(2)

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION	1b RESTRICTIVE MARKINGS
AD-A214 539	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
	5 MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 89-0972

6a NAME OF PERFORMING ORGANIZATION New York University	6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research/NL
6c ADDRESS (City, State, and ZIP Code) 550 1st Avenue New York, NY 10016		7b ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448
8a NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR	8b OFFICE SYMBOL (If applicable) NL	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-85-0341

8c ADDRESS (City, State, and ZIP Code) [redacted]
Building 410
Bolling AFB, DC 20332-6448

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
61102F	2313	A5	

11 TITLE (Include Security Classification)
Brain Peeling: Viewing the Inside of a Laminar 3 Demensional Solid

12 PERSONAL AUTHOR(S)
Eric Schwartz

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Reprint	FROM ____ TO ____		23

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

DTIC
ELECTE
NOV 22 1989
S B D

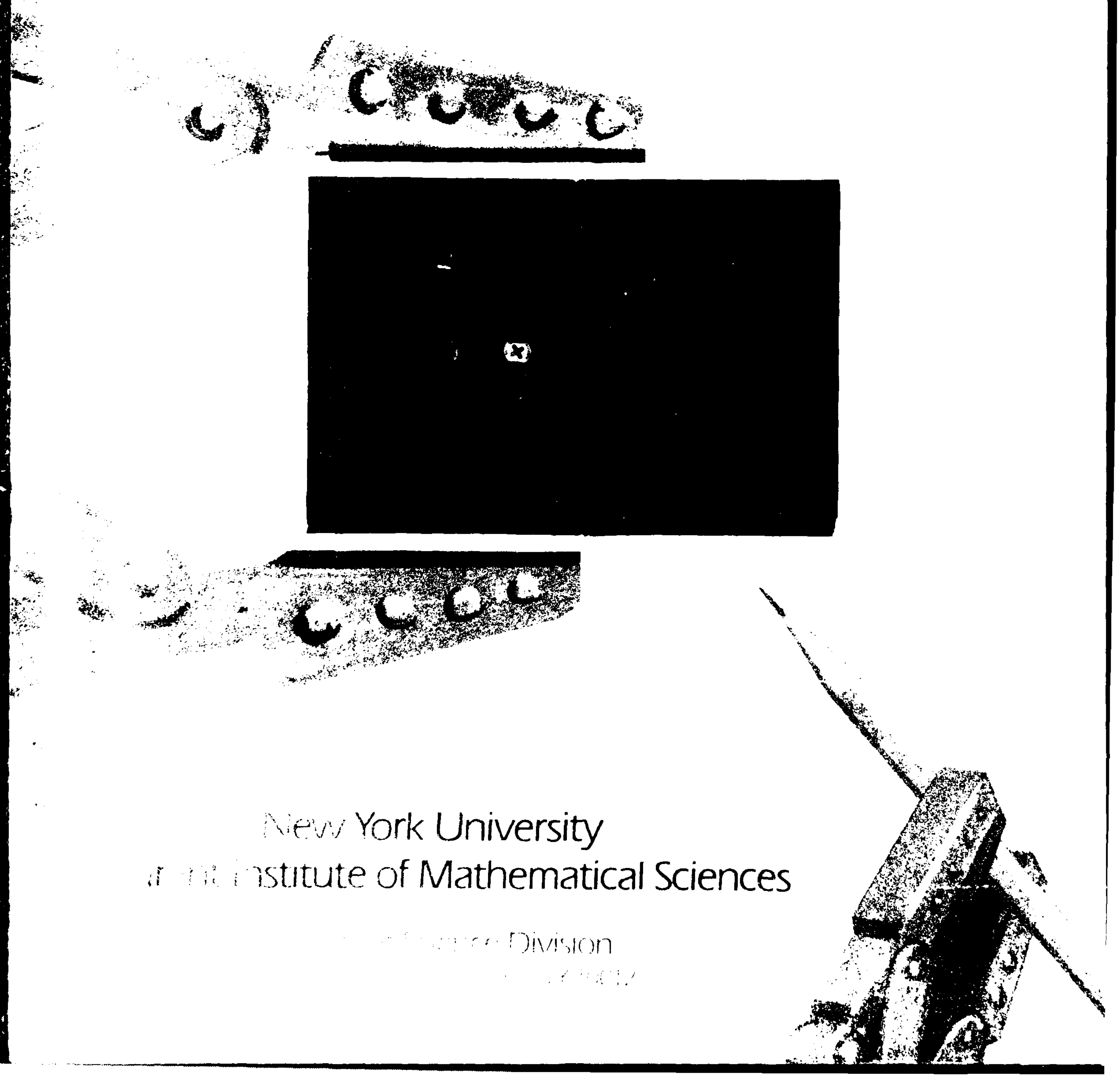
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL John F. Tangney	22b TELEPHONE (Include Area Code) (202) 767-5021	22c OFFICE SYMBOL NL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Robotics Research Technical Report



New York University
Courant Institute of Mathematical Sciences

Computer Science Division
[illegible]

Brain Peeling: Viewing the Inside of a Laminar Three Dimensional Solid

by

Carl Frederick
Eric L. Schwartz

Technical Report No. 393
Robotics Report No. 166
August, 1988

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by AFOSR 85-0341, System Development Foundation and the Nathan S. Kline Psychiatric Research Center.

BRAIN PEELING: VIEWING THE INSIDE OF A LAMINAR THREE DIMENSIONAL SOLID

†Carl Frederick

†‡Eric L. Schwartz

†Computational Neuroscience Laboratories
Department of Psychiatry
New York University School of Medicine
550 First Avenue
New York, N.Y. 10016

‡Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
715 Broadway
New York, N.Y. 10003

ABSTRACT

We describe a 3-dimensional surface tracking algorithm which is used to detect the interior laminar surfaces of a solid shell. Each of these surfaces is called a "peel". Successive peels may be generated, thus representing the solid shell by its tangential layers. This algorithm is based on voxel surface tracking methods, and solves the problems associated with transforming a surface tracking algorithm into a "brain peeler". We also discuss the properties of the voxel surfaces produced by this algorithm. Using the connectivity properties of these objects, we are able to convert voxel representations into polyhedral representations without human interaction. ~~We illustrate~~ this work with a high resolution reconstruction of a monkey visual cortex. Additional application domains of this work are in areas in which there is a natural laminar structure to a three dimensional solid, such as geophysics (earth strata).

August 18, 1988

supported by System Development Foundation, AFOSR 85-0341, and the Nathan S. Kline Psychiatric Research Center

BRAIN PEELING: VIEWING THE INSIDE OF A LAMINAR THREE DIMENSIONAL SOLID

†Carl Frederick

†‡Eric L. Schwartz

†Computational Neuroscience Laboratories
Department of Psychiatry
New York University School of Medicine
550 First Avenue
New York, N.Y. 10016

‡Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
715 Broadway
New York, N.Y. 10003

INTRODUCTION

The cortex of the brain is a laminar structure. The term "cortex" is derived from the Latin term for bark or rind. The cortex of the brain comprises the thin outer layer of the cerebrum. The cortex itself (typically about 1mm thick in monkeys) is composed of some six to ten layers†. In medical applications, the cortex is often imaged by a tomographic technique, such as CAT, NMR or PET scan. In experimental applications, it is much more common to image the cortex using serial sections of the brain, cut on a microtome (a more precise version of a sandwich meat-slicer.) Much work in voxel based graphics is motivated by CAT scan applications, due to the medical importance of these techniques. However, the experimental application associated with stained brain sections presents considerable algorithmic and scientific challenge. For one thing, physical brain sections can be imaged at much higher resolution than tomographic images. A typical resolution for CAT or NMR scan is in the range of 1 mm, and 10 mm for PET scan. The physical sections described in this report were imaged at 40 microns. Since the space complexity scales as the cube of the spatial scale, physical sections can involve three to six orders of magnitude more data than a typical tomographic problem. The much greater resolution available in this type of preparation allows us to obtain unique views of the architecture of the brain. And, the gray-scale aspect of this data is of direct importance. While CAT or NMR scans usually represent abstract surfaces, brain sections represent images embedded in surfaces. Since much of the interesting structure

supported by System Development Foundation, AFOSR 85-0341, and the Nathan S. Kline Psychiatric Research Center

†The number of cortical layers depends on the species, area of cortex, and the precise definition of the term "layer".

(columnar and topographic) of visual cortex is beyond the reach of current CAT or PET technology, studying physical sections of monkey brain provides the only route to imaging the important functional features of the brain.

SOME BASIC TERMS

In order to motivate the algorithms discussed here, we will briefly summarize some of the important techniques in use. CAT scans are tomographic reconstruction of X-ray density estimates. Since this technique only reveals tissue density, it has little utility for imaging functional architecture. PET scan is a tomographic technique which has the advantage of being able to image active metabolic activity in the brain. One example of the use of this technique is to inject a glucose analogue called 2-deoxyglucose, which is labeled with positron emitting isotopes (e.g. Flourine-18). The 2-deoxyglucose(2DG) is only partially metabolized, and is briefly bound by neural cells. Thus, active cells take up more 2DG. By exploiting the properties of the positron decay (co-linearity of emitted photons), it is possible to create a three dimensional tomographic image of the metabolic activity of the brain (e.g. [13]).

Unfortunately, the advantages of PET in providing an active metabolic image are undermined by the poor spatial resolution of the technique. Current PET scanners operate roughly in the 10 mm range of spatial resolution. This is far too coarse to provide a detailed image of the structure of a cortical area. However, in monkeys, it is possible to use the same 2DG technique, followed by physical sectioning of the brain. This can provide both an active metabolic image, as well as sufficiently high resolution (~100 microns) to allow a detailed study of the brain to be performed.

The algorithms described in the present paper were developed largely for the application of 2DG experiments. However, in the present paper we will illustrate these algorithms with yet another technique. Cytochrome oxidase(CO) is an enzyme which is produced in greater amounts when the long term (~days) activity of neural cells is increased. If the brain of a monkey with only one active eye is processed to reveal the amount of this metabolic enzyme, then an image of the terminations of the active eye is produced in the series of sections. This creates a well known pattern, termed ocular dominance columns, which resemble the stripes of a zebra. Figures 1 and 2 show computer reconstructions of this pattern.

The ocular dominance column pattern is ideal for an assay of voxel based reconstruction methods because the scale of these columns is about 500 microns. This is small enough to provide a challenging test of our technology (when several thousand square mm of cortex are to be handled), yet large enough to be within practical reach. Moreover, much of the interesting functional architecture of an area such as visual cortex can be revealed at this scale.

Having briefly reviewed the problem domain of cortical architecture, and some of the major techniques available in this domain, a brief review of the algorithmic steps

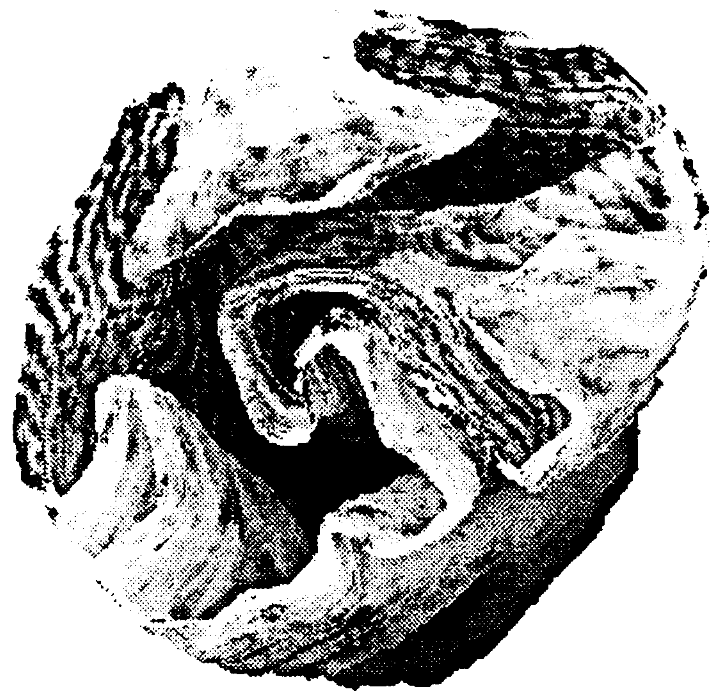


Figure 1.

This figure shows a surface composed of voxels. It is a single-voxel layer of the back half (occipital pole) of a monkey brain, in which a zebra-skin like pattern of metabolic activity appears. This pattern (ocular dominance columns) represents the terminations of the left (dark) and right (light) eyes in the brain. This structure was produced by repeatedly surface-tracking the entire brain. Each "surface" is saved, and the process repeated, resulting in some 25 voxel surfaces such as this one, which is located in the center of the cortex (layer IV). The size of these voxels is 40 μ x 40 μ x 40 μ, and the figure is composed of 414,211 voxels. Note that the voxel surface is independent of the gray-scale detail, which we have shown here merely for graphic interest.

necessary to form a full voxel based reconstruction of cortex, from serial sections, will be provided. Then, a detailed statement of two algorithms which have emerged form this work, the BRAIN_PEELER, and TRIANGULATION_FROM_VOXEL_SURFACES will be presented.

OVERVIEW OF THREE DIMENSIONAL RECONSTRUCTION FROM SERIAL SECTIONS

An overview of the process of three dimensional reconstruction and processing of serial brain sections, developed in our lab, has recently been published [15]. Briefly,

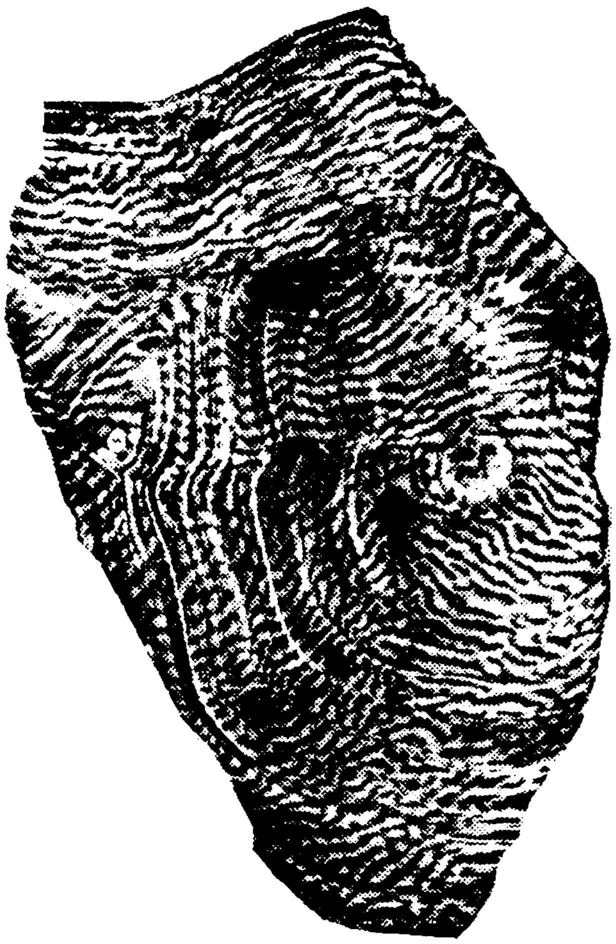


Figure 2.

This figure shows the flattening of the 3D polyhedral model of figure 1. A small section of the most peripheral extent of V1 in this data (about 1-2 mm) was lost, and is not shown.

there are several distinct stages of this work.

1. Data collection

The 2DG and cytochrome oxidase methods are outlined above; there are many other classical, histochemical, and radio-label methods available. For the present purposes, we will use a cytochrome oxidase stain of the pattern of connections of one eye of a macaque monkey, as reconstructed in primary visual cortex (see figure 1 and 2).

2. Digitization, image processing and alignment

Roughly two to four hundred serial sections, cut at 40 microns, provide a solids model ot a brain. In order to use these sections, they must be digitized (we use a Fairchild CCD camera), the sections must be image processed (contrast enhancement, thresholding, noise reduction, etc.). Finally, the sections must be aligned to their original three dimensional positions. This alignment step is extremely difficult, as well as critical. We use an interactive method in which an animation of the original solid, produced during brain slicing, is "toggled" with the corresponding processed section. This method provides an estimated three dimensional error of roughly 150 microns (i.e. the relative positions of voxels in the digitized model is within 150 microns, on average, of its original position, over the full posterior half of cortex, which is roughly 3000 square mm) This degree of precision is sufficient to accurately reconstruct the 500 micron ocular dominance column pattern, as shown in figures 1 and 2.

3. Brain peeling

The implicit solids model of the brain provided by a stack of image frames does not allow us to readily view the internal lamina of the cortex. In order to view the individual cortical lamina, it is necessary to "peel" the brain model. The details of the brain peeling algorithm will be presented shortly.

4. Triangulation

An aligned set of thresholded image frames is an implicit voxel model of the three dimensional brain. As such, it can be displayed via voxel rendering methods (as in figure 1). However, for some purposes it is necessary to obtain a polyhedral model of the same object. This requirement may be viewed as a problem in converting a voxel data structure to a polyhedral data structure, for the same 3D surface. We have developed an algorithm to perform this transformation, which will be described below. To the best of our knowledge, this is the first fully automatic method for triangulation of surfaces, and provides a means of using either a voxel or a polyhedral representation of the same surface.

5. Brain flattening

Brain flattening is one of the purposes for which a polyhedral model is required. Numerical algorithms for calculating minimal distances in polyhedral surfaces [18], and for using these minimal distances to flatten the brain [14] have recently been described. The interest of these methods is that the structure of cortex is most naturally displayed in an unfolded format, as in figure 2. In order to do this, both a voxel and polyhedral model of the cortical surface is necessary.

6. Texture mapping

Having digitized, aligned, peeled, triangulated and flattened a brain, the final step is to texture map the gray scale data in the 3D brain peel (figure 1) into its two dimensional (flattened) model (figure 2).

Having outlined our motivations, methods, and goals, we will know described in detail two algorithmic aspects of this work: brain peeling, and the use of voxel surfaces to effect an automatic triangulation of a surface.

TWO ALGORITHMS FOR BRAIN PEELING

ALGORITHM 1: INTERSECTION OF A POLYHEDRAL MODEL AND A 3D VOXEL MODEL

This algorithm is relatively trivial. Clearly, if one has access to a polyhedral model of some interior level of a solid, this polyhedron can be intersected with the 3D voxel model to obtain a voxel surface. The polyhedral model is likely to have been produced by hand tracing of contours on individual images of brain sections, followed by conventional (interactive) 3D triangulation. One advantage of this method is that it can produce a surface at a known (i.e physiological location), subject only to the accuracy and knowledge of the hand tracing of contours. However, this method will only reliably produce a voxel surface within a few voxel thicknesses of the traced surface. As one moves further from this canonical surface, the "peels" produced this way become unreliable, since they are following a relatively coarse approximation to the voxel surfaces. Figure 2 was produced with this method. However, if one wishes to produce a full set of peels (e.g. 25 peels at 40 microns thickness, spanning the full thickness of cortex), this method is inadequate. It would require a great deal of laborious hand tracing and hand triangulation.

ALGORITHM 2: 3D SURFACE TRACKING WITH A SHIELD

Excellent algorithms for 3D surface tracking are known (e.g. [2]). In the original implementation of our brain peeler, we used this algorithm as the basic surface tracking mechanism. However, there are several problems. First, the Artzy et al algorithm is memory intensive: it is necessary to store the full surface graph in active memory, and our high resolution surfaces can exhaust 10's of megabytes of memory. Below, we describe a surface tracking algorithm which requires memory storage of only a slice and its two neighbors†. Secondly, surface tracking provides two surfaces: the inside and the outside of the 3D voxel model. This is problematic, because we wish to consider the inside and the outside of the voxel model as distinct surfaces, which is the case physiologically. It is thus necessary to construct a "shield" which will prevent a surface tracker from producing both the inside and outside surfaces of the solid shell as a single connected component. Thirdly, successive peeling can introduce topological problems when applied to complex surfaces. A given surface may be perforated, thus changing the connectivity of the residual surface. Small connected components (which we call "dirt") may have escaped earlier imaging editing, or may be produced in the process of peeling.

†In principle, only a slice and its neighbor are necessary, but for simplicity of implementation, we store both neighbors.

These de[illegible]s must be carefully dealt with, as the process of peeling is subject to extreme topo[illegible]cal instability: even a single voxel connecting two surfaces inappropriately can change the qualitative nature of the entire peel being produced.

We use gray run-length encoded images (GRLE) [9] as our initial data representation, instead of voxels. In this format, we reserve gray-scale '0' for the background (i.e. '0' voxels are not legal in the object). The image is then represented by a run length code which describes, on each raster line the location of the starting 'x' position of the run, the length of the run, and then the full gray-scale voxel run. A modified GRLE format is also used (binary run length encoding) in which the actual gray intensities are dropped, since only the positions of the runs are needed (we call this format SRLE for stripped GRLE). If we measure the space requirement of a surface tracking algorithm in voxels and n represents the width of an arbitrary dimension in voxels, a surface tracking algorithm which requires the entire surface to be represented in memory requires $O(n^3)$ space whereas our algorithm results in a reduction to $O(n^2)$. This is true since the depth dimension is constant (three slices). Even though the space required by the width dimension is theoretically $O(n)$, the compression achieved by using GRLE typically reduces it to a small integer representing the maximum number of GRLE runs per scanline (~5 for our data). The use of GRLE format also increases the speed of our algorithm since not every voxel is addressed individually. Instead, GRLE runs are compared to each other to find the surface and to remove the peel from input data. Using SRLE runs as output also reduces the output size accordingly. To reconstruct full gray image peels from the SRLE output, we run a merge program to incorporate the gray intensities into the peels, thereby producing GRLE format data once again.

Algorithm Outline

The major concepts upon which the peeler algorithm is based are:

1) Capping the brain
2) Classification of regions
3) Generation of the shield
4) Preparation of the base
5) Generation of a peel
6) Removing a peel from the base

1. Capping the brain

This is a simple step which guarantees closure of the brain. We often wish to work with only a partial brain, meaning that the interior region will be exposed. To create a closed interior region, caps are added to either end of the set of sections with which we are currently working. A cap is simply an image completely filled with GRLE runs.

2. Classification of regions

In order to find the correct surface for peeling, all surfaces in the data must be classified. We define the surface we wish to peel as the one bordering on the largest interior hole in the brain. Other surfaces which we will encounter are: the exterior surface of the brain, the surfaces of small interior holes not connected to the main interior surface, and the surfaces of regions of data which are not connected to the true brain connected component (dirt). A sample input section to the peeler is shown in figure 3.

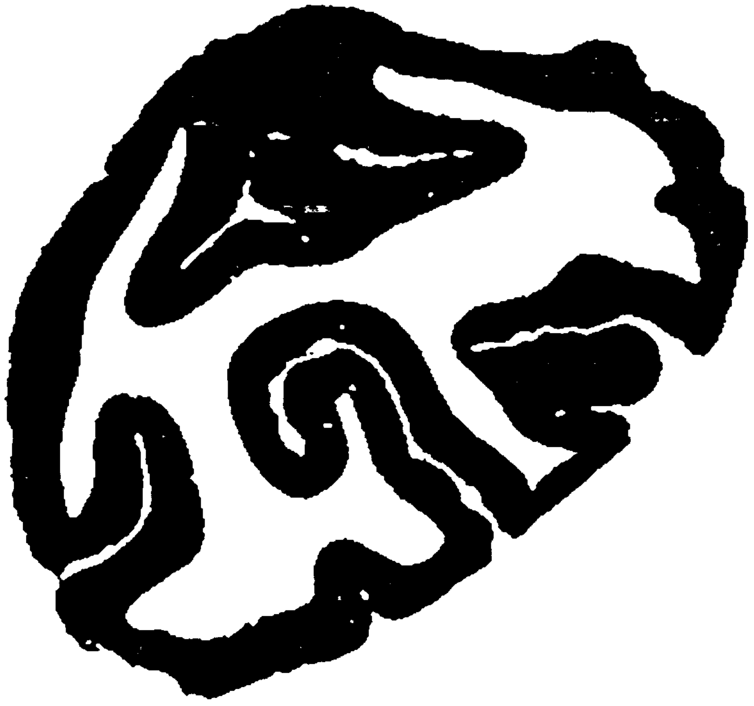


Figure 3. An input serial section.

These small interior holes and dirt, which for the peeler are topological errors, would in most cases be eliminated by the data editing processes [10] [12]. However, since the former is a human mediated procedure and the latter is not guaranteed to eliminate all topological errors, we must detect and handle any remaining topological errors to guarantee perfect peels for the triangulator. Alternatively, the editing performed here could be done in a separate program; but it seems a local requirement of the peeler, so it is done here.

Originally, we thought that we could pass through all the sections, to find each of the connected surface regions for each section. At the same time we would note how these connected surface regions were connected to each other from section to section. Then, the largest interior surface would be chosen as the true interior of the brain. This does not work for the following reason. If there exist thin walls in the brain, independent surfaces may be classified as connected to each other. For simplicity, we will use voxels in the following illustration. Imagine, for example, an interior surface separated from the exterior surface by a one or two voxel thick wall. All of the exterior surface voxels are chosen as surface elements as well as all of the given interior surface voxels. This means that the entire thin wall separating inside from outside will be entirely selected as part of the surface. Since there is now no way to distinguish the given interior region from the exterior, they will be considered part of the same surface. The same phenomenon can incorrectly connect distinct interior regions to each other.

To assure that disjoint surfaces are classified properly information about each face of each voxel could be kept, marking the proper surface faces instead of surface voxels. This is the approach taken by Artzy et al [2]. But keeping voxel face information is not necessary to correctly distinguish among distinct surfaces! To solve the problem we take the following approach.

The classification procedure makes a pass over the entire set of brain section images, processing two images at a time in a circular FIFO buffer. The output of this procedure are two graphs describing the adjacency of connected components in successive sections, and a set of output files containing a special variety of GRLE information that, in addition to specifying the begin location and length of the run, also specifies a tag identifying the connected component to which the run belongs in that image. At each iteration the following steps are performed:

1) split the current image into holes and walls

2) track and mark connected components within the holes images and walls images

3) match these connected components with those in their respective previous images

4) store the connectivity information of the two sections' holes and walls in the connectivity graph

5) merge the holes and walls components of the current image back into a single special GRLE image and store on disk

After all the images are processed, the connectivity graph for the walls is traversed, starting at the first component of the first image, marking all those nodes in the graph which are connected to it. This process is repeated on the graph until no more unvisited nodes remain. Each pass through the graph marks a separate connected component of the data. We define the largest of these connected components to be the brain proper and save the tag identifying it.

We perform a similar traversal of the connectivity graph for the holes, marking nodes on each pass to identify separate holes. In this case we are interested in identifying two regions, the unbounded exterior region (the exterior "hole") and the largest interior hole which we identify to be the desired interior region for peeling.

It is through the separate processing of the hole regions, then, that makes it possible to distinguish among distinct surfaces, even those which are separated by thin walls.

3. Generation of the shield

The shield provides a means of protecting the brain from having perforations made in it during the peeling process. Multiple peels can be generated by the peeler. As the peeling commences, beginning from the inside surface, peels will eventually encounter the outer surface of the brain. Since the brain is not of uniform thickness, a peel which encounters the exterior will do so only at intervals along its length. When it does so, the region of the exterior which is encountered is peeled away. This creates a gap in the remaining data, and unless handled, subsequent peels would not be closed. The shield assures that encounters with the exterior are detected and patched before the next peel is generated.

To generate the shield, we make another pass through the image files - this time processing with three images simultaneously in a circular FIFO buffer. In addition to the three base images we have in memory, we read in the special image, generated in the classification step, corresponding to the most recently read image of the three base images currently in memory. The special image modifies this newest base image by filling in all of its interior holes (figure 4). Note that the other two base images in the buffer had been previously modified in the same manner.

Now the three modified images are operated upon by a version of the surface-tracking algorithm which detects all exposed surfaces. In this case, the only exposed surfaces are the exterior ones, so the result is the edge-adjacent exterior subset [2] of the center image (rendered as GRLE runs instead of voxels, see figure 5). The resulting peel image is stored on disk, the oldest image in the buffer is dropped, a new one is read in, and the process is iterated until all the base images are processed.

4. Preparation of the base

This procedure is done in concert with the generation of the shield, explained above; but is presented separately for clarity.

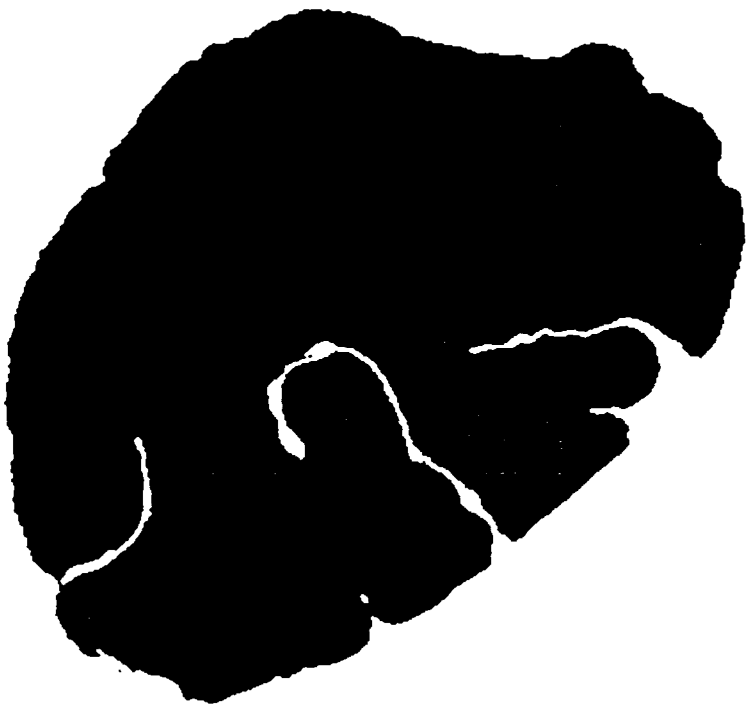


Figure 4. Interior holes filled.

The result of this procedure will be a set of modified base images upon which the surface tracker will operate. We wish to remove all dirt regions and fill all exterior regions, leaving only the desired interior region as an exposed surface to be detected by the surface tracker. The notion of filling the entire unbounded exterior is understood by recognizing that for each scanline of a brain image, the exterior is always encountered by a ray moving to the right from negative infinity and by a ray moving to the left from positive infinity. Therefore, by making a simple modification to the surface tracking algorithm, these cases can be detected without any filling operation. On the other hand, those exterior regions which are enclosed by data (bounded by GRLE runs) are not detectable by the same means and are filled explicitly. This filling not only enables a simple uniform procedure to be performed during later peeling, but also reduces further the amount of data to be handled, since each fill operation joins two GRLE runs into one.

Before the newest image in the buffer is operated upon by the special image, a copy of it is made which is also operated upon by the special image, but in a different way. This time, all dirt is removed from the base image and all exterior hole

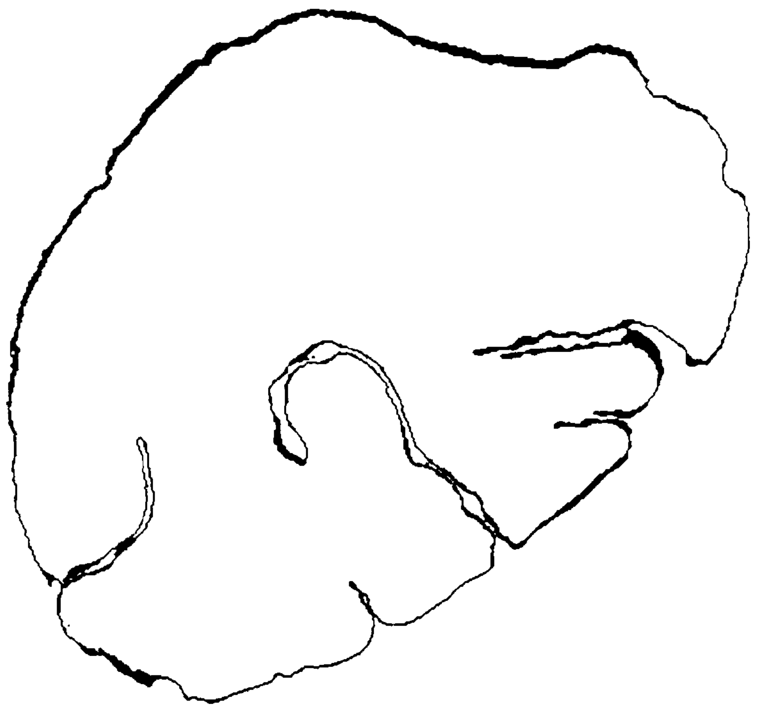


Figure 5. The shield.

regions are filled. The modified base image is stored on disk (figure 6).

5. Generation of a peel

The generation of a brain peel involves a pass over the prepared base sections using a three image circular FIFO, as in the shield generation. This time, the three images are simply operated upon by a version of the surface tracker which generates an image of all exposed surfaces excluding those which are encountered first and last on each scanline, since these are always exterior surface elements. The peel is stored on disk (figure 7).

6. Removing a peel from the base

Finally, the newly-generated peel must be removed from the base images. This is done in two steps. We consider removing the peel data from a single base image. First, the shield image corresponding to this base image is removed from the peel just generated for this base image. This does not change the peel unless it

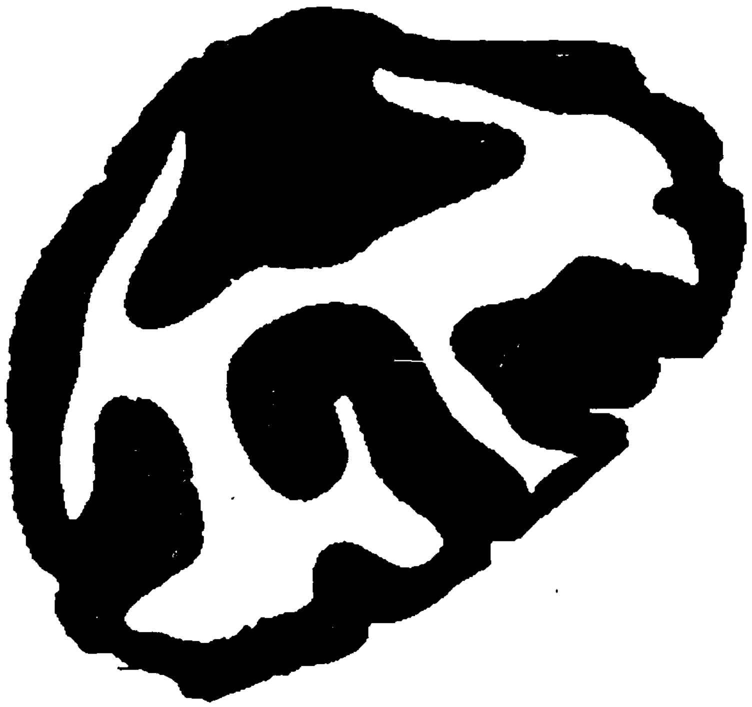


Figure 6. Exterior filled and dirt removed.

has overlapped the shield. Second, the remaining peel data is removed from the base image.

After the shield data coincident with the peel has been removed from the peel itself, the remaining peel can no longer create a gap in the base when it is removed from the base. This means that later successive peels will contain the same data where they coincide with the shield, but, for our purposes, a single connected surface is what is required.

7. Surface Tracker

The kernel of the surface tracker is the simple idea that a surface element is one which borders an empty region. A naive implementation using voxels would simply check each inhabited voxel's neighbors, selecting each inhabited voxel with at least one empty neighbor. In this way, all the surface elements for all surfaces would be selected.

Often, input data will contain many separate connected surfaces, but only certain surfaces (usually one) are the desired output. A traditional surface tracking algorithm,

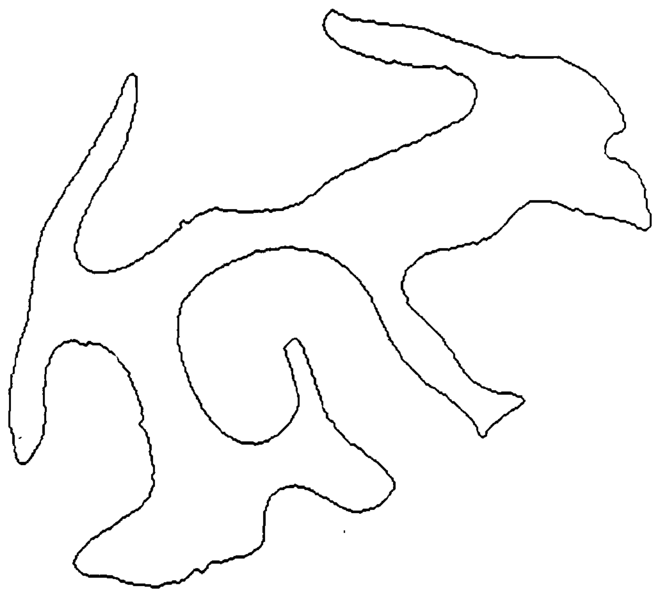


Figure 7. A peel section.

given a starting point of a given surface, moves from voxel to voxel locating all surface elements connected to the starting point and only those elements. Our tracker works in a different manner. Instead of crawling over the three-dimensional surface to find all its elements the data is pre-processed such that undesired surface elements are eliminated before the main surface tracking procedure. Then the surface tracker can simply move through the pre-processed data, selecting all surface elements which it finds, without regard to connectivity (since all surface elements are guaranteed to be desired ones).

GRLE format data is used to represent the sequence of brain images in memory. As mentioned before, we do not retain the actual gray values for each voxel in memory because they are not needed by the algorithm. What we do need are the (x,y) location and length of the run. We keep the runs in memory as an array of lists of runs, one list per scanline of image. Each run in memory has an x starting location, a length and a pointer to the next run. The y location is represented by the element of the array pointing to the list containing this run.

GRLE format data provides several advantages over voxels for surface tracking. First, they take up less space since each is able to represent many voxels. Second, since there is less data to handle, algorithms are faster. Third, their very definition implicitly encodes some surface data since their ends are always bordering .n empty region (no two runs may abut).

To translate the concept of determining whether a given voxel is bordering on an empty region into where a given GRLE borders on an empty region, imagine a given GRLE run and the surrounding (face-adjacent) runs. At any place along the length of the subject run where there is a gap in the surrounding runs, we have a surface element. The ends are always surface elements as well. The surface elements of a given subject GRLE run consist of either a set of disjoint regions or the entire GRLE run if it is completely exposed. To do so, we introduce the concept of the occlude list. As the occlude list is constructed, it represents regions of the subject GRLE which are potentially occluded from view (not exposed to an empty region). When the process of generating the occlude list is finished, any remaining elements in it represent truly occluded portions of the subject GRLE. The complement of this list is the set of disjoint surface elements, also GRLE runs.

The occlude list is constructed from the four lists of surrounding face-adjacent GRLE runs, two from the same image as the subject run (those above and below it), and one each from the previous and next images. This explains the necessary and sufficient condition that three images must be in memory at once to determine the surface elements of the runs which the middle image contains. The occlude list is initialized from one of these surrounding lists, of which up to two elements may be truncated to the subject GRLE if they overlap its ends. Then the occlude list is updated by each of the remaining three lists of surrounding runs, each one potentially removing some of the occlude list, possibly chopping it into a greater number of elements. An update consists of doing a logical "and" of the current occlude list and the new list. This reduces the occlude list to areas in which both the old occlude list and the new list cover the subject GRLE run. In this way, as each update is made to the occlude list, only those regions of the subject GRLE run which are occluded by all of the surrounding lists of GRLE runs remain in the occlude list.

Finally, the occlude list is applied to the subject GRLE run, chopping it up into disjoint surface GRLE runs. Note that the ends of the subject GRLE, if they are not already included in an exposed region as dictated by the occlude list, are included as well. Also note that if during the update of the occlude list, it is reduced to a null list, this means that the entire subject GRLE is exposed and it can be output as a single surface element immediately, without checking the remaining surrounding lists of runs.

Performance

Figure 1 was produced from 363 images (512x512x8) of coronal sections of a monkey brain (one functioning eye) stained for cytochrome oxidase. This data set was about 90 megabytes in voxel format and about 15 megabytes in GRLE format. On a SUN 3/50, the brain peeler required about 2.5 megabytes of memory. The pre-processing time was about 18 minutes, with each peel requiring a little less than 7 minutes to generate. About 20% of the peel generation cpu time was devoted strictly to I/O.

Future Work

The space requirement could be lowered still further, mainly by changing the surface tracker to operate on only two images at a time instead of three. Using this method the old image will preprocess the new one such that each GRLE run will be potentially partitioned into two types of runs: ones which are guaranteed to be exposed (because the old image did not occlude them), and those which may yet be occluded (because the old image covered them). When the new image in turn becomes the old one on the next iteration, the possibly occluded runs are compared to the new image and are truncated into exposed regions in the way described by this report. Finally, the previously generated exposed runs and the newly generated exposed runs are merged to produce the output SRLE runs for the current image.

THE CONNECTIVITY PROPERTIES OF VOXEL SURFACES: AUTOMATIC TRIANGULATION OF A VOXEL SURFACE

If a voxel surface produced by the brain peeler is represented by planar sections (of one voxel thickness), then the appearance of each of these sections is that of an 8-adjacent set or contour [2].

These contours resemble the contours which are produced by hand tracing through the original sections, but have one extremely important property: each voxel on one section has a neighboring voxel to which it is either edge or face adjacent on both neighboring sections. This is a trivial property of the fact that they are 2D sections of a 3D connected component. We call the contours produced in this way ''2.5D contours''. Just as a set of peels represents a voxel volume by its tangential surfaces, a (thinned) set of 2.5 contours represents a voxel surface by tangential curves. The set of 2.5D contours provide a means for constructing an automatic, minimal, topologically correct triangulation of the voxel surface, because they allow us to locate the critical points of the voxel surface, and to specify the connectivity rules to be applied to bridge these critical points.

We will outline the algorithm for this triangulation, but first will briefly describe the difficulty in triangulating serial contours of a 3D object.

Heuristic triangulation algorithms and topological requirements

There are many algorithms which are capable of triangulating generalized cylindrical surfaces [11] [8] [6] [5], and others which attempt to deal with more general three-dimensional structures [7] [4] [1] [19]. In our experience, none of these algorithms is sufficient to successfully model a complex surface, such as that of primate cortex, without human supervision. Part of this problem is that even for a simply connected (in

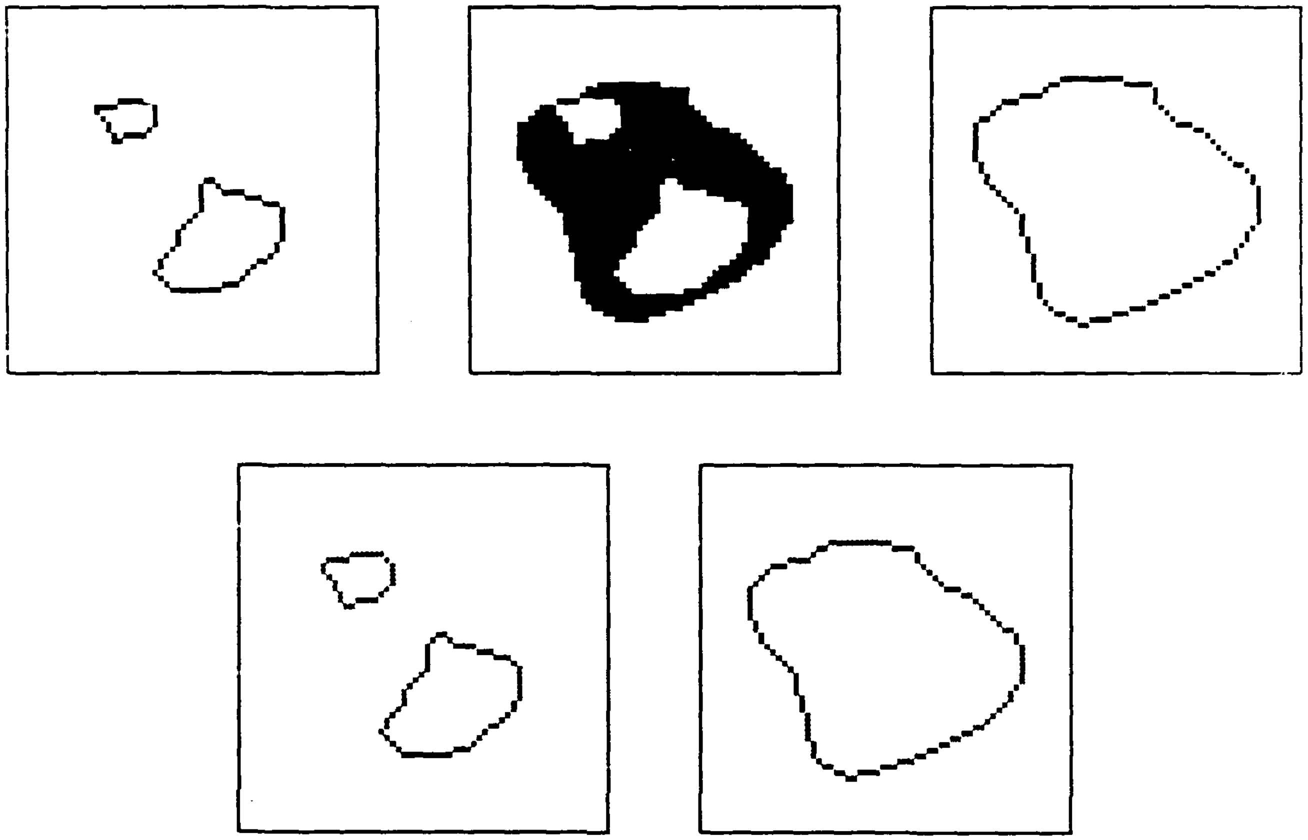


Figure 8.

top center: a contour of section 153.
top left and right: its neighbors on the previous and succeeding sections.
bottom: the *adjacency sets* in our contour of its neighbors. These are the sets of voxels in the middle contour that are actually edge-adjacent to voxels in the neighbor contours.

3D) surface, contours might merge, split, be "born", "die", or change their internal structure as one moves from section to section. Adjacent sections of a surface can manifest changes in topology or connectivity even if the surface is simply connected and closed. Such changes are purely an artifact of the sectional representation of a surface (see Figure 8). A major part of the solution to triangulation in this context is thus the proper association of contours on one level with those on adjacent levels. This association is supplied by the properties of sections of voxel surfaces.

The essential difficulty in triangulating a surface represented as serial sections thus arises from topological changes in the surface contours encountered during serial traversal of the sections. These changes can be localized at *critical points* of the surface. For the moment, assume that a voxel surface is a discrete sampling of a smooth surface (a differentiable manifold). A critical point occurs when the partial derivatives of the z

coordinat .r height function with respect to the local coordinate functions vanish [17]. Geometrically, this corresponds to a situation in which the tangent plane to the surface is "horizontal." This occurs at maxima, minima, and saddle points of the surface. A section containing a critical point will be denoted a *critical level*.

As a critical point is passed in the sectional traversal, a single contour can split into several disjoint contours, or several disjoint contours can merge into one. These topological changes at critical levels incapacitate existing algorithms for triangulating from serial sections, which become confused when a contour that is being tracked suddenly splits into multiple contours, or vice versa.

As an example of a sectioned differentiable manifold, consider the surface of a doughnut standing on edge, and the serial sections of this object by horizontal planes (Figure 9).

The following topological transformations occur. First, a point appears, the first critical point. We call this a "birth" event. At this level the section consists of a single point, which immediately opens into a loop as we ascend. A series of loops is produced until a second critical point of the doughnut is reached. At this point, the loop becomes a "figure eight", and splits into two disjoint loops. We call this event "splitting." Then, at the third critical point, the loops fuse back into a "figure eight". We call this a "merge" event. The figure eight immediately becomes a simple loop. Finally, the single loops shrink down to a point and disappear at the fourth and final critical point (a "death" event).

For the embedding of the torus implied by this sectioning, there are four isolated critical points. It can be proven that any differentiable 2-manifold can be embedded in Euclidan 3-space in such a way that its critical points are finite in number, and no more than one occurs on a given level. Also, it can be shown that the only topological changes which can occur (for an orientable surface) are the "birth" , "death", "splitting" and "merging" events illustrated for the torus [17].

Algorithm Outline

Having stated, by analogy of the critical points and critical levels of a differentiable manifold, the problems which a triangulation algorithm must solve, we now turn to a discrete approximation to the manifold, or surface, which we wish to triangulate.

Consider a series of voxel-thick contours obtained by intersecting the surface with parallel planes (see Figure 8). Contours on adjacent levels have adjacency relationships, which allow us to "parse" the surface into *generalized cylinders*, which a simple triangulation algorithm can process. (A generalized cylinder is a run of simple loops. It contains no critical points, and represents a (distorted) sausage-like object [3]). In our experience, even complex surfaces such as cortex consist of relatively few critical levels with long runs of generalized cylinder between them. We find the critical points where the four types of transitions — "birth", "death", "merging", or "splitting" — occur,

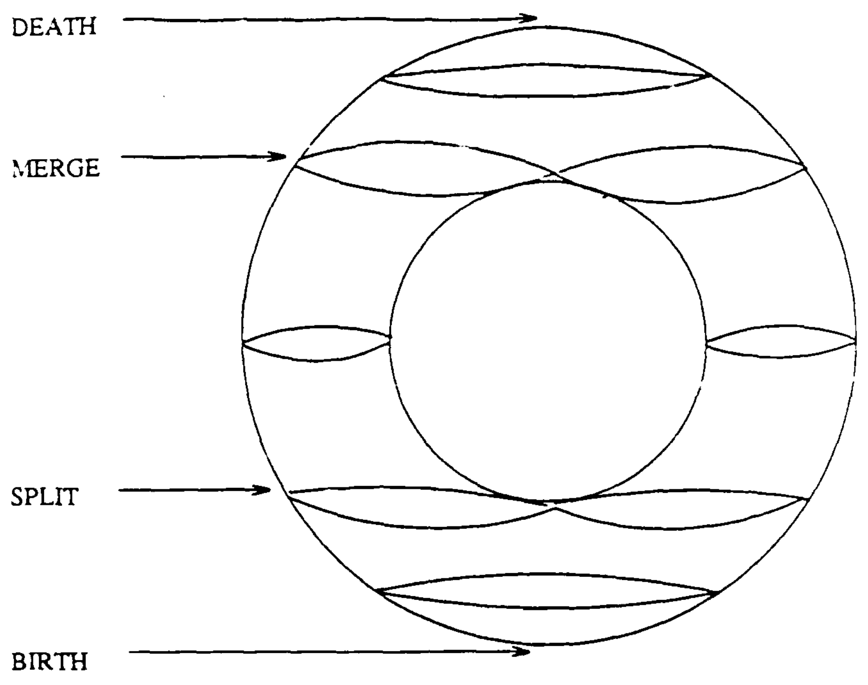


Figure 9.

This figure shows a torus, and the location of its four critical points: a maximum, two saddle points, and a minimum. Below the first saddle point, the sections of the torus are single loops. Above this critical section, whose contour resembles a "figure-eight", the contours split into two sets of loops/section. The torus can be represented in terms of its critical points, and the generalized cylinders which lie between critical levels. In general position, any differentiable manifold has a finite, separated set of critical points, so that the situation illustrated in this figure is generic for the kinds of surface which we wish to triangulate.

and write a program or script for the simple triangulator. This script contains the commands a human would normally be required to provide to instruct the simple triangulator on the proper association of contours around the critical levels. We use the Mosaic program of Movie.BYU for the simple triangulator stage of the process. (The "compiler approach" which we follow makes it easy to substitute other simple triangulators. In effect, we construct a high-level syntactic description of the surface consisting of critical points and generalized cylinders, and then write a program in a low-level language (such as a script for Mosaic) to effect the triangulation.

Figure 10 shows an example of the triangulation of a monkey brain from the voxel surface of figure 1.

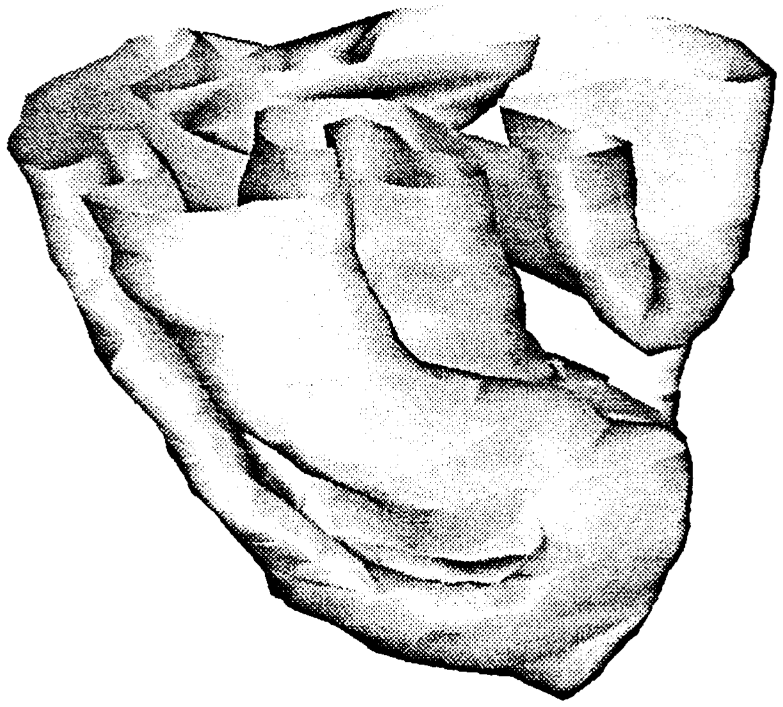


Figure 10.

This figure shows a polyhedral model of the same brain represented in figure 1, in terms of triangular surface patches, of typical size about .5 mm^2. This triangulation is the final output of the algorithm of the present paper, that is, this triangulation was produced completely automatically, with no human interaction or decision. There are 1776 triangles in this model.

This triangulation, consisting of about 2500 triangles, was accomplished purely automatically, using the above described algorithm. To the best of our knowledge, this algorithm is the only algorithm currently known which is capable of triangulating a surface as complex as that of the monkey brain, with no human interaction. It is based entirely on the connectivity properties of the voxel surfaces produced by the brain peeler. A full description of the implementation details of this algorithm can be found in [16].

SUMMARY

Given a solid shell, described by voxels, we discuss several methods of obtaining the tangential voxel surfaces whose union is this shell. We call this operation ''peeling'', and illustrate its application with data of monkey visual cortex. Secondly, we show that a representation of a voxel surface, consisting of the tangential curves whose union is the surface, has connectivity properties which allow us to perform a conversion into polyhedral representation of the surface. This algorithm is based on the observation that the sole difficulty in triangulating a surface is caused by the existence of critical points, and these critical points can be easily located and bridged by using the connectivity properties of the voxel representation of the surface. We demonstrate this triangulation algorithm on the same monkey data, and show that it is capable of constructing a polyhedral representation from the voxel representation of a highly complex surface.

References

1. Anjyo, K., T. Ochi, Y. Usami and Y. Kawashima, A practical method of constructing surfaces in three-dimensional digitized space, *The Visual Computer 3 no. 1*, (1987), 4-12.

2. Artzy, E., G. Frieder and G. T. Herman, The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm, *Computer Graphics and Image Processing 15*, (1981), 1-24.

3. Binford, T. O., Visual Perception by Computers, *Proc. IEEE Conf. Systems and Control*, Miami, Fla., 1971.

4. Boissonnat, J. D., Geometric structures for three-dimensional shape representation, *ACM Trans. on Graphics 3 No. 4*, (1984), 266-286.

5. Cavendish, J. C., D. A. Field and W. H. Frey, An approach to automatic three dimensional finite element mesh generation, *General Motors Tech. Rep. GMR-4533*, (1983), .

6. Christiansen, H. and M. Stephenson, *MOVIE.BYU*, University Press, Brigham Young University, 1983.

7. Faugeras, O. D., M. Hebert, D. Mussi and J. D. Boissonnat, Polyhedral approximation of 3-D objects without holes, *Computer Vision, Graphics, and Image Processing 25*, (1984), 169-183.

8. Fuchs, H., Z. M. Kedem and S. P. Uselton, Optimal Surface Reconstruction from Planar Contours, *Communications of the ACM 20*, (1977), 693-702.

9. Kaplow, W. K. and E. L. Schwartz, Data formatting and software tools for gray-scale run-length encoding of image data, *Comp. Neuro. Tech. Rep. CNS-Tech. Rep.-21-86*, (1986), , NYU Med. Ctr./Computational Neuroscience Laboratories.

10. Kaplow, W. K. and E. L. Schwartz, An interactive system for alignment and manipulation of images of brain sections: BITBOX, *Comp. Neuro. Tech. Rep. CNS-Tech. Rep.-20-86*, (1986), , NYU Med. Ctr./Computational Neuroscience Laboratories.

11. Keppel, E., Approximating complex surfaces by triangulation of contour lines, *IBM J. Res. Devel. January*, (1975), 2-11, IBM.

12. Landau, P. and E. L. Schwartz, A threc dimensional median filter: applications to repair of damaged brain sections and to interpolation of serial sections, *Comp. Neuro. Sci. Tech. Reports CNS-Tech. Rep.-1-87*, (1987), .

13. Schwartz, E. L., D. R. Christman and A. P. Wolf, Human primary visual cortex topography imaged via positron-emission tomography, *Brain Research 104*, (1983), .

14. Schwartz, E. L., A. Shaw and E. Wolfson, A numerical solution to the generalized mapmaker's problem, *IEEE Trans. Pattern Analysis and Machine Intelligence*, , in

press,1988b.

15. Schwartz, E. L., B. Merker, E. Wolfson and A. Shaw, Computational neuroscience: Applications of computer graphics and image processing to two and three dimensional modeling of the functional architecture of visual cortex, *IEEE Computer Graphics and Applications*, , (In press) 1988c.

16. Shaw, A. and E. L. Schwartz, Automatic construction of polyhedral surfaces from voxel representations, *Computer vision, graphics and image processing (submitted May 1988)*, (1988), .

17. Wallace, A. H., in *Differential Topology*, W. A. Benjamin, New York, 1968.

18. Wolfson, E. and E. L. Schwartz, Computing minimal distances on arbitrary polyhedral surfaces, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , in press,1988.

19. Zyda, M. J., A. R. Jones and P. G. Hogan, Surface construction from plane contours, *Comp. & Graphics 11*, (1987), 393-408.