1977-12

Computer Graphics: Conversion of Contour Line Definitions Into Polygonal Element Mosaics

Thomas W. Sederberg
Brigham Young University - Provo

COMPUTER GRAPHICS: CONVERSION OF CONTOUR LINE DEFINITIONS INTO POLYGONAL ELEMENT MOSAICS

A Thesis

Presented to the

Department of Civil Engineering

Brigham Young University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Thomas W. Sederberg

December 1977

ACKNOWLEDGMENTS

The author sincerely thanks the entire BYU Civil Engineering faculty and especially his mentor, Dr. Henry Christiansen, for continual encouragement, warm friendship, and professional guidance; his family for support and encouragement throughout his educational career; Brigham Young University for generous financing of computer costs; Comtal Corporation for hardware support; Mike Milochik for photo processing; and the Office of Naval Research for funding this research effort.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

gratefully dedicated

to Hank

Chapter 1

INTRODUCTION

There has been a disparity between the conventional method of describing topographic surfaces (i.e. contour line definition) and a format of surface description often used in continuous-line computer graphics (i.e. panel definition). The two differ enough that conversion from contours to panels is not a trivial problem. A computer program that performs such a conversion would greatly facilitate continous tone display of topographical surfaces, or any other surface which is defined by contour lines.

This problem has been addressed by Keppel[1] and alluded to by Fuchs[2]. Keppel's is an highly systematic approach in which he uses graph theory to find the panel arrangement which maximizes the volume enclosed by concave surfaces. Fuchs mentions an approach to the problem as part of an algorithm to reconstruct a surface from data retrieved from a laser scan sensor.

This thesis elaborates on a general conversion system. Following a brief overview of computer graphics, a simple algoritm

---

[1] E. Keppel, "Approximating Complex Surfaces by Triangulation of Contour Lines," Journal of Research and Development, IBM Vol. 19, No. 1 (January 1975), 2-11.

[2] Henry Fuchs, "The Automatic Sensing of 3-Dimensional Surface Points from Visual Scenes" (unpublished PhD dissertation, University of Utah, 1975.)

is described which extracts a panel definition from a pair of adjacent contour loops subject to the restriction that the two loops are similarly sized and shaped, and are mutually centered. Next, a mapping procedure is described which greatly relaxes the above restrictions. It is also shown that the conversion from contours to panels is inherently ambiguous (to various degrees) and that occasionally the amibiguity is great enough to require user interaction to guide the conversion algorithm. An important complication addressed in this thesis is the problem of handling cases where one contour loop branches into two or more (or vice versa).

Attention turns next to a contour line definition of the human brain, and special problems encountered in preparing those data for continuous tone display. The final chapters explain the fortran implementation, present an example problem, and show sample pictures of the brain parts.

Chapter 2

AN OVERVIEW OF COMPUTER GRAPHICS

The past decade has seen fantastic advances in the field of computer graphics. Today, it is a sheltered person who is not familiar with some form of computer graphics, be it Snoopy calendars or computer ping-pong on one end of the spectrum, or sophisticated airline pilot training simulators on the other end. Display mediums used in graphics are very diverse, and include raster scan cathode ray tubes, cathode ray storage tubes, conventional line printers, plotting machines, and film recorders. Perhaps the most life-like pictures are continuous tone images produced on raster scan cathode ray tubes.

Continuous tone display requires the capability of defining the light intensity of each pixel of a scan line - TV style. There are typically 512 scan lines per picture with 512 pixels per line, and 256 levels of light intensity for each pixel. For a color image, each pixel must know the light intensity for each of the three primary colors. Given the intensity information, a picture can be 'painted' pixel by pixel, scan line by scan line.

Whereas the display itself is strictly a hardware problem, the software problem is chiefly this: What intensity should each of the 250,000 odd pixels have in order to create the desired picture? The preceding question assumes a microscopic perspective, whereas the actual software development proceeds at a macroscopic level. The overall software problem divides itself into several major sub-problems, such as spatial orientation (translation and rotation of the object), perspective, hidden surface removal, and reflectivity. This brief overview omits discussion of the solution to these problems, but the reader is referred to a sampling of literature addressing these problems.[1,2,3]

One point must be made here, however. Continuous tone graphics concerns itself with surfaces - specifically surfaces of mathematical models. Consequently, only surface definitions, as opposed to line or point definitions, can be used as input data. One way to define an arbitrary surface is to approximate it as a network of discrete polygonal elements (triangles and quadrilaterals) which are defined first by vertices in 3-D space, and further by a connecting perimeter. Such a definition will herafter be referred as a panel definition.

The continuous tone pictures in this thesis were photographed off a Comtal Image Generator. The display files were generated using MOVIE.BYU - a powerful graphics package written by Dr. Christiansen

---

[1]Henry N. Christiansen, "Applications of Continuous Tone Computer-Generated Images in Structural Mechanics," Structural Mechanics Computer Programs - Surveys, Assessments, and Availability, University Press of Virginia, Charlottesville, Virginia, June 1974, pp. 1003-1015.

[2]Henry N. Christiansen, "MOVIE.BYU - A General Purpose Computer Graphics Display System," Proceedings of the Symposium on Applications of Computer Methods in Engineering, University of Southern California, Los Angeles, August 1977.

[3]William M. Newman and Robert F. Sproull, Principles of Interactive Computer Graphics (New York: McGraw-Hill, 1973)

(of Brigham Young University) and Dr. Stephenson (now at the University of Arizona). This thesis focuses on generating panel definitions from contour data in a format compatible with the requirements of MOVIE.BYU.

Chapter 3

CONVERTING CONTOURS INTO PANELS

A LIMITED TRIANGULATION ALGORITHM

A contour line can be viewed mathematically as the intersection of an arbitrary surface and a plane. In topography, the plane is generally horizontal at a specified elevation. If the surface is closed, its contour lines will likewise be closed loops. A set of contour lines on evenly spaced parallel planes comprise a contour definition of a surface.

Contour lines of an irregular surface, such as found in nature, do not lend themselves to curve fitting, or other attempts at precise mathematical description. The most convenient numerical description of a contour line is perhaps one where the line is approximated as a string of straight line segments. This digitized contour line offers two pieces of information: nodal coordinates, and connectivity of nodes. Connectivity is implied by the sequence in which the nodes are listed.

Triangulation - the process whereby a panel definition of triangular panels is extracted from a contour definition - is greatly facilitated by observing the connectivity inherent in contour data. That connectivity leads us to explicitly note an obvious rule in triangulation: If two nodes of the same contour are to be defined as nodes of the same triangle, they <u>must</u> neighbor each other on their contour line.

Also, no more than two vertices of any triangle may be recruited from the same contour line (except, of course, in the special case where the entire area enclosed by that contour is to be capped off).

Triangulation is most logically carried on between pairs of adjacent contour lines. Consider this pair of contour loops T (top) and B (bottom).

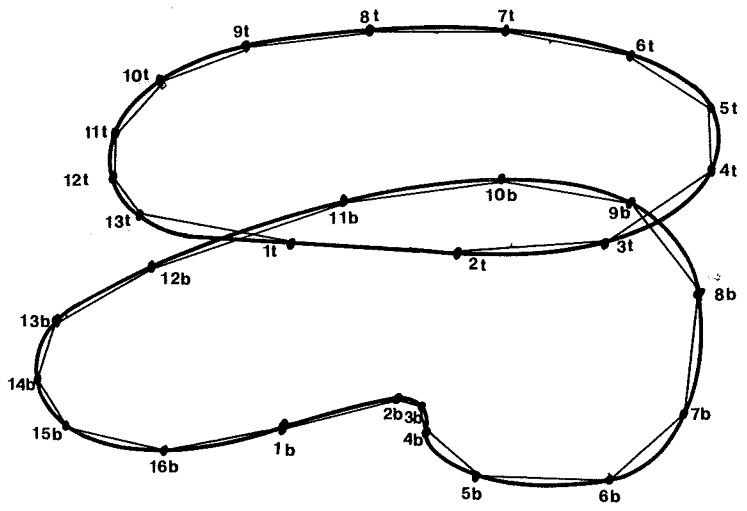


Figure 1

Contour Pair Prior To Triangulation

Two requirements must be met before triangulation commences. First, both loops must run in the same rotational direction, and second, the first nodes of each loop must be proximate. Both rules are met by these loops, and they are ready for triangulation.

Perhaps at this point discussion might best center on the finished product.

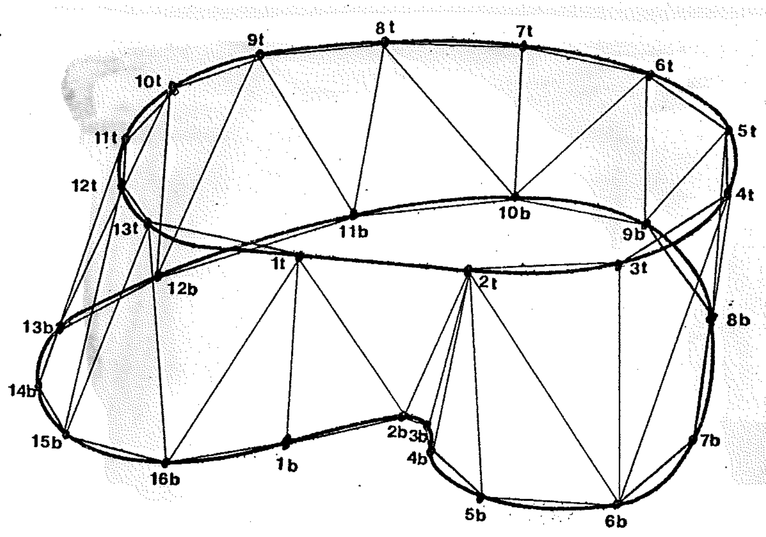


Figure 2

Triangulated Contour Pair

Observe in figure 2 the triangulated contour pair. If one were to ask oneself "How could a computer algorithm be taught to do this?", a few ideas would assert themselves. First, each contour segment can be considered to be the base of a triangle, with the third vertex being a node from the other contour. Secondly, each triangle appears to be as fat as possible. That is, the third vertex is always very near its counterparts on the other contour line.

With these ideas in mind, consider again the untriangulated loops. Referring to figure 3, triangulation commences by defining diagonal 1t-1b. Since contour connectivity requires 1t-2t and 1b-2b as bases of triangles, there are exactly two candidates for the first triangle: 1t-1b-2t, and 1t-1b-2b. Glancing back at the solution,

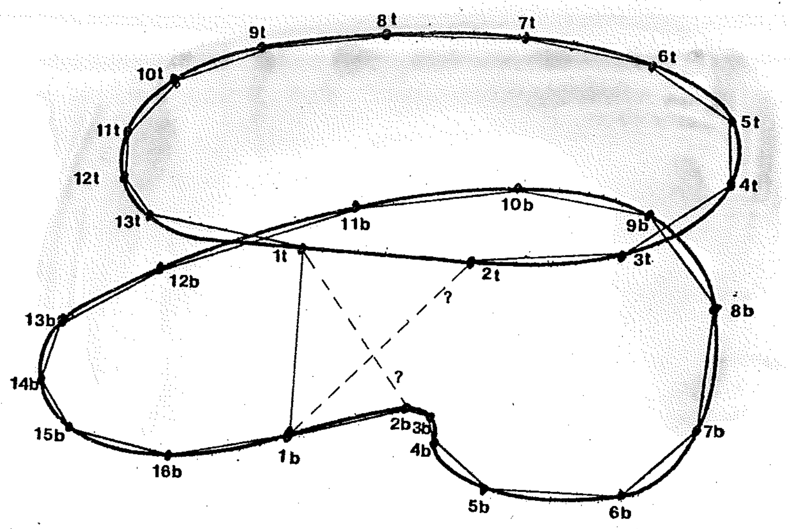


Figure 3

Commencing Triangulation

it is seen that 1t-1b-2b was selected. Moving on, once again there are exactly two possibilities for the second triangle: 1t-2b-2t, and 1t-2b-3b. This time, triangle 1t-2b-2t is selected. Notice that in each case, there are only two triangles to decide between, and that the triangle with the shortest diagonal is chosen. This procedure continues until both loops have been traversed.

This "shortest diagonal" algorithm is very easily implemented, and works fine as long as the two loops are mutually centered and are of reasonably similar size and shape.

MAPPING

The basic "shortest diagonal" algorithm fails for mildly complex cases. A typical example is found in this pair of offset contours.

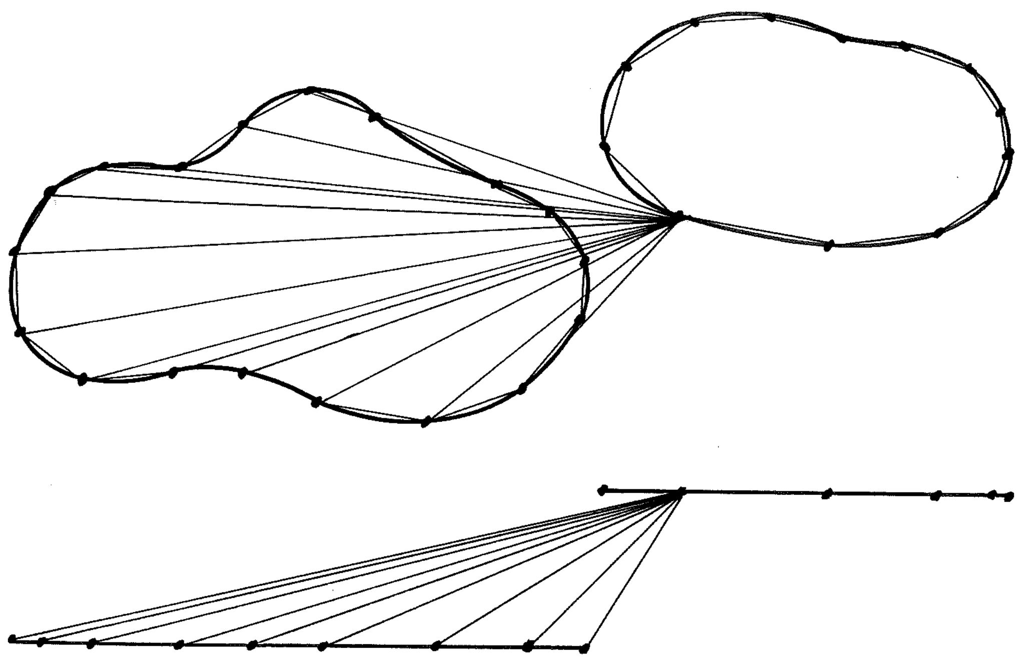


Figure 4

Failure Example

Here, the shortest diagonal search results in a cone. Rather than abandoning the algorithm, let's consider modifying the contour loops to make them more acceptable. As mentioned, the algorithm prefers contour pairs to be mutually centered, of similar size, and of similar shape. The first two requirements can be met by mapping the loops onto a unit square prior to triangulation. (Mapping also tends to make the shapes more uniform, though not always enough. This problem is addressed in the next section.)

Mapping is easily done using translation and scaling functions. Each contour is mapped consecutively in the following manner:

1. Define the rectangular window which encloses the contour.

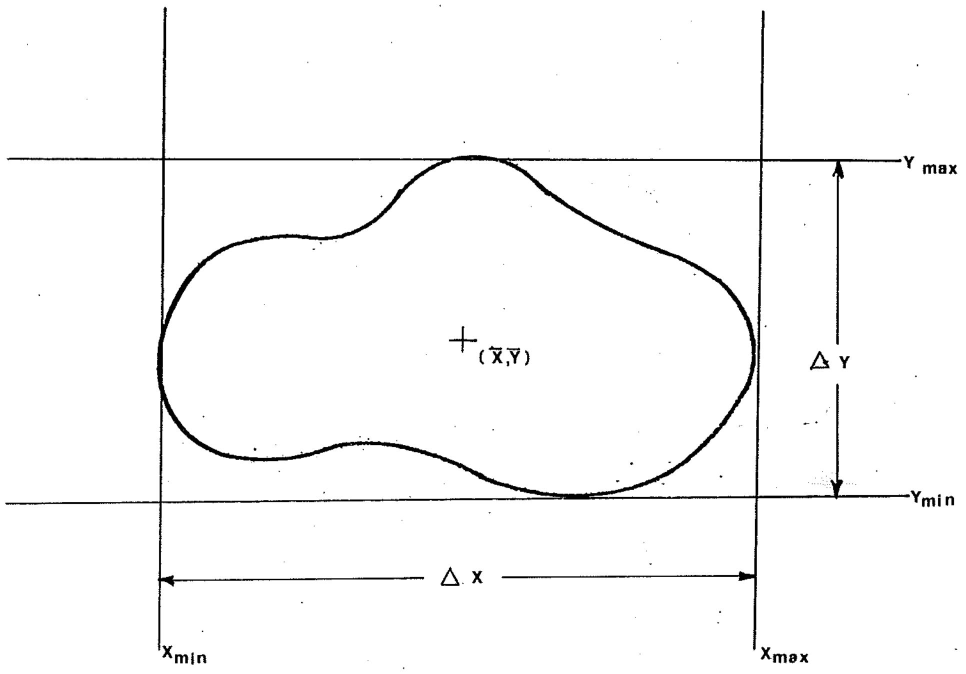


Figure 5

Window Parameters

2. Calculate $\Delta X, \Delta Y, \overline{X}$, and $\overline{Y}$.

3. Map onto a unit square centered at (0,0) by translating and scaling the contour such that its window matches the unit square's window. The equations for this are:

$$X' = (X - \overline{X})/\Delta X$$

$$Y' = (Y - \overline{Y})/\Delta Y$$

The mapped contour pair looks like this:

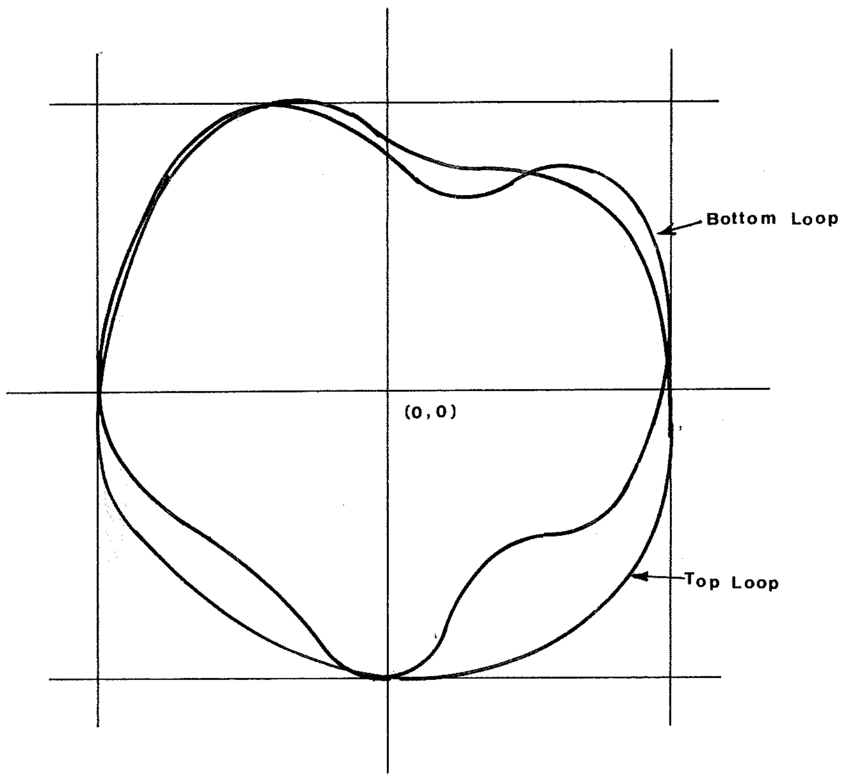


Figure 6

Mapped Contour Pair

With both contours thus mapped, they are easily handled by the original algorithm.

A fringe benefit of mapping is that the resulting triangles tend to align themselves with diagonals that are biased in the direction of the offset. This creates a desirable longitudinal texture.

ULTIMATE AMBIGUITY

A set of contour lines contains the following mathematical information:

1. Exact coordinates of some points on the surface.
2. Approximate gradients in the X-Y contour plane.
3. A general idea of the range of possible Z-gradients.

A panel definition contains items 1 and 2 and improves on item 3 by pinning down approximate Z components of surface gradients. Consequently, there is a degree of ambiguity inherent in the triangulation problem.

When two loops are similarly shaped, the ambiguity is negligible. To illustrate, consider these two solutions of the same triangulation problem;

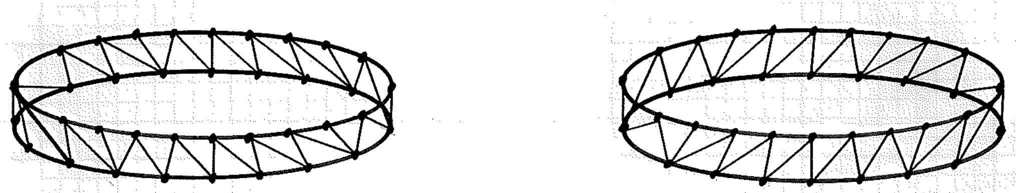


Figure 7

Synonimous Triangulation Interpretations

Since these two solutions are different, one of them is probably a more exact approximation of the actual surface. But, since the true surface gradients are not available for comparison, and since the two solutions are so similar, either solution is probably adequate. After all, contour lines form a skeletal framework that cast rather rigidly the shape of the surface.

However, as the respective shapes of a contour pair become increasingly divergent, the ambiguity becomes increasingly pronounced. The following convolution provides a good example:

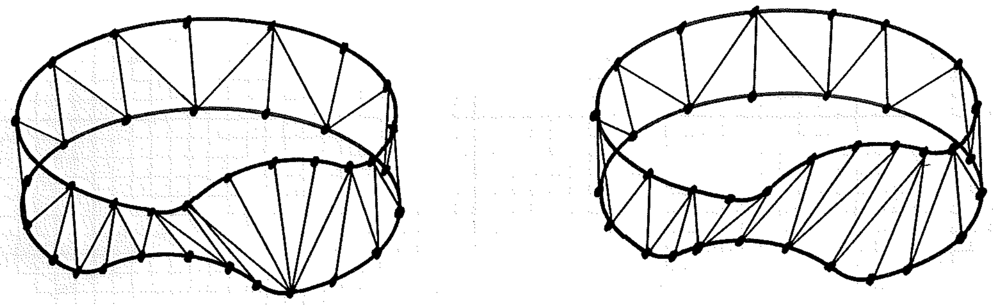


Figure 9

Non-Synonimous Triangulation Interpretations

Here, the variation in interpretation is not as tolerable. Both solutions are reasonable, yet one is wrong. Clearly, more information is required to resolve this problem.

There are two ways to provide the needed information. First, one could require the contour planes to be close enough together that there is minimal variation between adjacent contour lines. This approach has the advantage of tending towards an exact description, and the disadvantage of being uneconomical.

The second approach (adopted in this thesis by default) is to request user interaction to guide the triangulation over cases of excessive ambiguity. This is a more general solution to the problem. Here, the user is called upon to resolve the ambiguity with his knowledge of the true shape of the surface. For the mechanics of how this is implemented in the computer program, refer to the user documentation and the example problem.

BRANCHING

An important feature of this algorithm is the capability to handle branching. Consider this simple case where one contour loop branches into two:

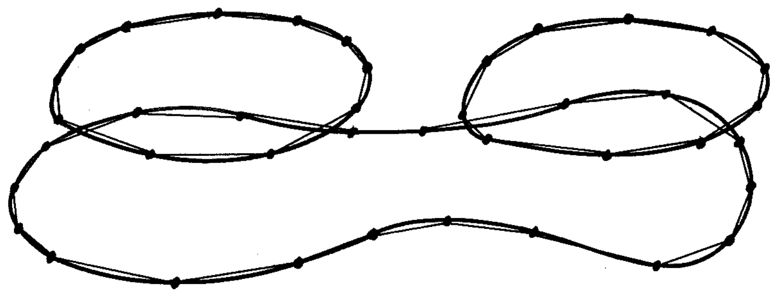


Figure 9

Simple Case of Branching

One way to handle this is to respectively treat each contour as if it were alone, neglecting the other branch. The resulting triangulation would appear like this:

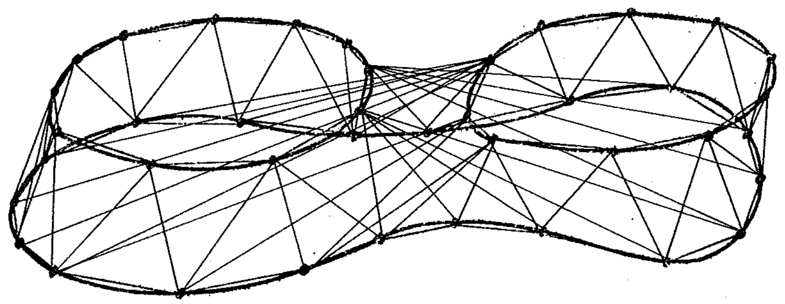


Figure 10

Uneconomical Handling of Branching

Garbled as it looks, hidden surface elimination cleans it up, and provides a smooth transition between branches.

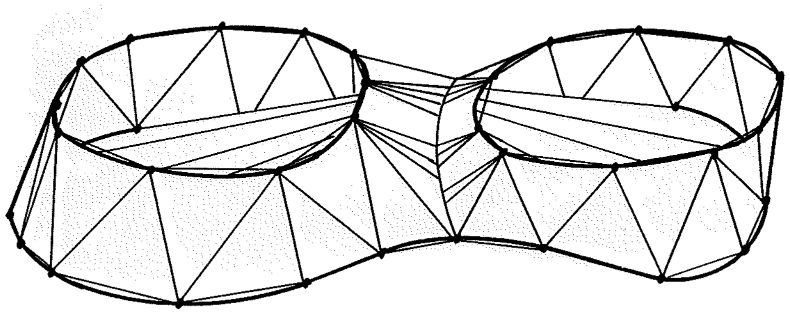


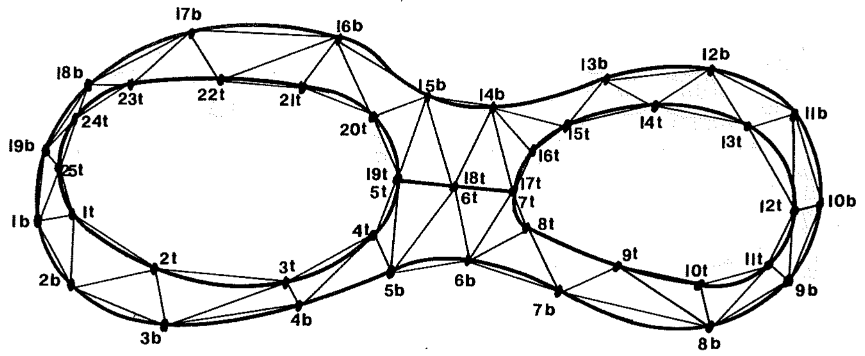
Figure 11

Preceding View With Hidden Surface Elimination

Drawbacks are that it is uneconomical, and it is unacceptable in even mildly complex branching situations.

A more economical, and more general, approach to branching is outlined in this thesis. The idea is to treat all branches as one continuous closed loop by introducing a new node midway between the closest nodes on the branches and renumbering the nodes of the branches and the new node(s) such that they can be considered as being one loop. The Z coordinate of the new node is the average of the Z coordinate of the two levels involved.

Plan



Elevation

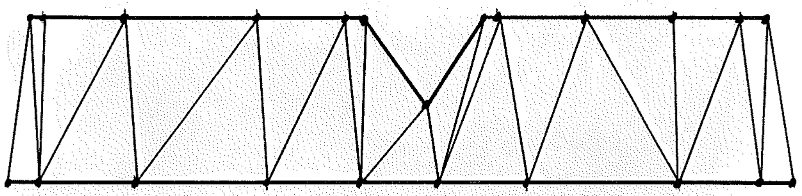


Figure 12

Preferred Handling of Branching

As seen from figure 12, the new node and its immediate neighbors are numbered twice to give the effect of one continuous loop. Triangulation can now proceed as normal. The scheme is easily expanded to handle more than one branch.

Often, there are several contour loops on adjacent planes, posing the problem of loop connectivity. Which loops should be triangulated one-on-one, and which are cases of branching? Judgment, in clear cut cases, can be made on the basis of window overlap.

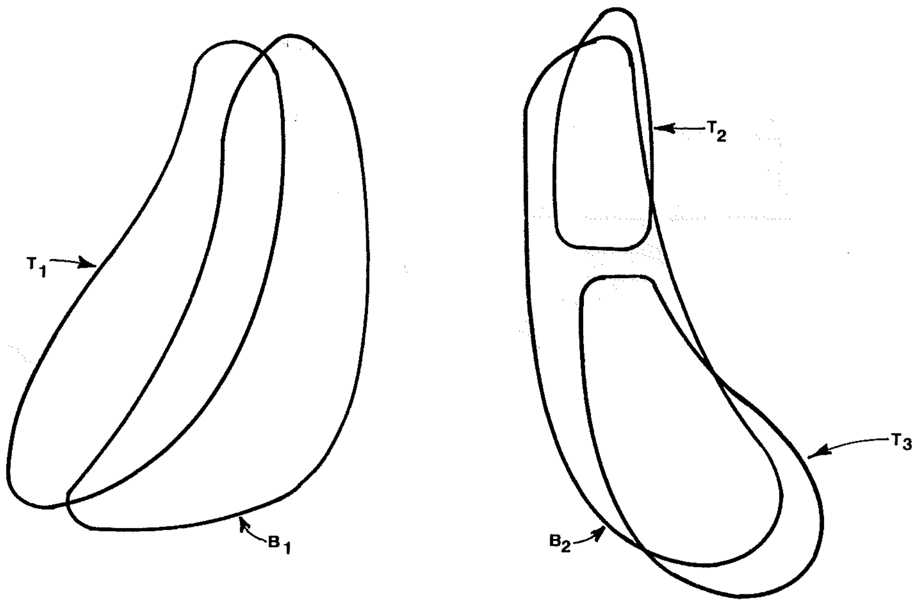


Figure 13

Typical Problem in Connectivity

Here, T_1 and B_1 clearly go together, and B_2 clearly branches into T_2 and T_3. Window overlap is best found by default: IF they don't _not_ overlap, they overlap.

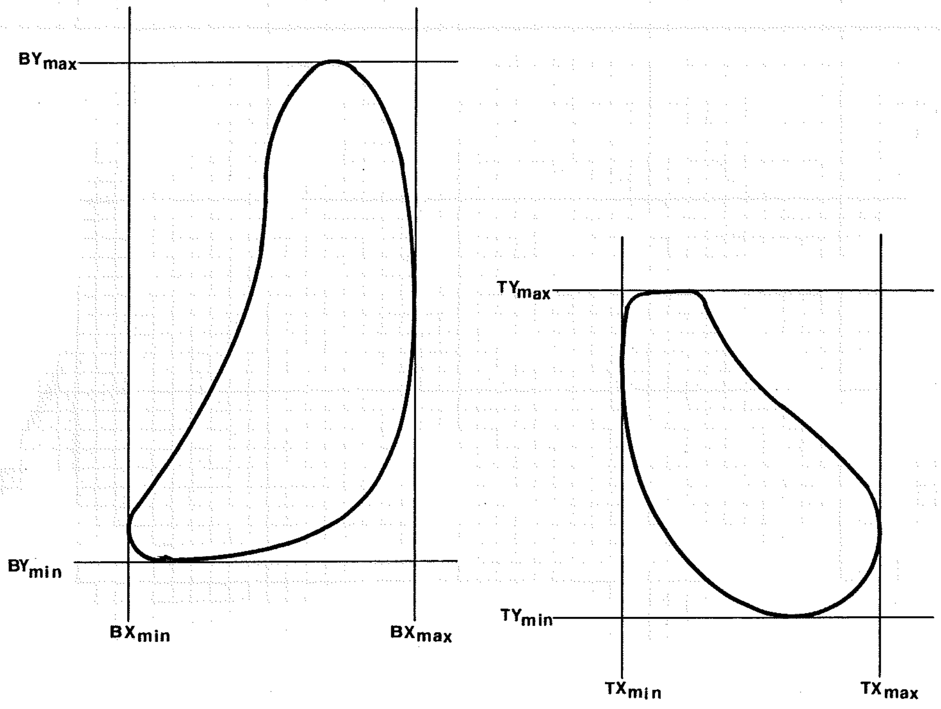


Figure 14

Overlap Test

The rectangular windows definitely do <u>not</u> overlap if:

$TY_{max} < BY_{min}$ or

$TY_{min} > BY_{max}$ or

$TX_{max} < BX_{min}$ or

$TX_{min} > BX_{max}$.

On the other hand, if all four inequalities are false, the windows necessarily overlap.

The algorithm works well for mildly complex cases, with optional user interaction capabilities to handle complex branchings.

Chapter 4

BRAIN CONTOUR DATA

ORIGIN AND DESCRIPTION

The brain data to be used for example purposes in this thesis has an interesting history. In 1967, the first of several movies was made of a human brain at the University of California at San Diego. Using the process of cinemorphology, an entire human brain was placed in a microtome capable of shaving off a slice 25 microns thick. After each slice, a frame of movie film was shot. The entire brain was sliced through, with each successive newly exposed surface recorded on film. Every nth frame of the movie was exploded photographically and outlines traced of each distinct brain structure. Figure 15 shows a cortex contour. In all, 22 separate structures were recorded. The contour outlines were laid on an acoustic tablet and a graduate student (of course!) selected appropriate nodes with the acoustic pen. The nodes were then digitized and recorded.

This digitized data base was accessed by a line drawing graphics package which can produce real time line drawing movies (in color!) on an Evans and Sutherland Picture System.

Each contour plane is referred to as a "page", and there are 98 pages total, ranging from page 3 at the top of the cortex to page 100 at the bottom of the brain stem. The data base is

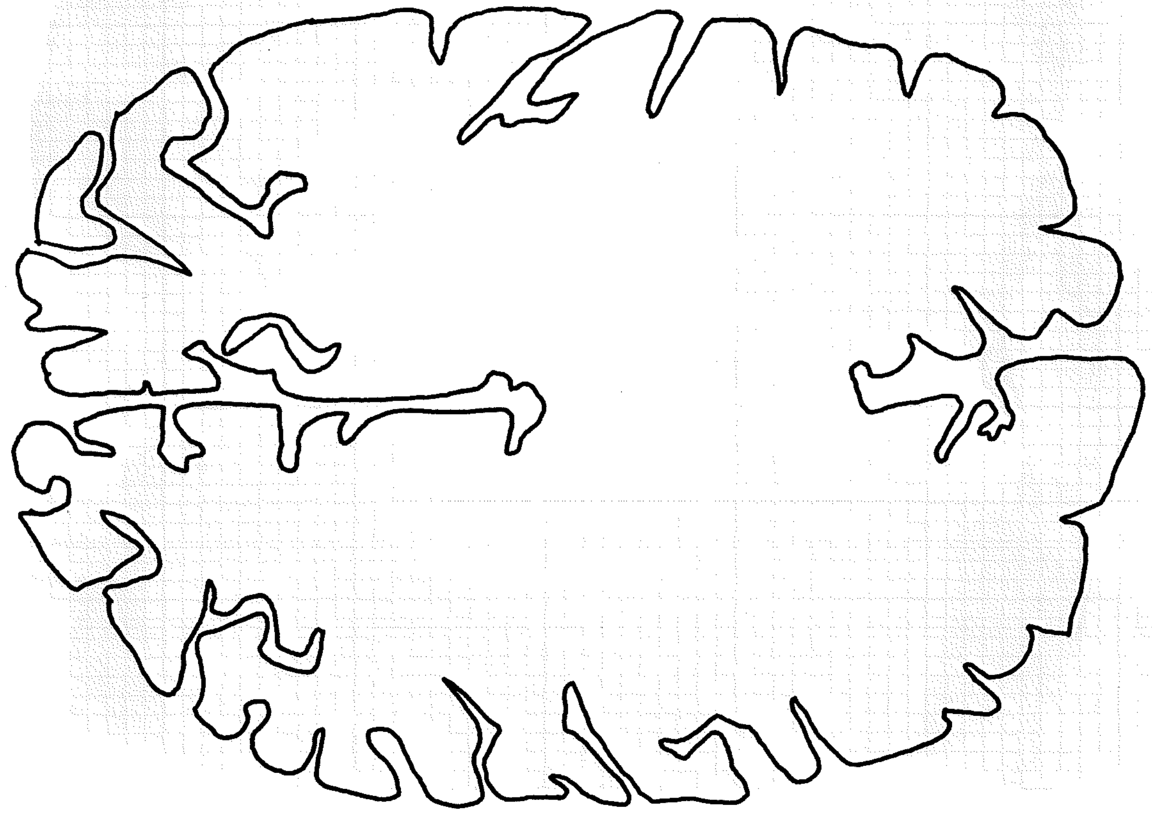


Figure 15

Contour Line of Brain Cortex

massive -- totaling 78,651 nodes. Table 1 shows the number of nodes per structure, as well as their page limits. Pages are spaced approximately 1/25" apart, corresponding to a brain that is roughly 4" tall.

Table 1

INDEX TO BRAIN DATA

	STRUCTURE	PAGES	NODES
1	Cortex	3-78	52,870
2	Caudate	30-61	1,922
3	Ventricles	30-84	4,707
4	Fornix	35-57	1,081
5	Putamen	37-54	1,075
6	Thalamus	38-58	1,248
7	Corpus Callosum	41-46	35
8	Globus Pallidus	43-52	725
9	Hippocampus	47-66	1,576
10	Hypothalamus	50-61	400
11	Pineal Body	51-55	92
12	Subthalamic Nucleus	50-56	142
13	Red Nucleus	52-60	238
14	Brain Stem	54-100	1,960
15	Amygdala	55-63	395
16	Substantia Nigra	56-62	243
17	Cerebellum	59-99	6,800
18	Optic Chiasm	60-62	78
19	Mammillary Bodies	57-59	87
20	Mesopallium	19-69	2,385
21	Mammillothalamic Tract	43-56	303
22	Septum	42-49	289

BRAIN DATA FORMAT TRANSMUTATION

The brain contour data arrived at BYU on magnetic tape as 16bit integers in binary format. The data are grouped into 22 structures, which in turn are divided into segments. A segment is a string of contour points and a contour line is formed from one or several segments. Segments represent portions of surfaces which are shared by two structures. Hence, a contour line that is composed of say 5 segments is bordered by 5 neighbors.

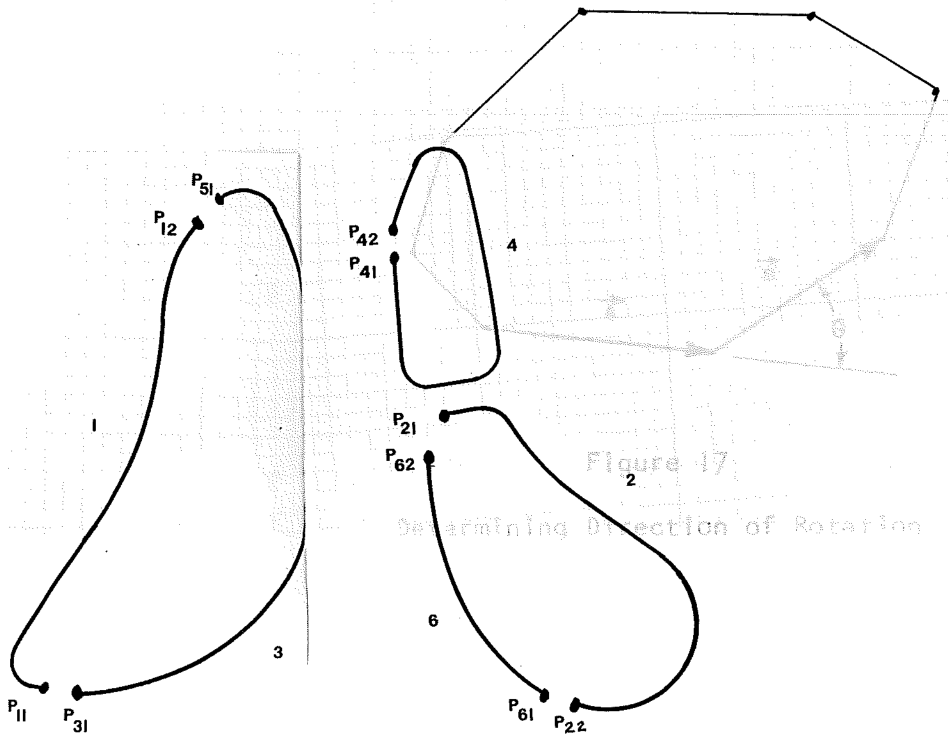
Segment definition, which was initially imposed on the data to facilitate line drawing display, somewhat hampers triangulation because all contours must be reconstructed from their constituent segments before triangulation can commence. The problem is aggravated because the segments are randomly sequenced and, furthermore, no convention is observed in clockwise and counterclockwise ordering of nodes.

Segment definition is illustrated by this typical configuration where 3 closed contour loops are defined by 6 segments:

Figure 16
Segment Definition

Notation wise, Pn_1 is the first node of segment n and Pn_2 the last node of segment n. The transmutation algorithm begins by assigning segment 1 to loop 1. A search is made for the nearest neighbor of P_{12} which is P_{51}, and segment 5 is appended to segment 1. Next, the nearest neighbor of P_{52} is sought. Its nearest neighbor is P_{32}. This indicates that segment 3 is sequenced in an order contrary to that of segments 1 and 5. Consequently, segment 3 is joined to loop 1 in reverse order. Since P_{31} neighbors P_{11}, the loop is complete. A flag is set for loops 1, 3 and 5 preventing future assignment. This logic repeats until all segments are joined to a loop.

It is important to impose the convention that loops run uniformly in a clockwise (or counter-clockwise) direction. To enforce this convention, all nodal angles of a contour line are summed for monitoring in the following manner:



Theta is the angle by which each succedent vector deviates from a straight line. The sum of all such angles will be 360 degrees for counterclockwise sequencing and -360 degrees for clockwise. Theta is computed from vector cross and dot products.

$$\sin\theta = \frac{\vec{A}\times\vec{B}}{AB}$$

$$\cos\theta = \frac{\vec{A}\cdot\vec{B}}{AB}$$

$$\theta = \begin{cases} \sin^{-1}(\sin\theta) & , \cos\theta>0, \\ \sin^{-1}(\sin\theta)+90 & , \cos\theta<0 \text{ and } \sin\theta>0, \\ \sin^{-1}(\sin\theta)-90 & , \cos\theta<0 \text{ and } \sin\theta<0. \end{cases}$$

In a few instances, the brain data invalidates this approach by having a contour line cross itself like this:

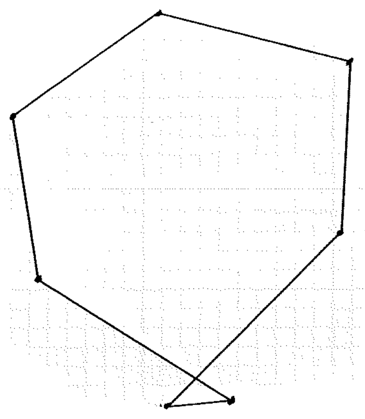


Figure 18

Brain Contour Error

This error causes a figure 8 which results in $\Sigma\theta$ approaching 0. Often, this causes a violation of the rotation convention.

Chapter 5

ECONOMIZING

NODE ELIMINATION

If a data base is too refined (i.e. contains nodes you could do without) it is desirable for reasons of economy to eliminate the less essential nodes. Consider this node:

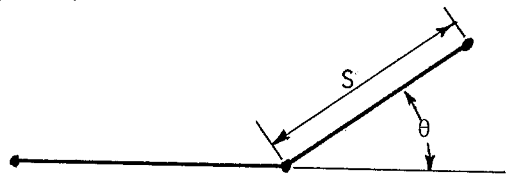


Figure 19

Node Elimination Parameters

The node is accepted (or rejected) upon the following criteria:

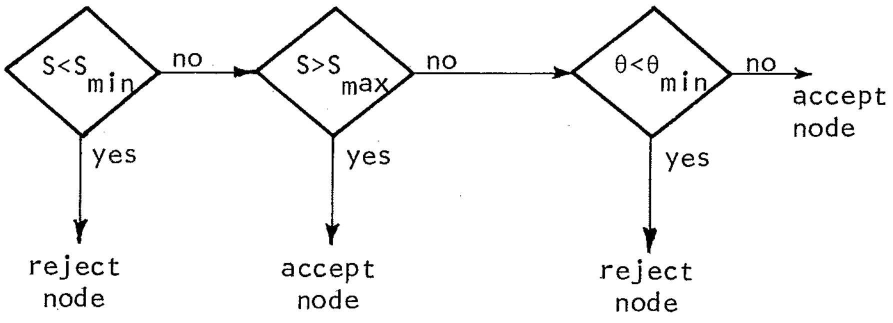


Figure 20

Node Reduction Flow Chart

S_{min}, S_{max} and θ_{min} are user definable parameters. Every node is screened using this logic. To assure acceptance of every node, all three parameters may be set to zero.

This is a logical place to interject a few thoughts on interpolation of new nodes. Since a digitized contour line is an approximation comprised of a series of straight line segments, it is reasonable to assume, and important to prescribe, that nodes are selected such that the digitized approximation does not deviate intolerably from the actual contour line. This would imply a correlation between nodal density and contour line curvature. That correlation suggests that it is desirable to re-distribute nodes around the contour loop according to a curvature vs. node density function using curve fitting procedures. This is an appealing thought, since it would reduce angularity in the continuous tone display. This would be great, provided the actual surface isn't angular. Of course, however, that assumption is not always valid. Take, for example, a simple four node contour definition of a square. If curve fitting were imposed in an attempt to extrapolate extra nodes, the result would tend towards a circle. Angularity is reduced at the expense of accuracy. In conclusion, the burden of providing acceptable data rests with the person who actually does the digitizing.

QUADRILATERAL FORMATION

For economy of storage, it is desirable to join pairs of adjacent triangles together into quadrilaterals. Not all ajoining triangles are thus combined. The decision is made on the basis of how warped the resulting quadrilateral would be, that is, the angle by which the two triangles are out of plane. That angle is easily found using vector algebra.

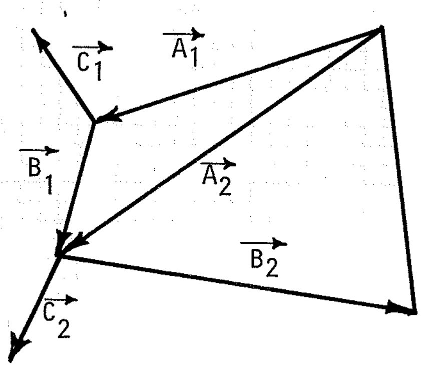


Figure 21

Warp Angle Determination

$$\vec{C}=\vec{A}\times\vec{B}$$

$$\sin\alpha = \frac{\vec{C}_1 \cdot \vec{C}_2}{C_1 \; C_2}$$

If $\sin\alpha < \sin\alpha_{max}$, the two triangles are redefined as a quadrilateral. α_{max} is user definable and defaults to 45 degrees.

Chapter 6

FORTRAN IMPLIMENTATION

A few explanatory remarks are offered here to the reader who is bent on deciphering the source code.

As mentioned, the brain data arrived at Brigham Young University in the form of 16 bit integers on tape in binary format. To facilitate use on the DEC-10 computer, these data were re-formatted into 7 bit ASCII data files, one file for each of the 22 structures, with 8 integers per line. The first two integers of a file comprise the "structure heading", the first integer being the structure number, and the second being the number of segments in the structure. These two integers are ignored by the triangulation program.

The next four integers form the segment header of the first segment. The first integer of the segment header is the page number or horizontal level of the segment. The page numbers range from -3 to -99. The Z coordinate of the segment is computed from the formula RZ=-(Z+51)*450*SCALE where RZ is the Z coordinate, Z is the page number, and SCALE is the scale factor. The second and third integers in the segment header are ignored. The fourth integer, NPL, is the number of nodes in the segment.

Immediately following the segment header are the X-Y coordinates of the segment nodes, totalling NPL pairs. The next segment header immediately follows the last node of the preceding segment, so there is an uninterrupted string of integers from start to finish in the data file. Nodes of all segments of the same page are stored in array P2. The segments are joined together to create closed loops (as described in chapter 4) and stored in array P3. Node elimination is imposed, and the nodes are finally stored for triangulation in array P.

Two pointers are used in accessing the nodes in array P. The Loop Pointer - LPP - indexes the global loop numbers of the first loop on any contour level. Pointer P1 addresses the global node number of the first node on a loop. For example, the X coordinate of the n^{th} node of the j^{th} loop on the k^{th} contour level would be:

```
P((P1(LPP(k)+(j-1))+n-1)),1)
```

Knowing these conventions, the fortran coding should be relatively lucid. The coding, written for use on a DEC-10 system, is given in the appendix. An additional assembly language Tektronix interface, GRTEK.REL, must also be loaded.

Chapter 7

EXAMPLE PROBLEM

A simple yet dramatic example of branching is provided by the caudates - the symmetric pair of brain structures shown below.

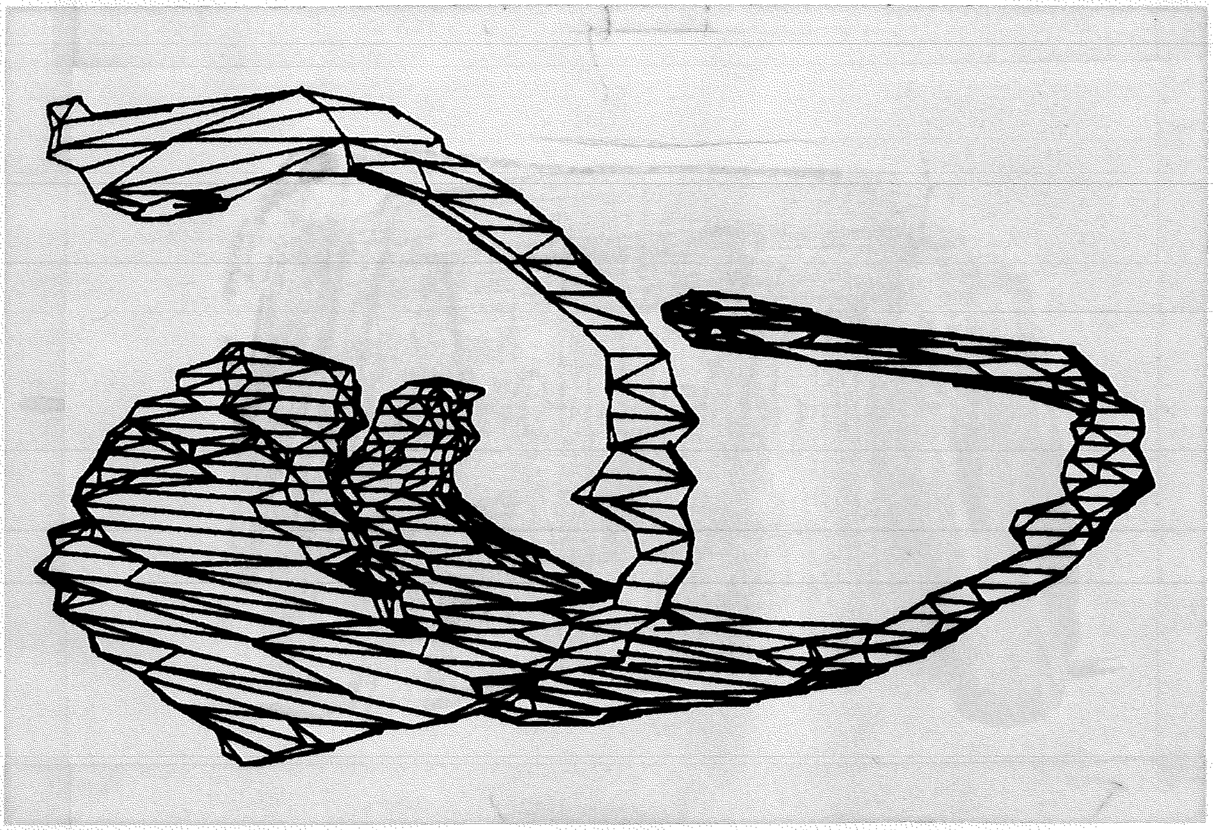


Figure 22

Caudates

Triangulation of the caudates occurred as follows. User statements in the computer dialogue have been underlined to distinguish them from computer generated prompts. Refering to figure 23 the computer begins by asking if the data it is about to read is brain data. It is, and the caudate data file name, B2, is given. Next, the menu of commands is printed, of which PARAMENTERS is

```
BRAIN DATA? Y

FILENAME OF INPUT DATA? B2

READ> HELP

PARAMETERS,TOTALS,LEVEL,SCALE,EXIT
BRANCH,MANUAL,CLOSE,DEVICE,KLOCKWISE

READ> P

MINIMUM SEGMENT ANGLE=  15

MIN. &  MAX. SEGMENT LENGTHS: .3,2

READ> L

Z-SPACING= 1

LEVEL RANGE= 1,60

DATA ENDED AFTER LEVEL 32
BRANCH> HELP

AUTOMATIC,WARP,MANUAL, INSPECT,SINGLE,CAP,EXIT,TOTALS
BRANCH> T

 955 NODES.    1 ELEMENTS
BRANCH> A

START WITH WHICH LEVEL? 1

POST-EDIT? Y
```

Figure 23

Computer Dialogue

selected. We opt to set the minimum segment angle to 15 degrees, S_{min} to .3 and S_{max} to 2. (These parameters are explained on page 26). Having returned to the READ> prompt, we now choose to read in the data, being satisfied with the default values for SCALE, CLOSE, and KLOCKWISE. After setting the LEVEL parameters, the computer goes to work reading in contour segments, reconstructing them into loops, thinning them out, and assuring that all loops run in a clockwise direction. The algorithm encounters end of file before 60 levels are read in, and informs us that all 32 available levels have been read in and processed. The TOTALS command receives the response that there are 955 nodes in array P, which means that over half of the available 1922 nodes have been thinned out by the node elimination algorithm.

The Caudates are quite regular, requiring little or no interaction to triangulate properly, so the AUTOMATIC option is invoked beginning, as usual, with level 1. As a safeguard, post-editing is requested. If the data were unquestionably obedient, the post-editing could justifiably be circumvented, but this way good results are guaranteed.

The algorithm now proceeds to first determine connectivity, then to triangulate all loops implicated in the the window overlap connectivity check, and finally to display the resulting panel definition. The user glances at each successive display and grants acceptance with a carriage return, or occassionally rejects the triangulation, as the case may be. (Usually it is immediately clear when triangulation is unacceptable. Normally, failure occurs in areas where the two loops are excessively dissimilar, and the resulting panels are often bizzare).

Two of several simple one-on-one triangulations are shown in figures 25 and 26 , each of which is accepted. However, the branching loops in figure 27 are triangulated incorrectly, and a change is requested. This change is granted through erasure of the screen, re-drawing of the untriangulated loops, and issuing of the TRIANGULATE> prompt. The INTERACTIVE command is given. As always, only the first letter of the command is required. Referring to figure 28 the nodes are numbered for identification - even for the top loop and odd for the bottom.

The INTERACTIVE command allows the triangulation to be controlled by allowing specification of nodal delimiters between which triangulation will occur. For example, if the delimiters 1,1 for the top and 1,2 for the bottom were chosen, only one triangle would be formed. Basically the selection of delimiters is a trial and error process. The user delimits as large a span as reasonable. If the resulting triangulation is adequate, great! If not, try again with a smaller span. A general rule might be to procede quickly over sections where the two contours are similar, and cautiously over sections where they are dissimilar.

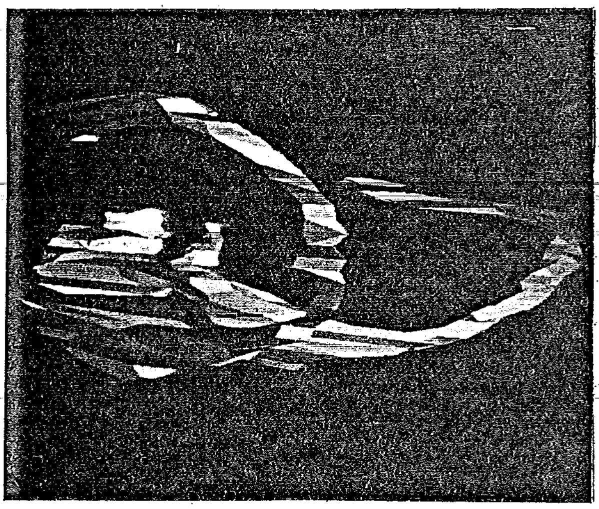
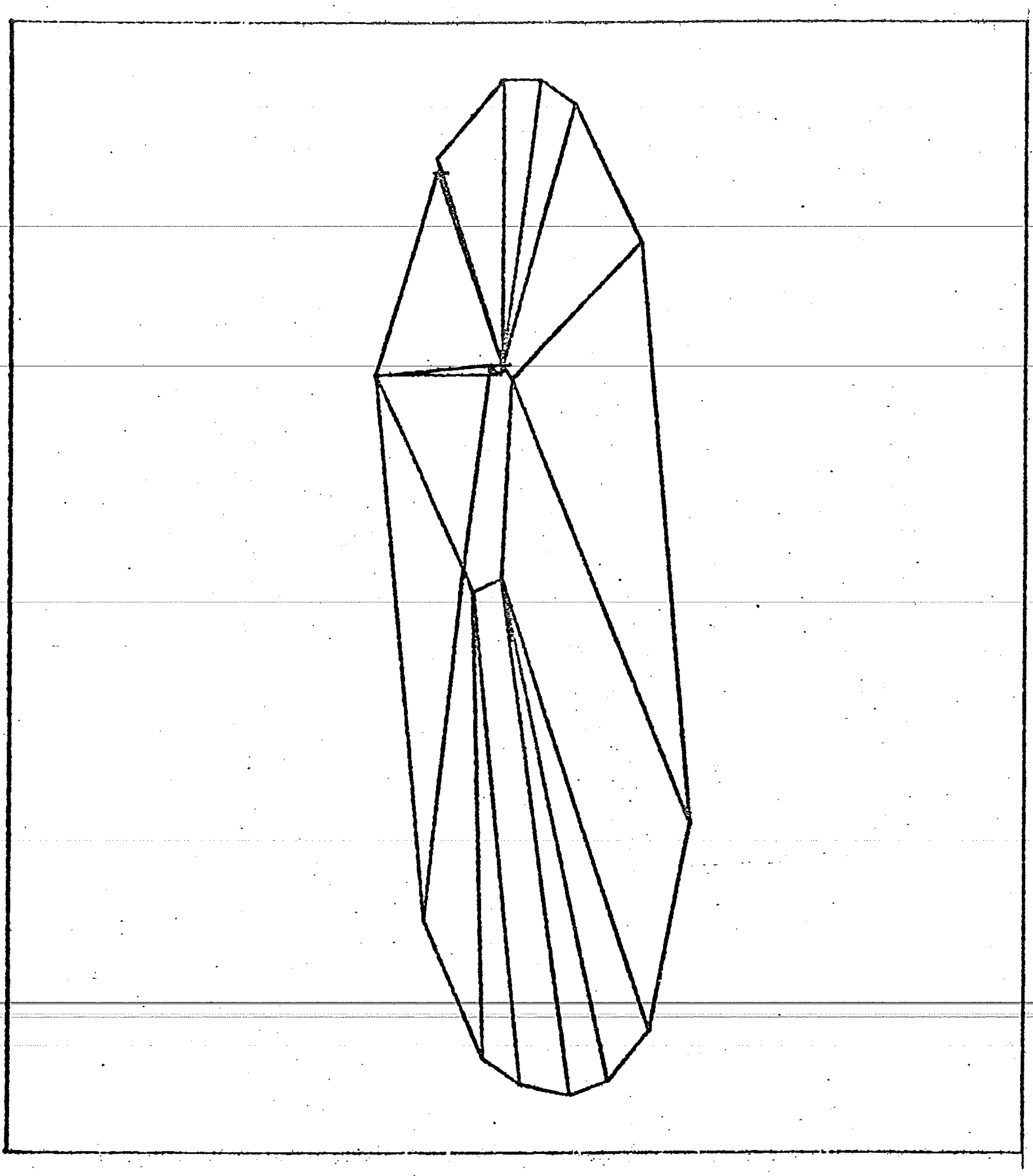


Figure 24

Continuous Tone Rendering
of Caudates

Referring to figure 26, the triangulation was successful through node 17 of the bottom loop, but then failed to traverse the branch properly. This suggests the selection of delimiters 1,28 for the top and 1,17 for the bottom. The resulting triangulation shown in figure 28 looks good. Now, one more span (top:28,33 ; and bottom:17,20) should suffice, and figure 29 confirms the hope. Now that the entire circuit is complete, AUTOMATIC mode is re-entered and triangulation proceeds smoothly to the conclusion. Upon exiting from the program, the panel definition is written into a disk file, available for display using MOVIE.BYU. A continuous tone image created by that panel definition is shown in figure 24.



CHANGES? NO

Figure 25
Example

CHANGES? <u>NO</u>

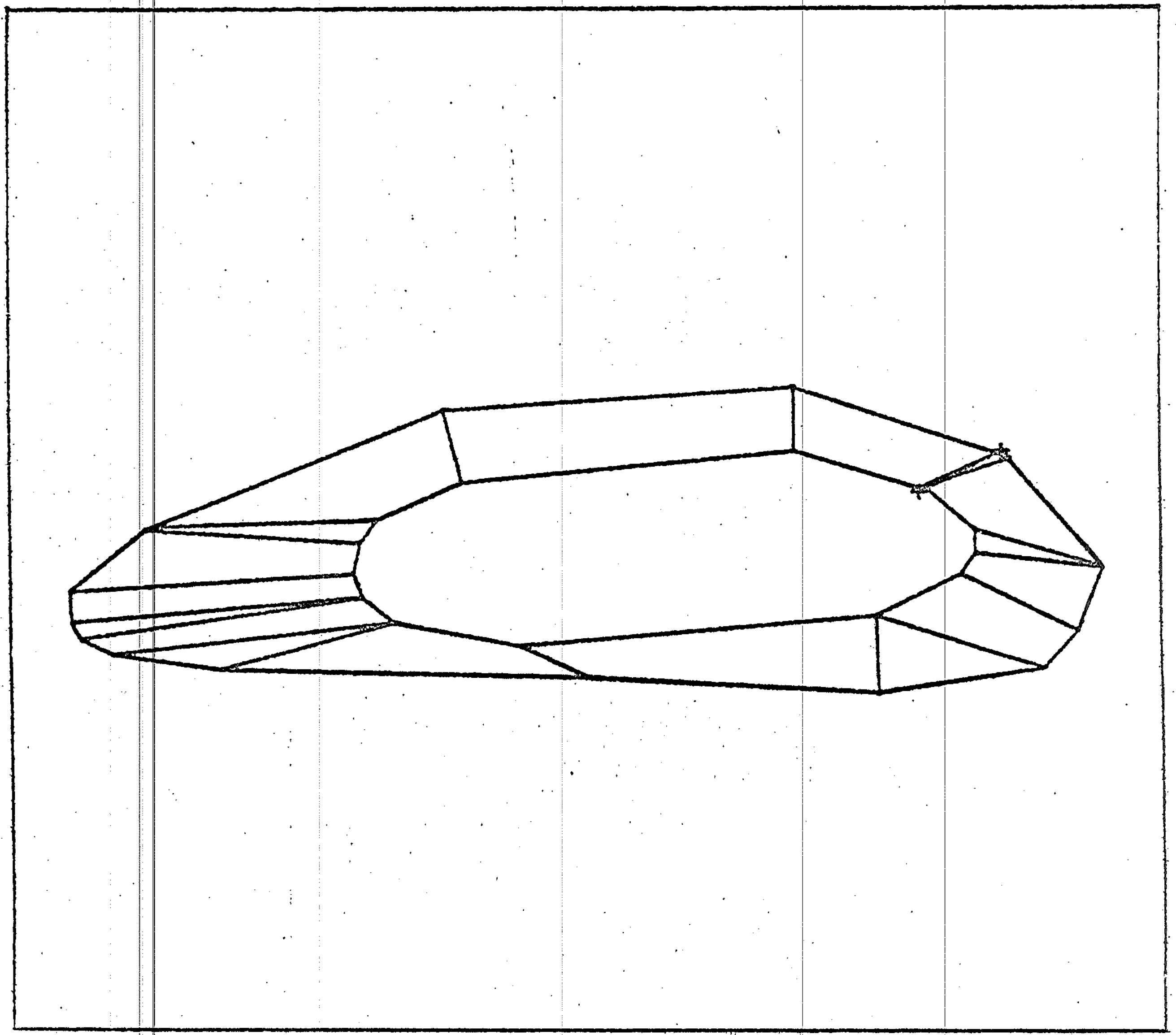


Figure 26

Example

CHANGES? YES

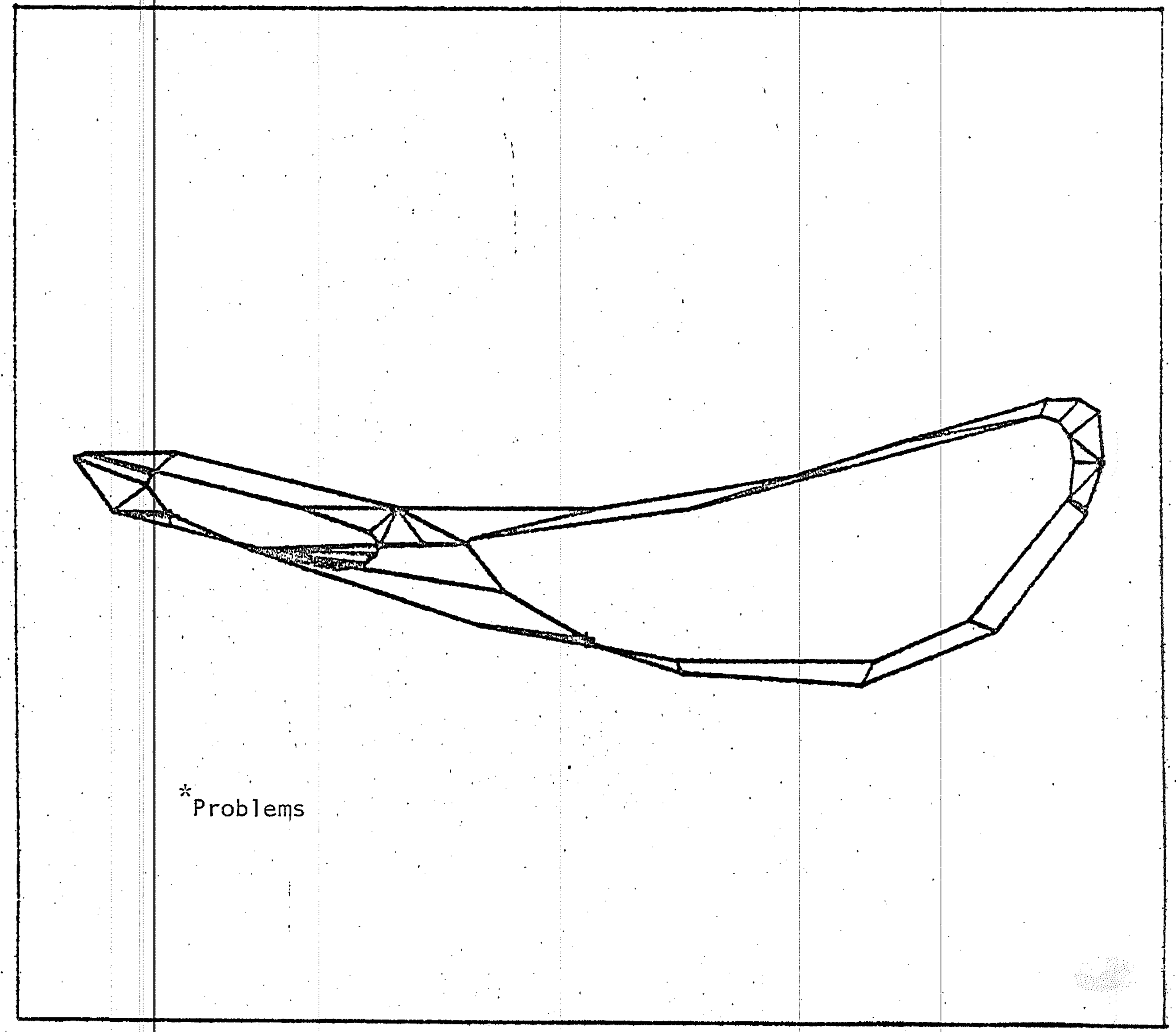


Figure 27
Example

```
TRIANGULATE>I

TOP:  1-33
1,28

BOTTOM:  1-20
1,17

CHANGES? NO
```

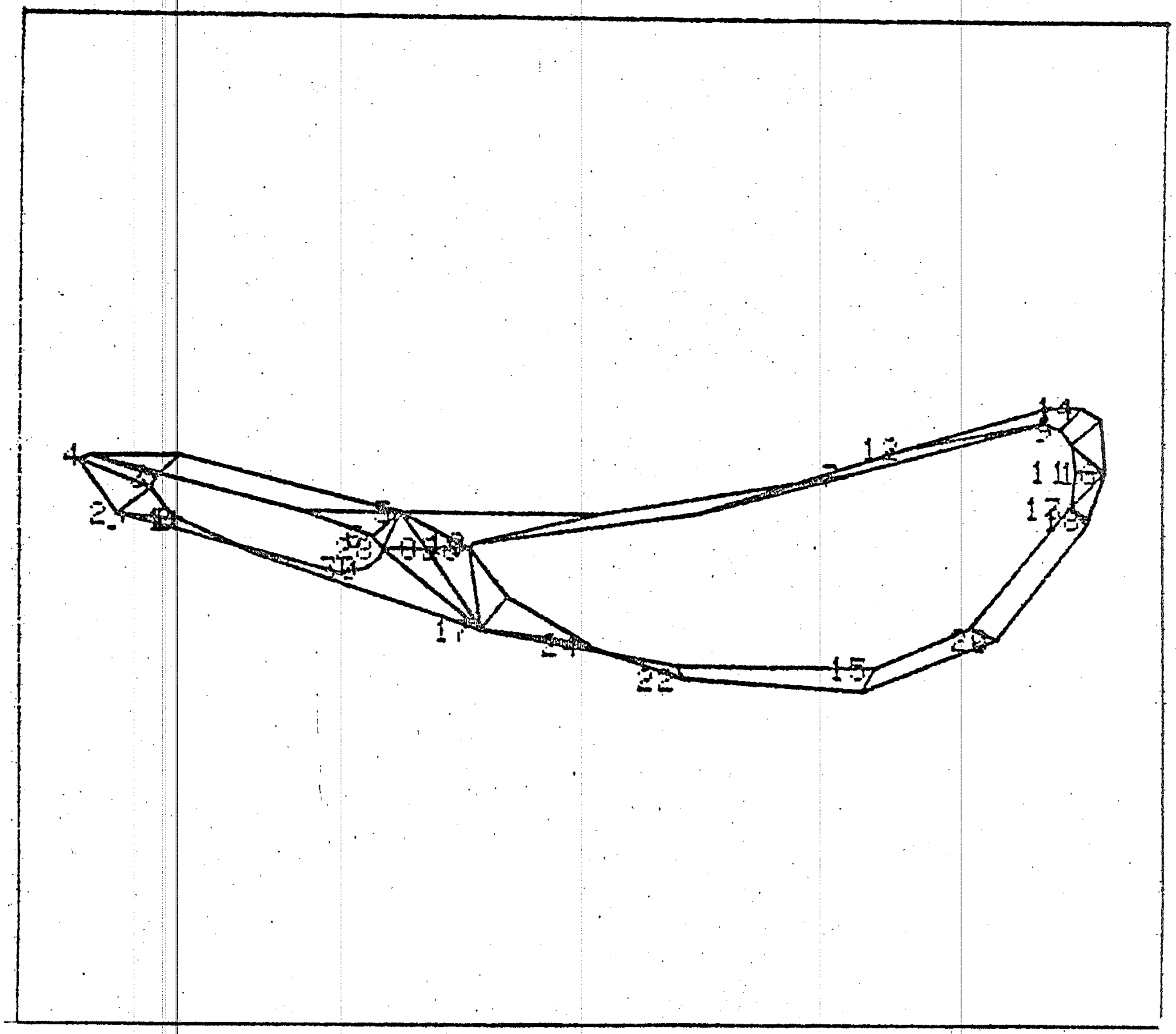


Figure 28

Example

TOP: 28-33
<u>28,33</u>

BOTTOM: 17-20
<u>17,20</u>

CHANGES? <u>NO</u>

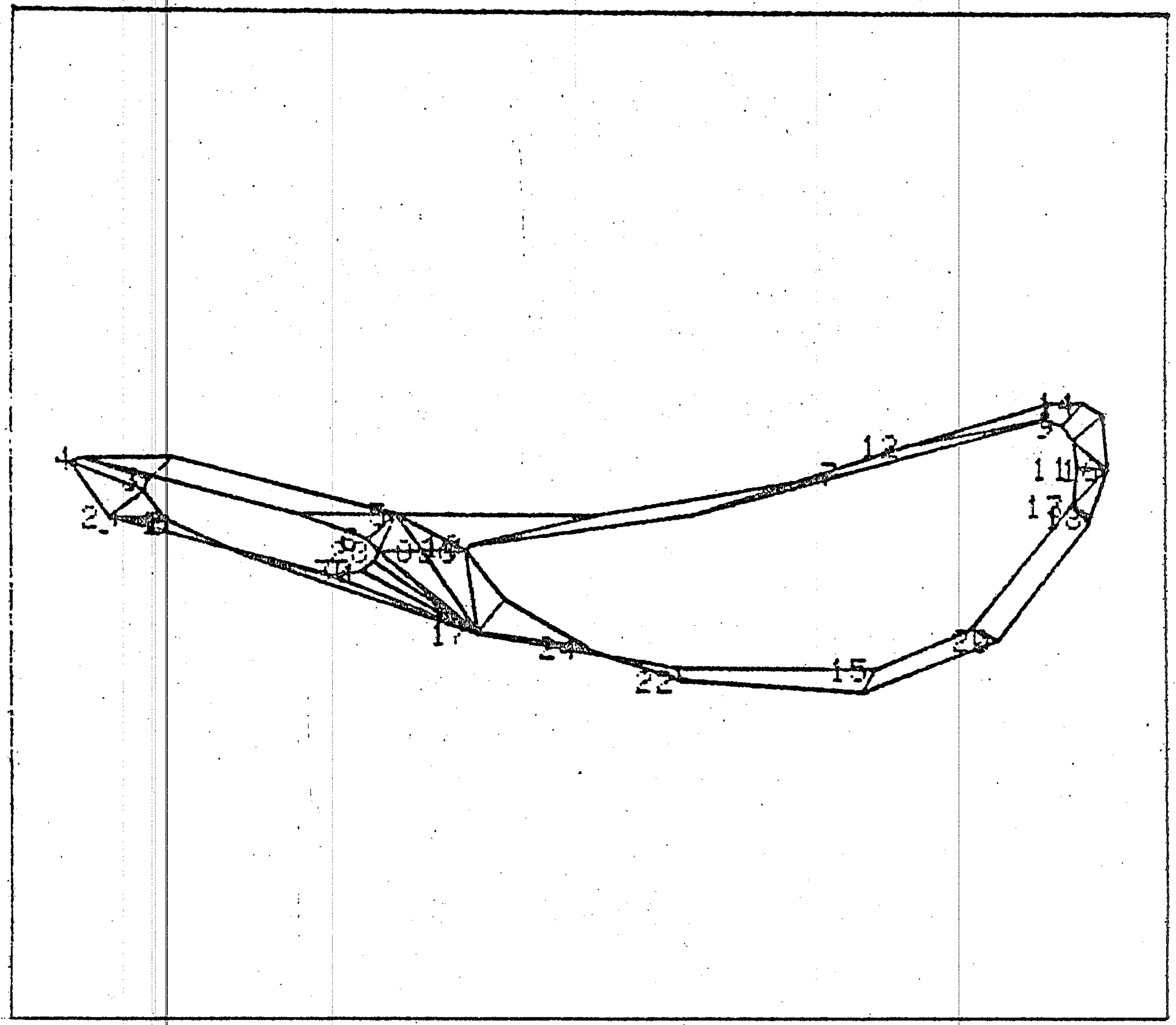


Figure 29
Example

Chapter 8

PICTURES

This chapter presents examples of the finished product. It is difficult to judge how true to life the images are, due to the highly esoteric nature of the subject matter. Nonetheless, it is generally evident that the triangulation algorithm has performed reasonably.

The first structure presented is the Brain Stem. The Brain Stem was a straighforward triangulation problem. There are no branches, and there is no sreious variation in the shape of its respective contours. The only difficulty was a case of illeagle rotational direction, as described on page 25. Other than that, the entire triangulation was handled automatically. Two panel definitions were generated for the brain stem. The first file was made with $S_{min}=S_{max}=\theta_{min}=0$, Z-SPACING=1, and LEVEL RANGE =5,36. The resulting panel definition had 1609 nodes and 1991 panels. The second brain stem panel definition has 378 nodes and 492 panels. It was generated with $S_{min}=.2$, $S_{max}=1$, $\theta_{min}=15$, Z-SPACING =4, and LEVEL RANGE =5,40. Line drawings are shown in figures 30 and 31, and continuous tone images in figures 34 and 35.

The next pair of images, figures 36 and 37 are detailed studies of the thalamus. Here, 4 loops branch into 2, then 2 into 1. Figure 37 has a more biological look due to Gouraud smooth surface simulation.

Figure 38 is a striking composition of 6 different structures, each in proper relative orientation. The sructures are identified in figure 32 and figure 33 shows a line drawing. To enable so many parts, the larger structures have coarser panel definitions.

All of the preceding examples were triangulated with virtually no user interaction. The cortex slice, in figure 39 was not so oblidging. This image is presented to demonstrate the degree of complexity the algorithm can accomodate. To help orient the reader, this image represents about a $\frac{1}{2}$" thick slice of the cortex, centered about $\frac{1}{2}$" from the top of the brain. The many oddly shaped holes are due to the fact that the top most convolutions are decapitated in this view. This panel definition - shown here with smooth shading - consists of 1752 nodes with 1778 panels. This particular data did not cooperate with the automatic algorithm, and required nearly 2 hours time to interactively triangulate. The difficulty was not so much the complexity of the shapes, but the dissimilarity between adjacent contours. Also, the data had a disproportionate number of glitches. Nonetheless, the continuous tone image is quite convincing.

One final image is the panoramic shot of Mt. Timpanogos as it might be seen from an airplane flying to the west of Timp. This is submitted to illustrate a possible application of triangulation to display of topographic surfaces.

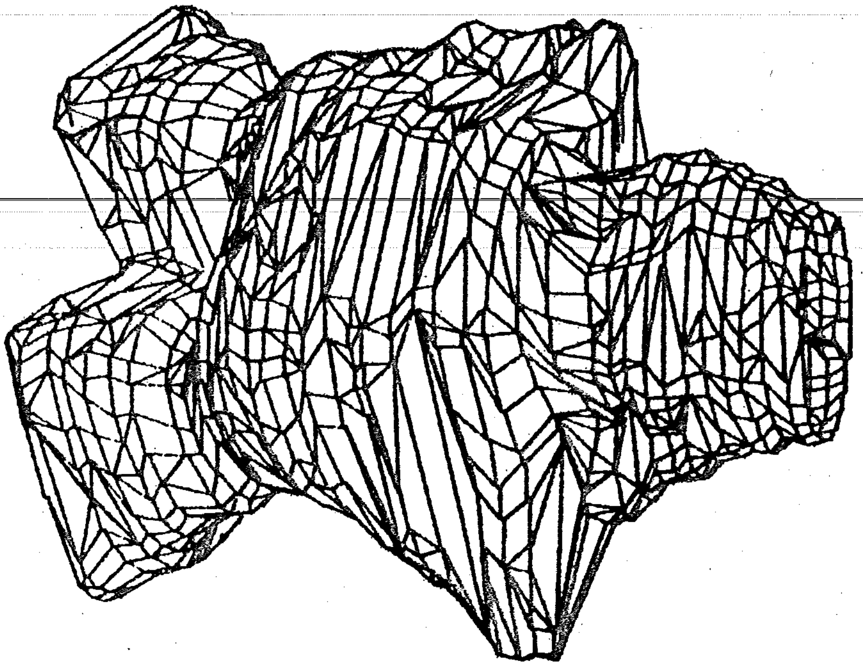


Figure 30

Brain Stem - 1991 Panels

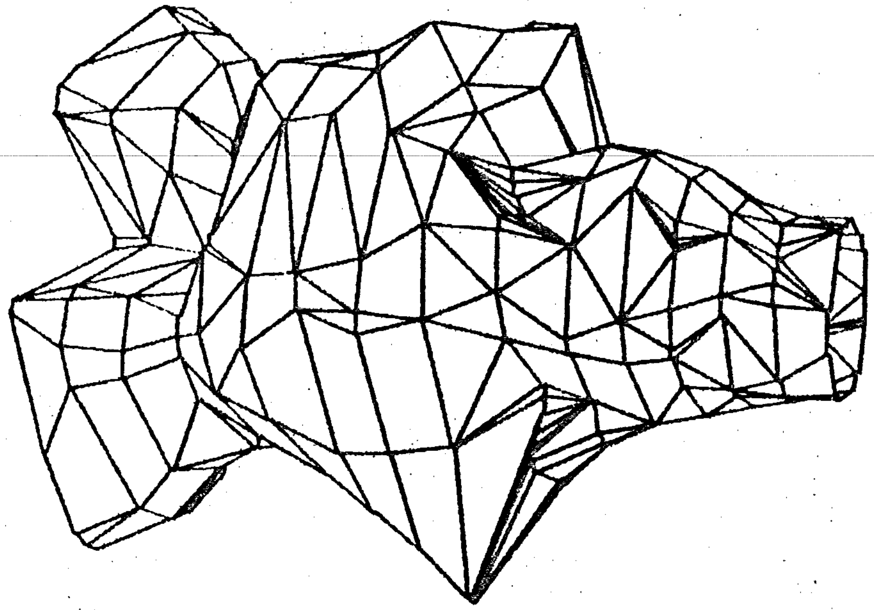


Figure 31

Brain Stem - 492 Panels

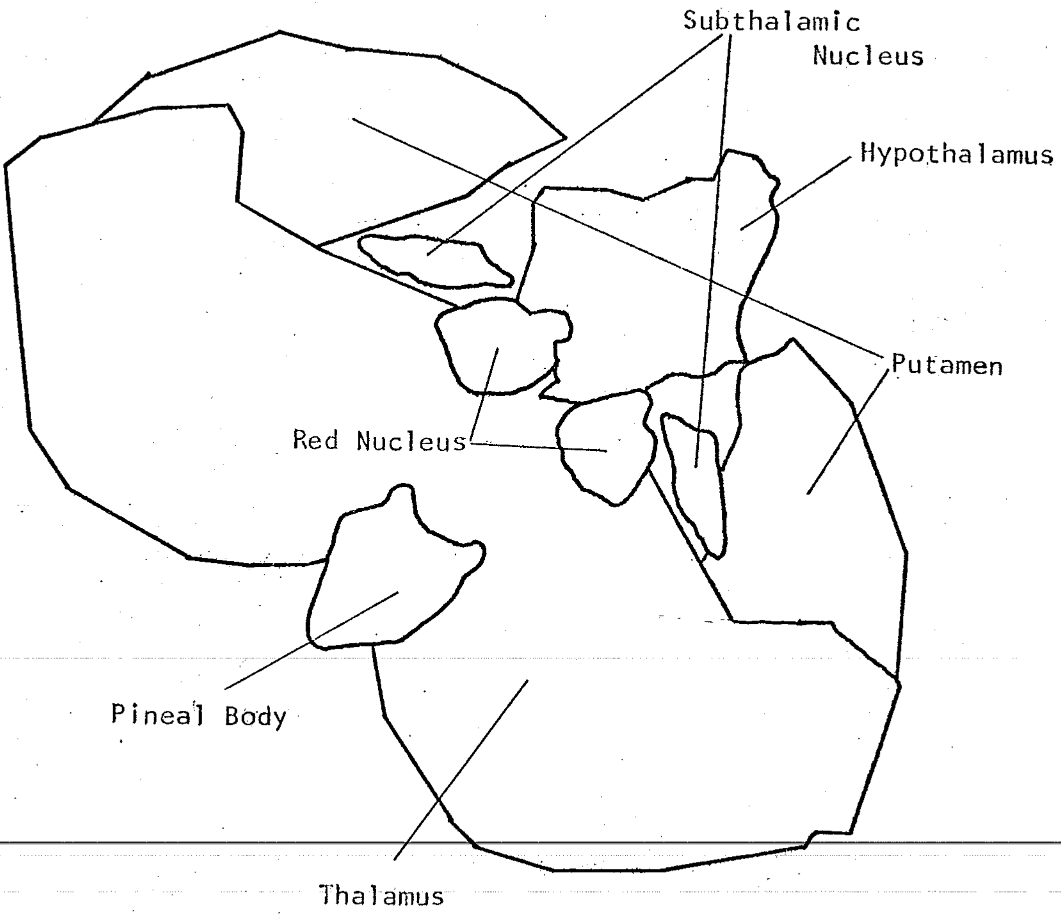


Figure 32

Labelled Composite View

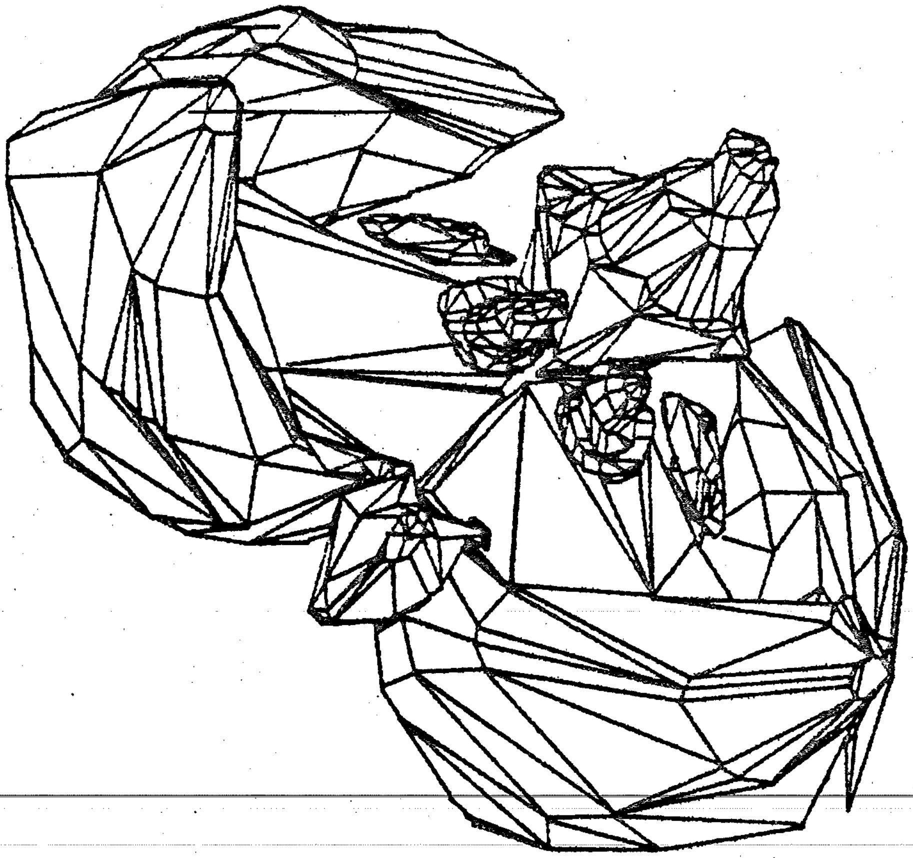


Figure 33

Composite View
Line Drawing

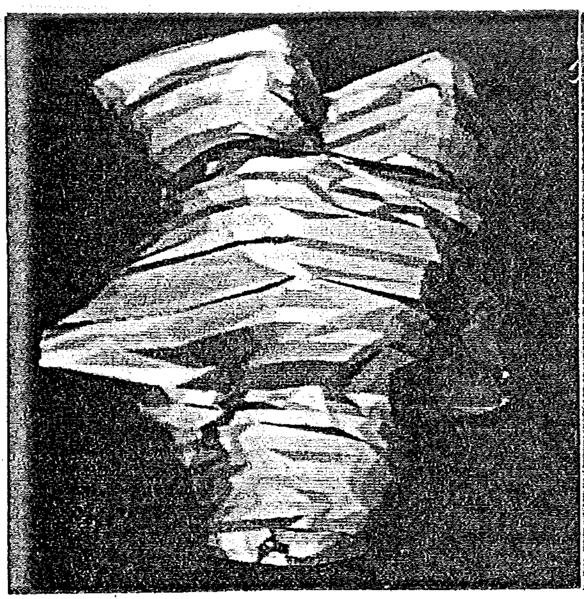


Figure 34

Brain Stem with
1991 Panels

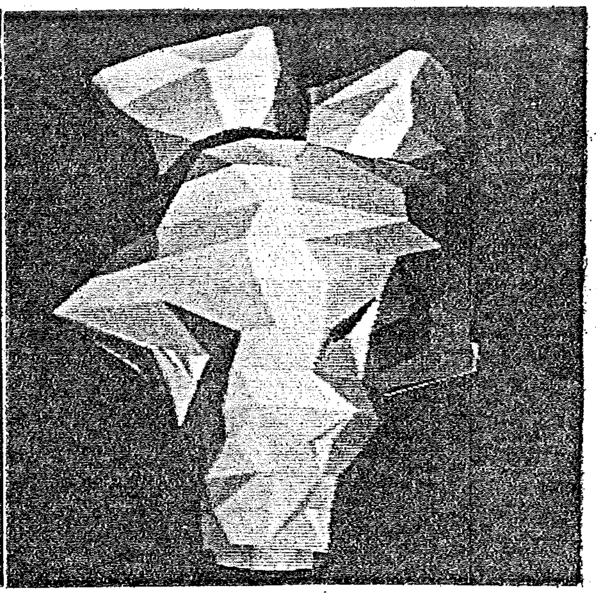


Figure 35

Brain Stem with
492 Panels

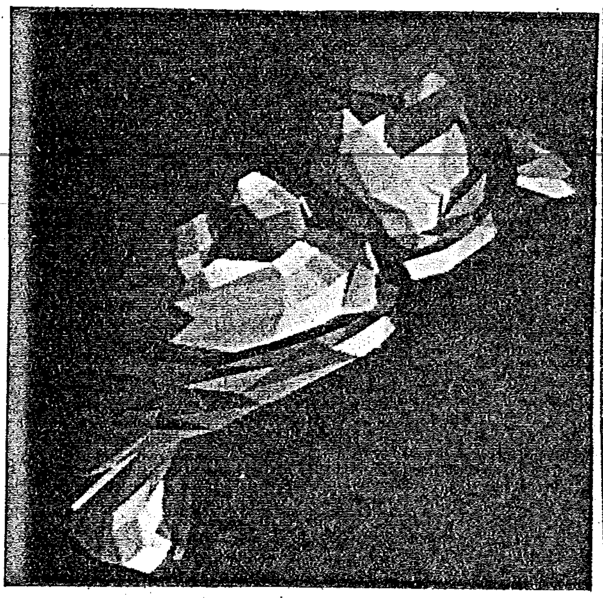


Figure 36

Thalamus with
Flat Shading

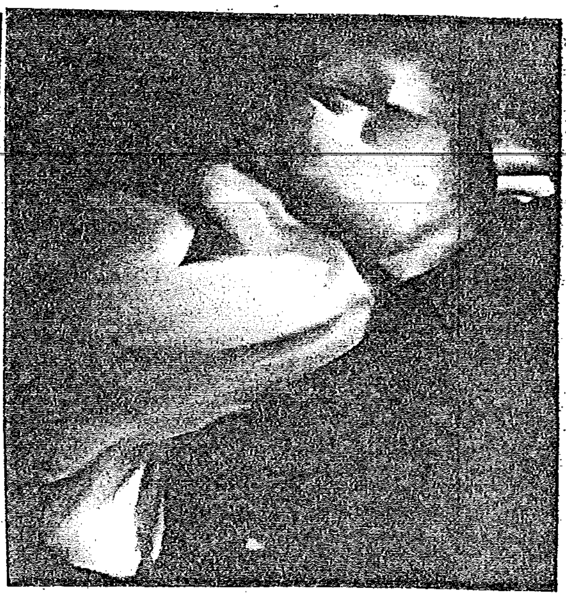


Figure 37

Thalamus with
Smooth Shading

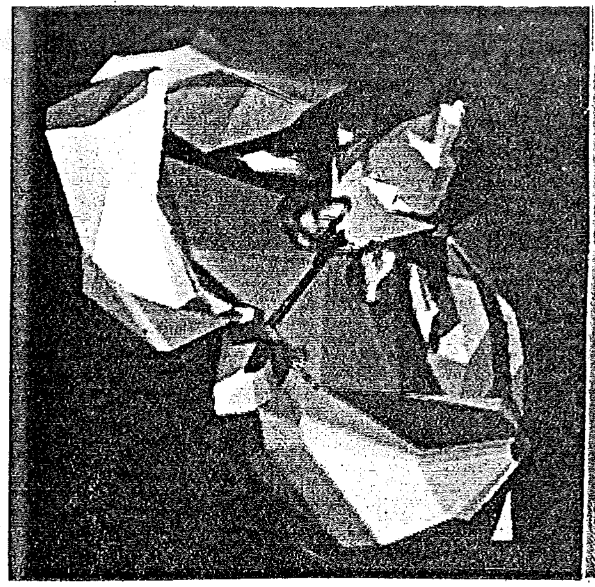


Figure 38

Composite View

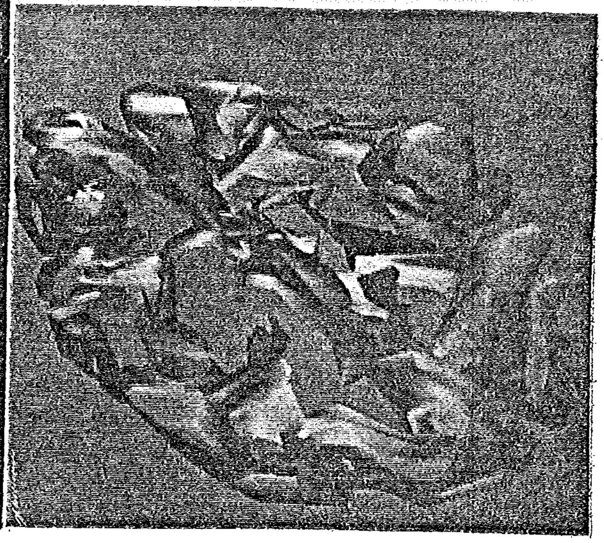


Figure 39

Cortex Slice

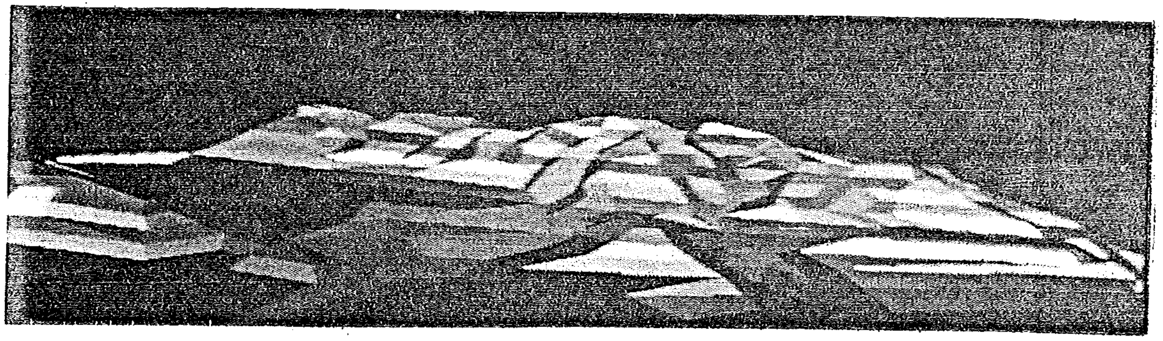


Figure 40

Mount Timpanogos - Flat Shading

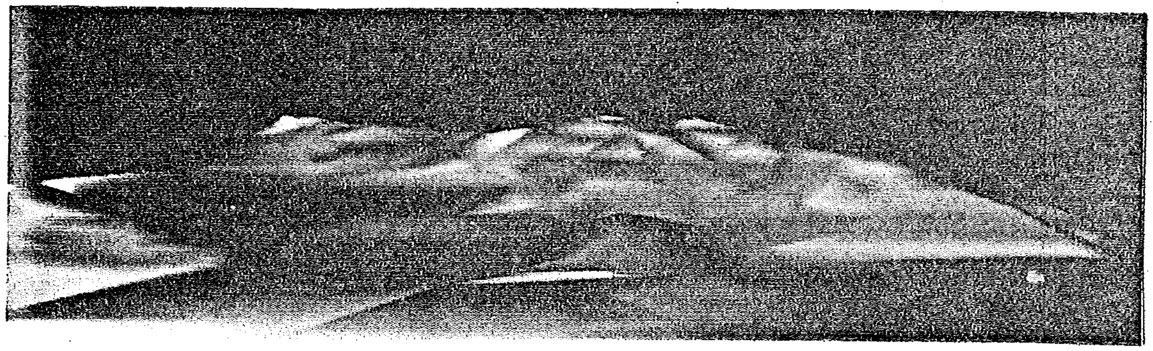


Figure 41

Mount Timpanogos - Smooth Shading

Chapter 9

CONCLUSIONS

This thesis purports to present a general solution to the problem of converting a contour definition of an arbitrary surface into a panel definition. That assertion is rigorously tested by the brain data, and experience with that highly complex data base lends credence to the claim of a general solution. Total user interaction capabilities virtually gaurantee a general algorithm.

Work might be done on reducing the amount of user dependence in the algorithm, though most reasonable cases require no interaction at all. Also, it would be helpful to improve graphical interaction by using, for example, a tablet to input interaction parameters. Study might also be made on how the economy parameters (S_{min}, S_{max}, θ_{min}, and α_{max}) effect the continuous tone image.

BIBLIOGRAPHY

Christiansen, Henry N. "Applications of Continuous Tone Computer-Generated Images in Structural Mechanics," Structural Mechanics Computer Programs - Surveys, Assessments, and Availability, University Press of Virginia, Charlottesville, Virginia, June 1974, pp. 1003-1015.

________. "MOVIE.BYU - A General Purpose Computer Graphics Display System," Proceedings of the Symposium on Applications of Computer Methods in Engineering, University of Southern California, Los Angeles, August 1977.

Fuchs, Henry. "The Automatic Sensing of 3-Dimensional Surface Points from Visual Scenes." Unpublished PhD dissertation, University of Utah, 1975.

Keppel, E. "Approximating Complex Surfaces by Triangulation of Contour Lines," Journal of Research and Development, IBM Vol. 19, No. 1 (January 1975), 2-11.

Newman, William M., and Robert F. Sproull. Principles of Interactive Computer Graphics. New York: McGraw-Hill, 1973.