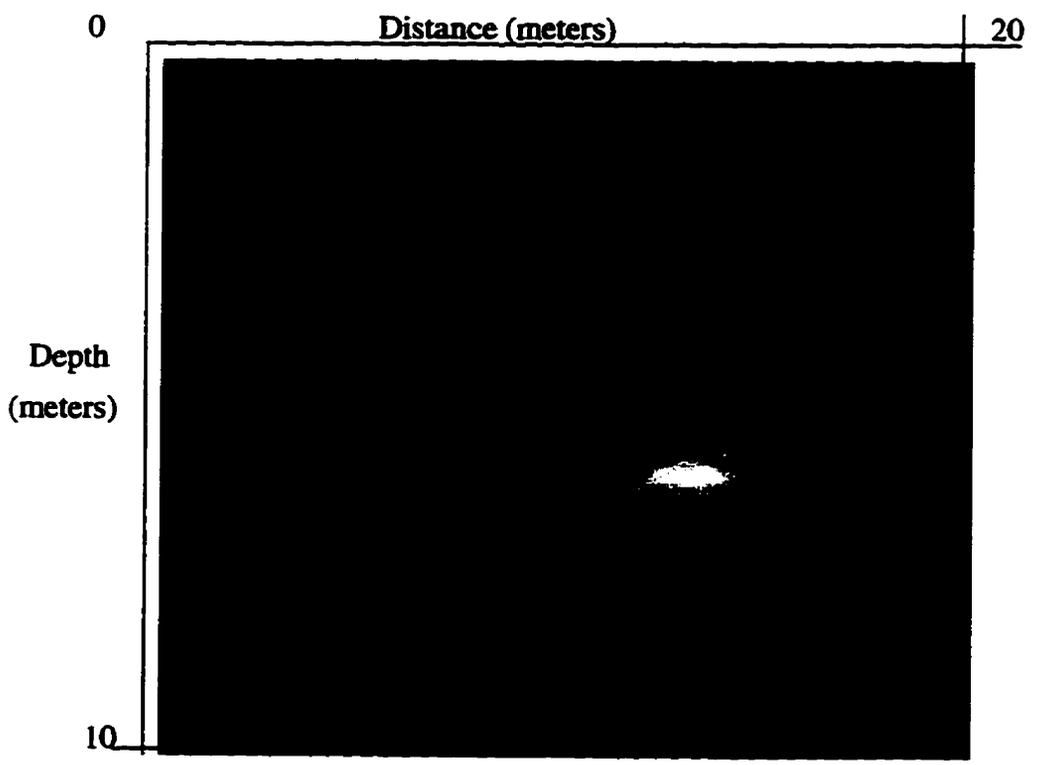

Figure 6.19: Migrated slice 4. Notice how the upper surface of the treasury is much brighter than the smaller cavity of the entrance.

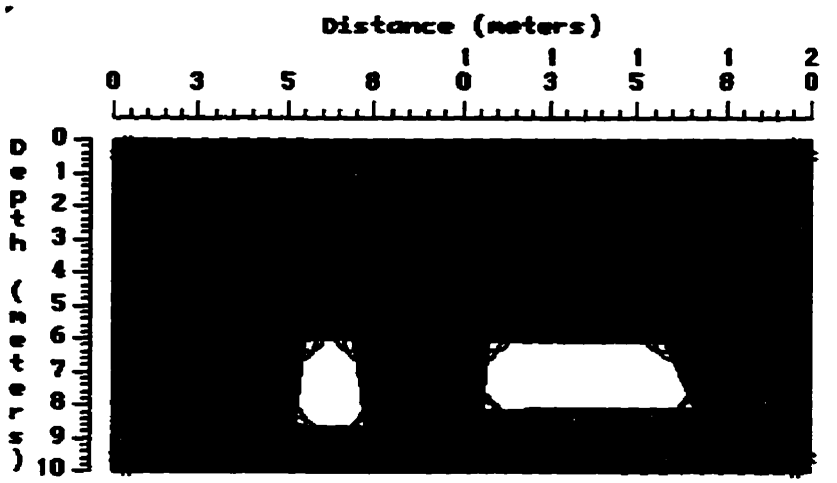

Figure 6.20: Slice 5. The last part of the entrance to the left at depth 6 meters and the treasury at the same depth to the right.

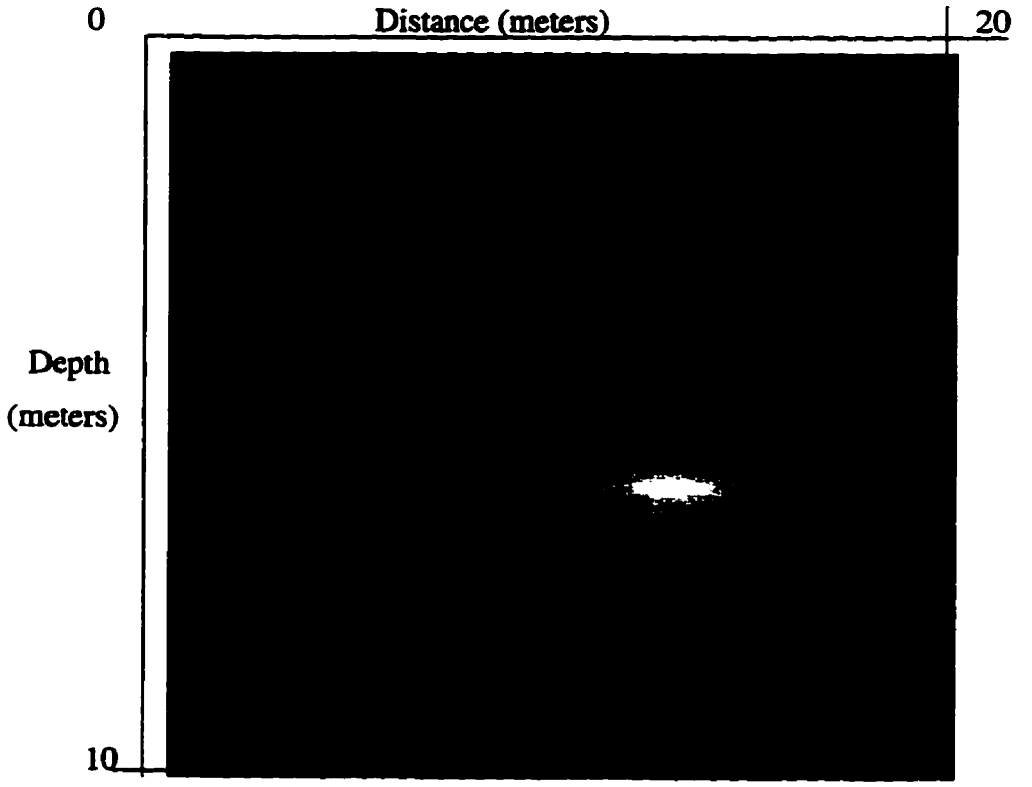

Figure 6.21: Migration of the signals produced by slice 5.

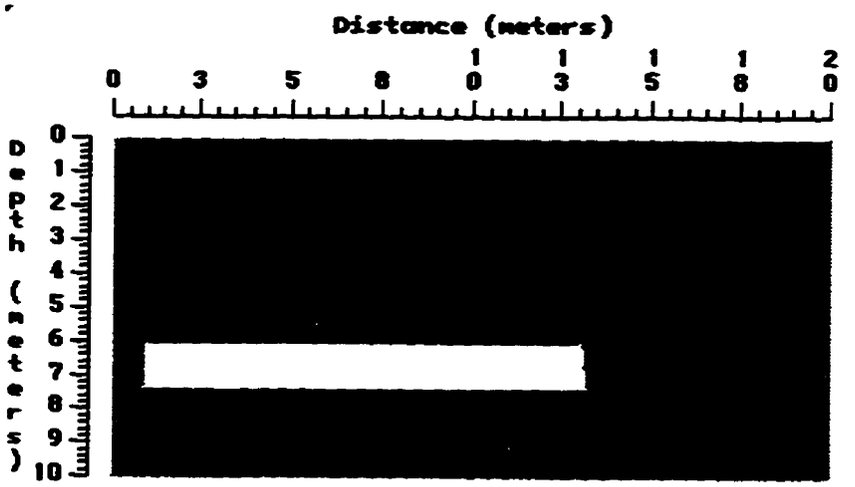

Figure 6.22: Slices 6 and 7. The sarcophagus chamber.

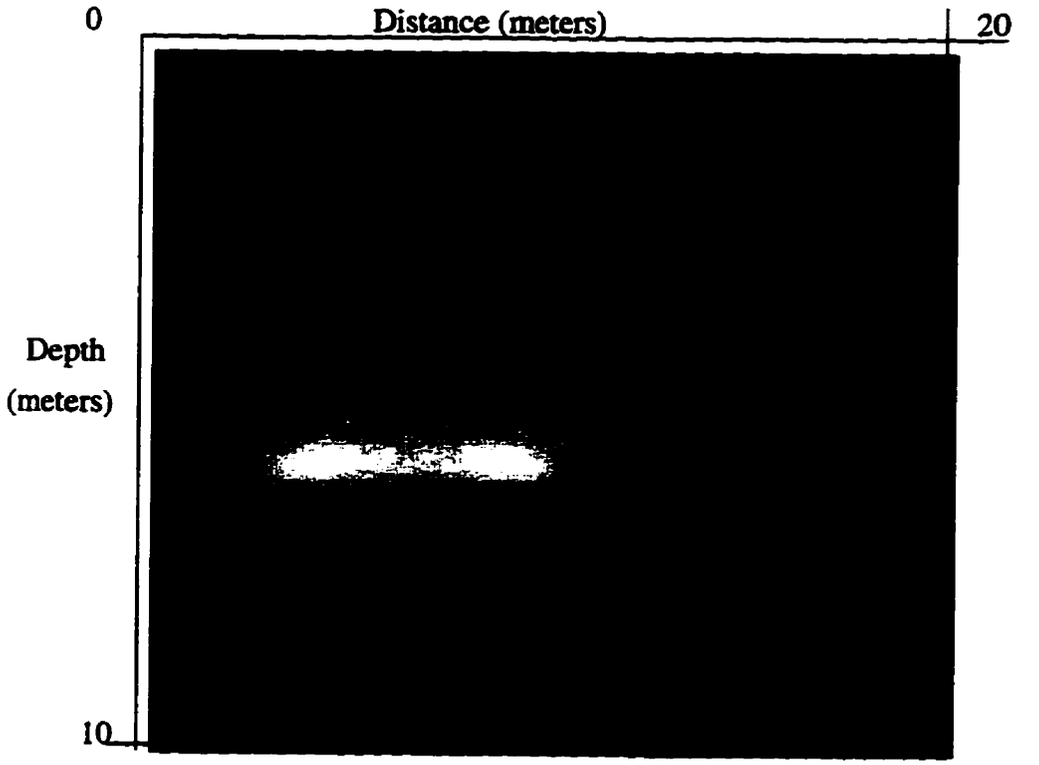

Figure 6.23: Migration of slices 6 and 7.

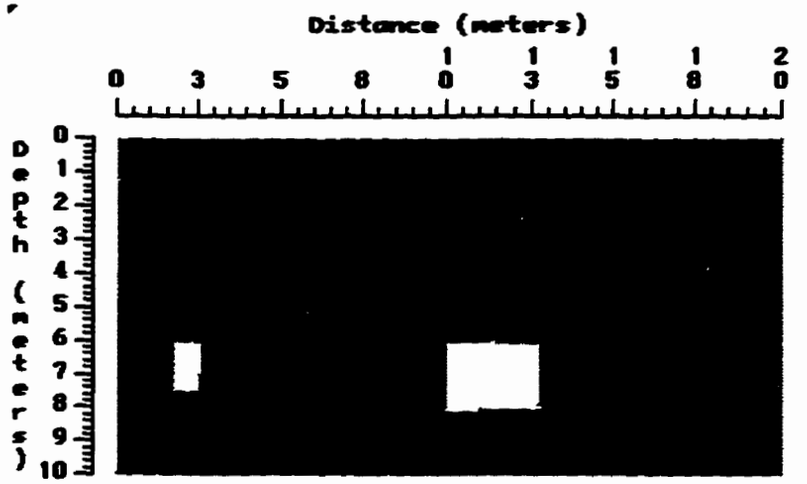

Figure 6.24: Slice 8. A junction connecting the sarcophagus chamber to the annex to the left and another compartment to the right.

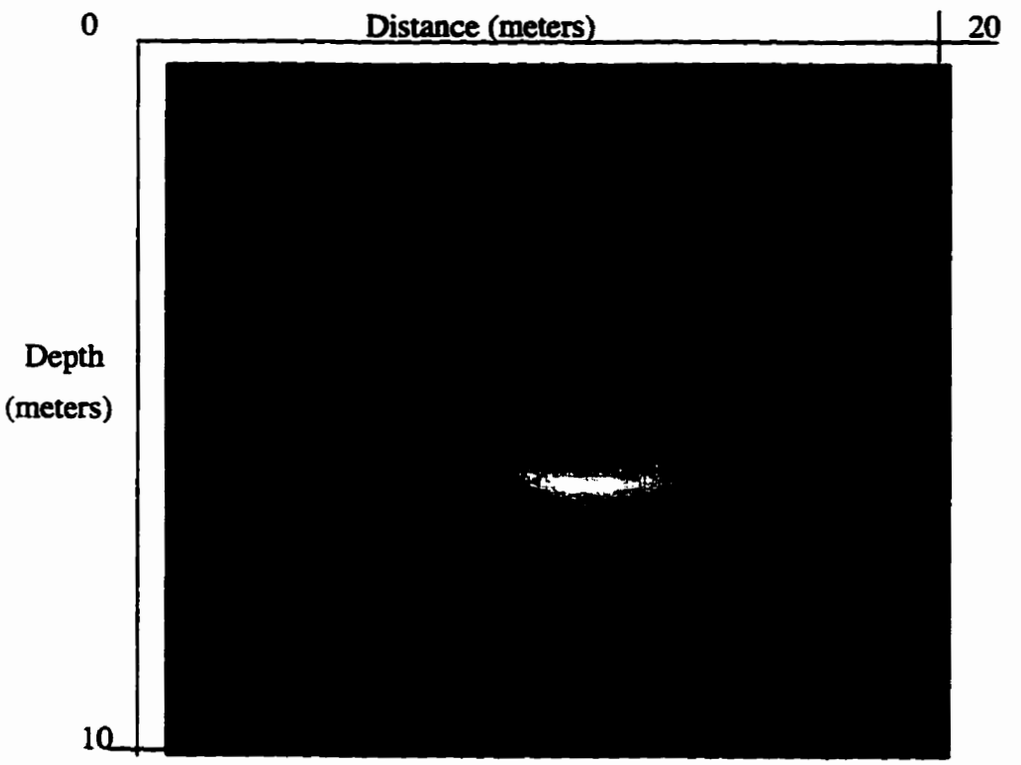

Figure 6.25: Depth migrated image of slice number 8.

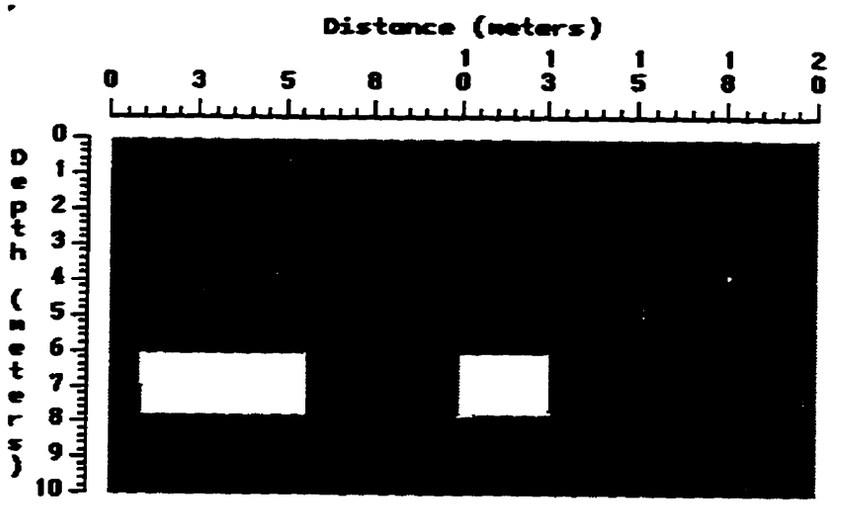

Figure 6.26: Slice 9. The annex to the left and the other compartment to the right.

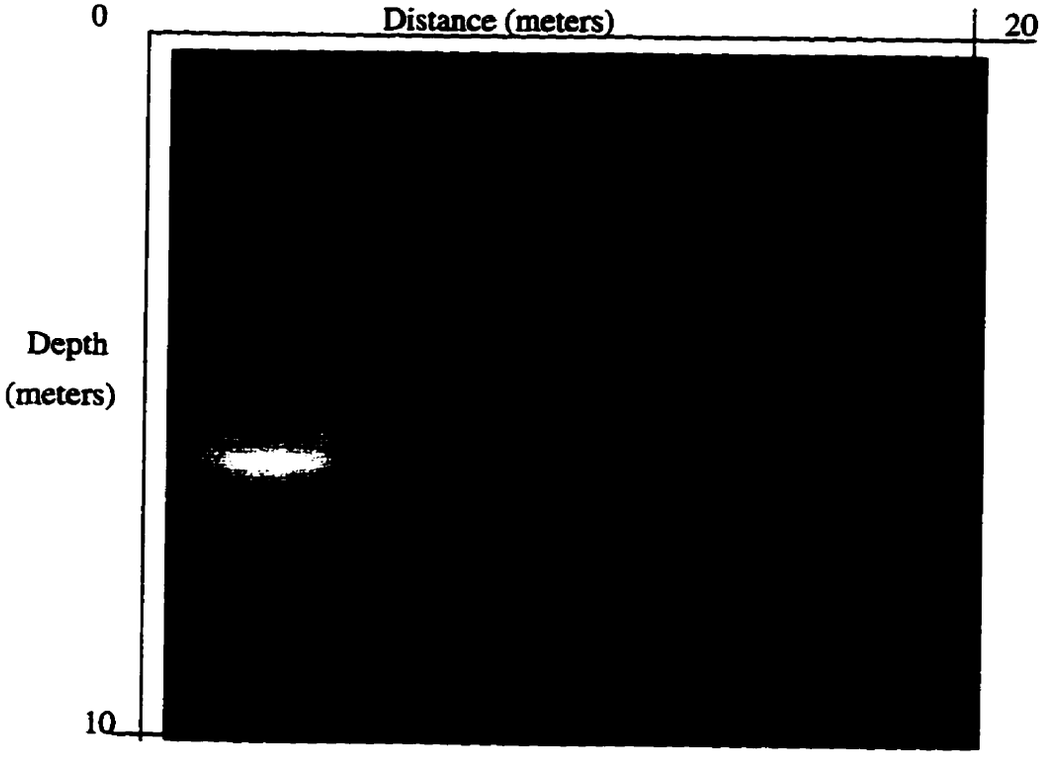

Figure 6.27: A migrated image of slice 9.

The migrated images were used in the sequence shown (with slice number 6 and 7 being used twice) to reconstruct a 3-D model for the tomb of Tut Ankh Amen (figure 6.29). The time taken to produce this model was 32 seconds on an SGI machine (app. 80% of this time was for preprocessing). The following parameter values were used:

- Thresholding algorithm: Entropy method as described in [Kapur 1985].
- Area threshold: 100 pixels (Regions in the segmentation process with an area of less than 100 pixels were discarded since they could be considered as noise).
- Interpolation method: linear.
- Features used to map contours: Center of mass only.
- Similarity threshold for mapping contours: 0.8
- Triangulation method: A variation of the Ganapthy method as described above in Section 5.3.

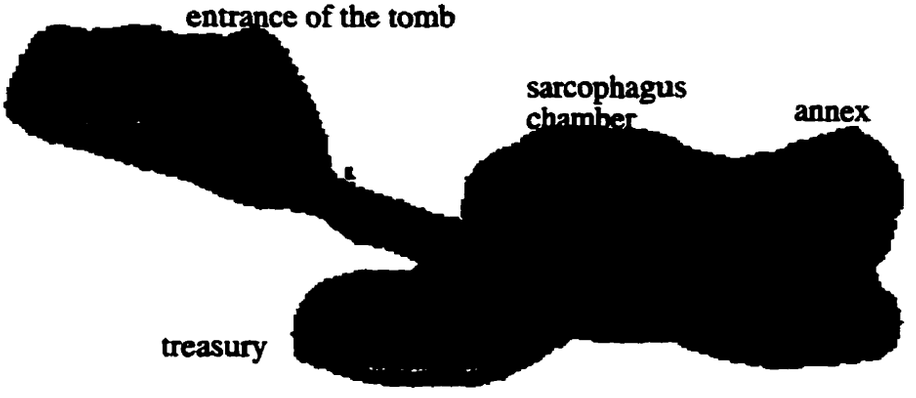

Figure 6.28: Tut's tomb produced by 3DVIEWNIX.

The same slices of Tut's tomb were imported to 3DVIEWNIX and the image shown in figure 6.28 was produced in 270 seconds on an SGI machine excluding the time needed to select a threshold interactively (compared to the performance of our system this is approximately 9 times slower). Comparing the 2 models, the visual quality is the same where all the different compartments could be easily identified.

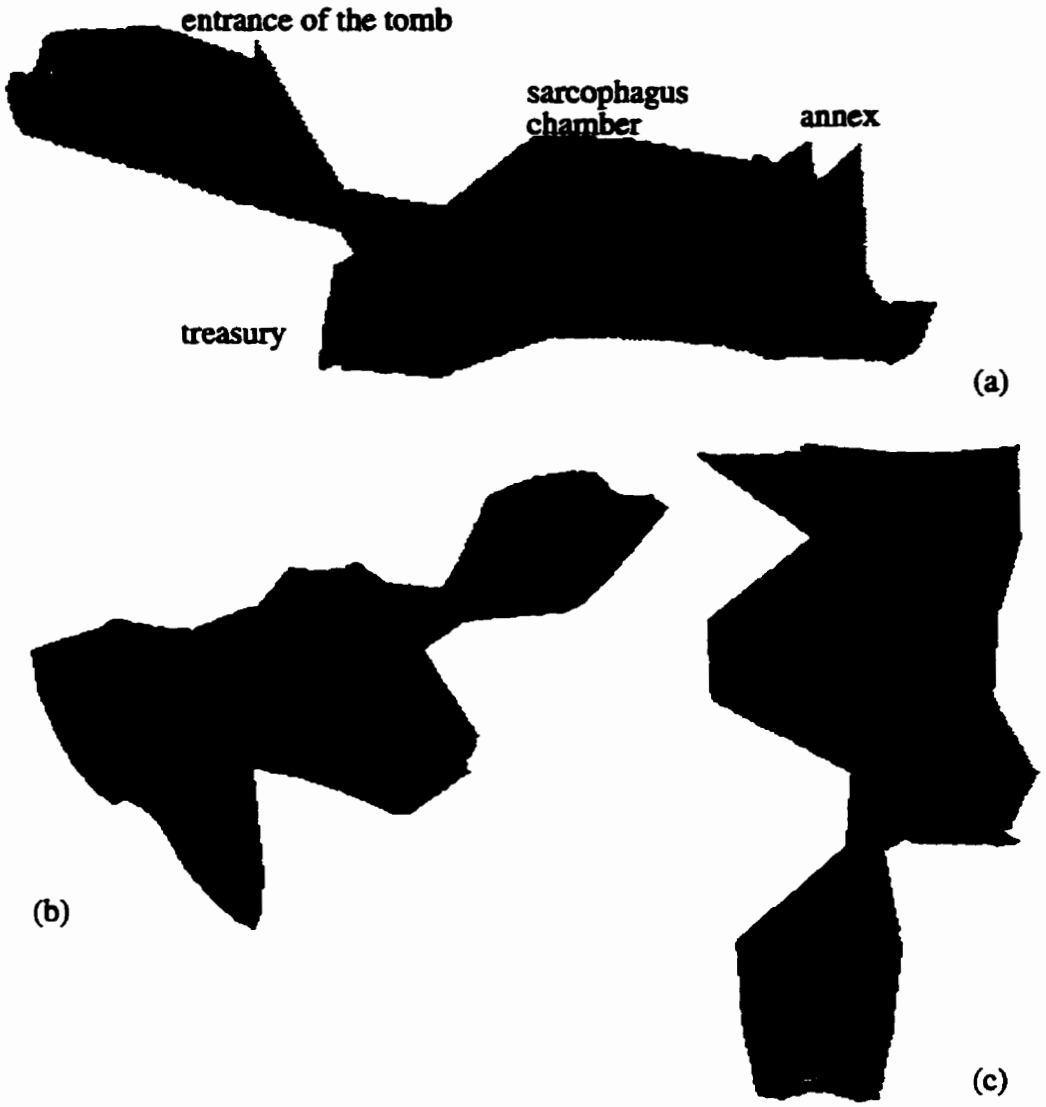

Figure 6.29: (a) A reconstruction of the tomb of Tut Ankh Amen. Each of the different compartments could be easily identified. (b) A rear view of the tomb. (c) A top view of the tomb.

Chapter 7 Conclusions

And though I have ... all knowledge ...
and have not charity, I am nothing.
-- St. Paul (1Corinthians 13:2)

A new method for reconstructing 3-D models from cross-sectional sonar images is presented in this thesis. The presented algorithms proved to be superior to other tested algorithms in branching and interpolation yielding a smooth model at the end. Also, it is superior to other available software packages in terms of being automatic (user does not have to set parameters such as for thresholding) and fast. The process involves preprocessing the images to find a series of contour points for the triangulation process and then a variation of the Ganapthy method is used to triangulate each pair of contours at a time. Contour mapping is performed to find similar contours on source and destination slices. The mapping could be 1:1 (no branching), m:1 (branching from destination to source) or 1:m (branching from source to destination). Each pair of matched contours is triangulated and the distance between each slice is taken into

consideration such that whenever this distance is greater than a certain threshold a linear interpolation process between those two contours is performed to produce a smooth object.

Major contributions and further research enhancements are discussed in the following subsections.

7.1 Primary goal and major contributions.

The primary goal of this thesis was to develop a software solution that could be used to reconstruct 3-D models from simulated 2-D seismic slices taking into consideration how realistic that object is, the speed of reconstruction and the automation of the whole process. The major contributions of this thesis are:

- The branching/interpolating algorithm using the contour mapping technique discussed above.
- The VSP program for non-parallel slices.
- The automation of the thresholding process using the entropy method discussed in [Kapur 1985].
- The modifications for the triangulating algorithm explained in [Ganapathy 1982].

A completely automatic system seemed to be very difficult to achieve. Any user who uses our system should change the set of parameters for every class of images. Thus, the system might be called a semi-automated one.

7.2 Future research avenues.

Further research could be done in testing this algorithm with real data. Some changes to

the preprocessing phase might take place in this case. For example, a smoothing of the produced sonar migrated images, as mentioned above for the simulated noisy images, might yield good results.

Recognizing the 3-D objects reconstructed might have many applications. Recognizing underground cavities or at least classifying them could be used in detecting land mines (by trying to know the kind of mine being surveyed).

Further automation of the system is possible by automating the process of choosing the interpolation threshold and the image thresholding method for a larger class of image slices.

The existence of surfaces inside the reconstructed model, resulting from the branching method described in Section 5.4, could be removed while interpolating between two slices. The union of overlapping contours, produced by the interpolation algorithm, could be triangulated with the source contour as explained in the branching algorithm of [Ekoule 1991]. This will help removing the inside surfaces if they are not required in any application.

7.3 Final word.

Research will never end in the area of 3-D reconstruction from cross-sectional slices. More and more applications will be handled and improvements will take place all the time since no one single solution will be globally approved upon. The only criterion for the success of a certain 3-D reconstruction solution is how far this solution contributed to the understanding of its users in terms of making them perceive what they can't see in a realistic way.

Bibliography

[Albaho 1995] Albaho, Tareq I.J. and Thomas M. Simms. The Kings' Valley's Hidden Secrets. Research Proposal. 1995.

[Ballard 1982] Ballard, Dana H. and Christopher M. Brown. *Computer Vision.* Prentice Hall, Englewood Cliffs, New Jersey, 1982.

[Boissonnat 1988] Boissonnat, J-D. Surface Reconstruction From Planar Cross-Sections. In *Computer Vision, Graphics and Image Processing*. Pages 1-29, Vol. 44, 1988.

[Boissonnat 1988b] Boissonnat, J-D. Surface Reconstruction From Planar Cross-Sections. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. Pages 393-397. 1988.

[Burr 1981] Burr, D.J. Elastic Matching of Line Drawings. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Pages 708-713, Vol. 3, No. 6, 1981.

[Christiansen 1978] Christiansen, H. N. and T. W. Sederberg. Conversion of Complex Contour Line Definitions into Polygonal Element Mosaics. In *Proceedings of SIGGRAPH*. Pages 187-192. 1978.

[Devaney 1984] Devaney, A.J. Geophysical Diffraction Tomography. In *IEEE Transac-*

tions on Geoscience and Remote sensing. Pages 3-13 , Vol. 22, 1984.

[Dohr 1981] Dohr, Gerhard. *Applied Geophysics.* Halsted Press, New York, 1981.

[Ekoule 1991] Ekoule, A.B., F.C.Peyrin and C.L.Odet. A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours. In *ACM Transactions on Graphics.* Pages 182-199, Vol. 10, No. 2, April 1991.

[Faugeras 1993] Faugeras, O.D. *Three Dimensional Computer Vision.* MIT Press, Cambridge, MA. 1993.

[Freeman 1961] Freeman, H. and R. Shapira. On the Encoding of Arbitrary Geometric Configurations. In *IEEE Transactions on Electronic Computers.* Pages 260-268, Vol. EC-10, 1961.

[Ganapathy 1982] Ganapathy, S and T. G. Dennehy. A New General Triangulation Method for Planar Contours. In *Computer Graphics.* Pages 69-75, Vol. 16, No. 3, 1982.

[Gonzalez 1992] Gonzalez, R. C. and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company. New York. 1992.

[Haralick 1983] Haralick, R.M. Image Texture Survey. In O.D. Faugeras, editor, *Fundamentals in Computer Vision.* Pages 145-172. Cambridge University Press, 1983.

[Huang 1995] Huang, L-K and M-J. J. Wang. Image Thresholding By Minimizing The Measures of Fuzziness. In *Pattern Recognition Letters.* Pages 41-51, Vol. 28, No. 1, 1995.

[Johannsen 1982] Johannsen, G. and Bille, J., A Threshold Selection Method Using Information Measures. In *Proc. of the 6th International Conf. on Pattern Recognition,* IEEE CS Press, Los Alamitos, CA. Pages 140-143, 1982.

[Kapur 1985] Kapur, J.N., Sahoo, P.K. and Wong, A.K.C., A New Method for Grey Level Picture Thresholding Using The Entropy of The Histogram. In *Computer Vision, Graphics and Image Processing.* Pages 273-285, Vol. 29, 1985.

[Kehtarnavaz 1988] Kehtaranvaz, N., Simar, L.R. and R. J. P. De Figueiredo. A Syntactic/ Semantic Technique for Surface Reconstruction from Cross-Sectional Contours. In *Computer Vision, Graphics and Image Processing.* Pages 399-409,

Vol. 42, 1988.

[Keppel 1975] Keppel, E. Approximating Complex Surfaces by Triangulation of Contour Lines. In *IBM Journal of Research and Development*. Pages 2-11, Vol. 19, 1975.

[Levy 1995] Levy, Thomas E. From Camels to Computers. In *Biblical Archaeology Review*. July/August 1995.

[Lin 1989] Lin, Wei-Chung et al. A New Surface Interpolation Technique for Reconstructing 3-D Objects from Serial Cross-Sections. In *Computer Vision, Graphics and Image Processing*. Pages 124-143, Vol. 48, 1989.

[Mao 1987] Mao, X. , T.L. Kunii, I. Fujichiro, and T. Noma. Reconstruction and Cross-Section Generation of Hierarchically Represented 3D Gray-Scale Digital Images. In T.L.Kunii, editor, *Computer Graphics*. Pages 461-479. Springer-Verlag, 1987.

[Meyers 1992] Meyers, D., S. Skinner and K. Sloan. Surfaces from Contours. In *ACM Transactions on Graphics*. Pages 228-258, Vol. 11, No. 3, 1992.

[Otsu 1979] Otsu, N. A Threshold Selection Method From Grey-level Histograms. In *IEEE Transactions on Systems, Man and Cybernetics*. Pages 377-393, Vol. 9 No. 1, 1979.

[Parker 1996] Parker, J.R. *Algorithms for Image Processing and Computer Vision*. John Wiley & Sons, New York, 1996.

[Parker 1996b] Parker, J.R. and Emad N. Attia. Hidden Object Reconstruction From Acoustic Slices. The University of Calgary. Research Report No. 96/590/10, October 1996.

[Parker 1994] Parker, J.R. *Practical Computer Vision Using C*. John Wiley & Sons, New York, 1994.

[Parker 1991] Parker, J.R. Gray Level Thresholding in Badly Illuminated Images. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*. Pages 813-819, Vol. 13, Aug. 1991.

[Pavlidis 1982] Pavlidis, T. *Algorithms for Graphics and Image Processing*, Computer Science, Rockville, MD, 1982.

[Pratt 1991] Pratt, William K. *Digital Image Processing.* John Wiley & Sons, New York, 1991.

[Pun 1981] Pun, T. Entropic Thresholding: A New Approach. In *Computer Vision, Graphics and Image Processing.* Pages 210-239, Vol. 16, No. 3, 1981.

[Ridler 1978] Ridler, T.W. and S. Calvard. Picture Thresholding Using an Iterative Selective Method. In *IEEE Transactions on Systems, Man and Cybernetics.* Pages 630-632, Vol. SMC-8, No. 8, 1978.

[Robb 1989] Robb, R. A. and C. Barillot. Interactive Display and Analysis of 3-D Medical Images. In *IEEE Transactions on Medical Imaging.* Pages 217-226, Vol. 8, No. 3, 1989.

[Roch 1988] Roch, M., J.Weber and C.Pellegrini. 3D Images of Molecular Properties by Traingulation of Contour Lines. In *Computer Graphics Forum.* Pages 195-201, Vol. 7, 1988.

[Sahoo 1988] Sahoo, P.K., S. Soltani and A.K.C. Wong. A Survey of Thresholding Techniques. In *Computer Vision, Graphics and Image Processing.* Pages 233-260, Vol. 41, 1988.

[Sirjani 1988] Sirjani, A. and George R. Cross. An Algorithm for Polygonal Approximation of a Digital Object. In *Pattern Recognition Letters.* Pages 299-303, Vol. 7, 1988.

[Srihari 1981] Srihari, S.N. Representation of Three-Dimensional Digital Images. In *Computing Surveys.* Pages 399-424, Vol. 13, No. 4, 1981.

[Thrussel 1979] Thrussel, H.J. Comments on "Picture Thresholding Using an Iterative Selection Method.". In *IEEE Transactions on Systems, Man, and Cybernetics.* Page 311, Vol. SMC-9, No. 5, May, 1979.

[Ulupinar 1995] Ulupinar, F. and R. Nevatia. Shape from Contour: Straight Homogeneous Generalized Cylinders and Constant Cross Section Generalized Cylinders. In *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Pages 120-135, Vol. 17, No. 2, 1995.

[Witten 1995] Witten, Alan J. , Thomas E. Levy, James Ursic and Paul White. Geophysical Diffraction Tomography: New Views on the Shiqmim Prehistoric Subter-

ranean Village Site (Israel). In *Geoarchaeology*. Pages 97-118, Vol. 10, No. 2, 1995.

[Weszka 1978] Weszka, J.S., A Survey of Threshold Selection Techniques. In *Computer Graphics and Image Processing*. Pages 259-265, Vol. 7, 1978.

[Woodward 1987] Woodward, Charles D. Cross-Sectional Design of B-Spline Surfaces. In *Computer and Graphics*. Pages 193-201, Vol. 11, No.2, !987.

[Yager 1979] Yager, R.R. , On The Measures of Fuzziness and Negation. Part 1: Membership in the Unit Interval. In *International Journal of Gen. Sys.* Pages 221-229, Vol. 5, 1979.

[Yilmaz 1987] Yilmaz, Ozdogan. *Seismic Data Processing*. Society of Exploration Geophysicists, Tulsa, 1987.