BIOLOGICALLY CONSISTENT GRID GENERATION FOR BRAIN VISUALIZATION

A Thesis

by

DAVID ALLAN BATTE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE

August 1996

Major Subject: Computer Science

BIOLOGICALLY CONSISTENT GRID GENERATION FOR BRAIN VISUALIZATION

A Thesis

by

DAVID ALLAN BATTE

Approved as to style and content by:

Bruce H. McCormick (Chair of Committee)	John J. Leggett (Member)
Donald H. House (Member)	Richard A. Voltz (Head of Department)

August 1996

ABSTRACT

Biologically Consistent Grid Generation for Brain Visualization. (August 1996)

David Allan Batte, B.S., Southwest Texas State University

Chair of Advisory Committee: Dr. Bruce H. McCormick

Numerical grid generation is used to provide a framework for brain and neuron visualization. Smoothing spline surfaces are fit to contour data to generate 3D solid model reconstruction of brain tissues. Finite element methods are then used to subdivide the solid models into biologically-consistent finite elements. Numerical grid generation is employed to provide a curvilinear coordinate system within the finite elements. Synthetic and manually traced neurons are mapped into the gridded solid model using the curvilinear coordinate system. Grid generation tools, neuron mapping tools, and visualization tools have been implemented.

To my family

ACKNOWLEDGMENTS

I would like to thank Dr. Bruce H. McCormick for his invaluable support and guidance. I also thank my colleagues Kishore Mulchandani and Sandeep Tewari for their suggestions. A special thanks to my friends Allan Wehrman, Sarma Vanguri, and Terry Larson for being there when moral support was needed. Finally, I thank my parents for making me what I am today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Neurons are densely packed into neural tissue and this presents a barrier to the visualization and interpretation of globally-stained neural tissue scanned at the limit of optical resolution. Advances in confocal microscopy and fluorescent staining of individual neurons allow the tracking of their fibers and neural processes in neural tissue. The interpretation and modeling of these neuron data sets rests on having a biologically-consistent framework in which the neuron populations can be visualized. For brain visualization at the cellular and tissue level, a coordinate system within a brain nucleus or cortical area must be generated to produce a biologically-consistent model of the volume filled with a sparse representative population of neurons. Numerical grid generation [22, 11, 25] is a well-established method for producing boundary-conforming curvilinear coordinate systems in irregular 2D and 3D regions. Its development arose from the need to solve partial differential equations within physical regions with complex geometry. Due to the complex, folded nature of the cerebral cortex we use numerical grid generation to embed a 3D coordinate system inside cortical tissue. Neurons, either individually traced or synthetically generated, are then embedded inside the gridded solid model of the cortical tissue and visualized in this environment. More generally, similar considerations apply to brain nuclei.

Journal model is *IEEE Transactions on Visualization and Computer Graphics.*

A. Objectives

A.1 Biologically-Consistent Finite Element Decomposition of Brain Tissue

Our primary objective is to embed biologically-consistent grids within solid model reconstruction of cortical areas and brain nuclei. To establish grids in a manner consistent with developmental neurobiology, the following constraints should be held:

1. Cell counts between logical space and physical space should be preserved, and
2. Grid curves normal to the solid boundaries should follow glial trellises where these are known.
3. Grid curves parallel to the solid boundaries should follow the natural "fuzzy" layers found in brain tissues.

To achieve grids which conform to these constriaints, the solid model is decomposed into blocks using finite element methods. The finite elements are constructed such that they conform to these constraints. The blocks serve as the boundaries which are used by the numerical grid generation techniques. The grids will establish a curvilinear coordinate system within the solid model. This coordinate system will provide a means of specifying physical barriers and chemical gradients within the volume.

A.2 Biologically-Consistent Mapping of Neurons into the Gridded Solid Model

The second objective is to embed the gridded solid model with a population of neurons. Sparse populations of neurons can be from databases of measured neurons,

or stochastically generated using L-system modeling as described by McCormick and Mulchandani [14, 16, 17]. The grids provide a means of:

1. positioning the neurons in the solid model reconstructed tissue, and

2. specifying positional forces within the volume.

The embedded neurons should be statistically consistent with those found in the actual tissue.

A.3 Visualization of Neuron Populations within the Gridded Solid Model

The third objective is to visualize a population of neurons within reconstructed tissue using computer graphics along with the grid generation and neuron mapping methods. An interactive approach, developed below, allows the user to freely explore the environment.

A.4 Provide Spatial Normalization of Brain Tissue

Neurons are formed into locally-connected networks in cortical areas and brain nuclei. The idiosyncratic configuration of these environments must be spatially normalized to allow:

1. a comparison of neural tissue independent of the spatial environment conditioning growth, and

2. comparison of population-based neuron morphology across specimens.

B. Significance

The techniques presented in this thesis will serve as a basis for research on how neuronal structure reflects the cortical environment in which it develops and in turn how the incoming afferents of these neurons structure the local anatomy of the cortex. This future research will address the folowing neuroscience issues:

- What constitutes normal and abnormal neuron development in the neocortex?
- How do pyramidal and stellate cells scale with their position within the cortex?
- How do neurons accomodate to convolutions in the neocortex?
- Why does the cerebral cortex fissure and fold?

CHAPTER II

3D SURFACE RECONSTRUCTION

A. Introduction

The first step in the grid generation process is to generate a solid model representation of the tissue being modeled. This solid model can be generated by reconstructing the surfaces of the tissue from cross sectional data. Cross sectional data can be obtained from two sources: digital imaging (Computer Tomography-scans, Magnetic Resonance Imaging, and nuclear medicine systems), and optically or physically-sectioned tissue.

Once cross sectional data has been obtained, defining contours must be placed on each image around the areas of interest which are to be reconstructed. The contours are then aligned and the surfaces bounding the solid model are generated from the contour information.

B. Contouring

There are two approaches to generating contours on an image: manual and automatic methods. For manual contouring, spline curves are placed and manipulated by hand. The advantage of this method is that the user has total control over the contours that are generated. The disadvantage is that manual contouring is very time consuming and tedious. Automatic countouring methods, on the other hand are fast although direct control over the curves is lost. An example of an automatic

contouring method is Active Contours, or "Snakes". A snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it toward features such as lines and edges [10].

C. Surface Reconstruction

Since its beginnings in the 1970's there has been a great deal of research into methods for reconstructing surfaces from image contours [23]. The two primary approaches to this problem are polygonal surface fitting and parametric surface fitting. Since the grid generation methods which are presented in this thesis rely on boundng surfaces which are described parametrically, only this latter method will be discussed here.

In the parametric surface fitting approach, the contours are first sampled. Interpolating or approximating spline functions are then fit to the samples. Most work in this area has focused on representing the reconstructed surface as an embedding of a 2D parameter domain. Typically the parameter domains are topologically simple, such as a plane [9, 1], cylinder [19, 20, 8], or sphere [2]. The method of surface fitting used in this thesis uses a parametric domain of a rectangular mesh of data points.

D. Surface Fitting to a Rectangular Mesh

An efficient method of finding a smooth spline approximation s to a function R, where the sample data points $R_{q,r}$ are given on a rectangular mesh is described in *Curve and Surface Fitting using Splines* by Paul Dierckx [5].

D.1 Tensor Product Splines

Given a strictly increasing knot sequence

$$a = \lambda_0 < \lambda_1 < \ldots < \lambda_g < \lambda_{g+1} = b,$$

$$c = \mu_0 < \mu_1 < \ldots < \mu_g < \mu_{g+1} = d,$$

then a function $s(u, v)$ is a spline on a rectangle, $D = [a, b] \times [c, d]$, of degree k in u and degree l in v with knots $\lambda_i, i = 1, 2, \ldots, g$ in the u direction, and $\mu_j, j = 1, 2, \ldots, h$ in the v direction.

If k and l additional arbitrary knots are added to the edges so that the knot sequence is defined as $\lambda_i, i = -k, -k+1, \ldots, g+k, g+k+1$, and $\mu_j, j = -l, -l+1, \ldots, h+l, h+l+1$, then every spline on D can be expressed as

$$s(u, v) = \sum_{i=-k}^{g} \sum_{j=-l}^{h} c_{i,j} M_{i,k+1}(u) N_{j,l+1}(v).$$

where

$$M_{i,k+1}(u) = (\lambda_{i+k+1} - \lambda_i) \Delta_t^{k+1}(\lambda_i, \ldots, \lambda_{i+k+1}) g_k(t; u),$$

$$N_{j,l+1}(v) = (\mu_{j+l+1} - \mu_j) \Delta_t^{l+1}(\mu_j, \ldots, \mu_{j+l+1}) g_l(t; v),$$

are B-spline basis functions, with

$$g_m(t : x) = (t - x)_+^m = \begin{cases} (t-x)^m & \text{if } t \geq x, \\ 0 & \text{if } t < x, \end{cases}$$

and $\Delta_t^m(z_i, z_{i+1}, \ldots, z_{i+m}) f(t)$ is the mth divided difference (see de Boor [4]) of the function $f(t)$ on the points $z_i, z_{i+1}, \ldots, z_{i+m} f(t)$.

D.2 Surface Fitting with Tensor Product Splines

Given the sampled data values $R_{q,r}$ at the grid points $(u_q, v_r), q = 1, 2, \ldots, M_1; r = 1, 2, \ldots, M_2$ of a rectilinear mesh, a spline approximation $s(u, v)$ which fits the data values closely, but smoothly, needs to be found.

A smoothing spline $s_p(u, v)$ is defined for all positive values of a parameter p. Its B-spline coefficients $c_{i,j}$ are the least-squares solution of the following system of equations:

$$\sum_{i=-k}^{g} \sum_{j=-l}^{h} M_{i,k+1}(u_q) N_{j,l+1}(v_r) c_{i,j} = R_{q,r}, q = 1, 2, \ldots, m_1, r = 1, 2, \ldots, m_2,$$
$$\frac{1}{\sqrt{p}} \sum_{i=-k}^{g} \sum j = -l^h M_{i,k+1}(u_q) b_{j,r} c_{i,j} = 0, q = 1, 2, \ldots, m_1, r = 1, 2, \ldots, h,$$
$$\frac{1}{\sqrt{p}} \sum_{i=-k}^{g} \sum j = -l^h N_{j,l+1}(v_r) a_{i,q} c_{i,j} = 0, q = 1, 2, \ldots, g, r = 1, 2, \ldots, m_2,$$
$$\frac{1}{p} \sum_{i=-k}^{g} \sum j = -l^h a_{i,q} b_{j,r} c_{i,j} = 0, q = 1, 2, \ldots, g, r = 1, 2, \ldots, h.$$

The goal is then to find the value of p for which $F(p) = S$ where

$$F(p) = \sum_{q=1}^{m_1} \sum_{r=1}^{m_2} (R_{q,r} - s_p(u_p, v_r))^2$$

and S is a nonnegative parameter which is called "the smoothing factor". This parameter is supplied by the user, and governs the tradeoff between closeness of fit and smoothness of fit. Some care needs to be taken when choosing a value for S. If

S is too large, $s_p(u,v)$ will badly approximate the data. If S is too small, the spline will pick up too much noise and will be unacceptably wiggly.

To find a spline which satisfies $F(p) = S$, a set of knots must be found for the spline. A simple and efficient iterative method is used to find a suitable set of knots. First, the least-squares polynomial $P_{k,l}(u,v)$ is determined. It is simply the least-squares spline $S_{0,0}(u,v)$ without interior knots. If this is an acceptable solution, nothing furthur needs to be done. If it is not acceptable, an iterative process of adding interior knots and checking if the resulting spline is acceptable is used. On each iteration, the point in which the fit of the current spline, $S_{g,h}(u,v)$, is particularly poor is determined and a new knot is inserted at that point. This scheme is adaptive in the sense that more knots are placed where the data is more difficult to approximate.

E. Reconstruction of Surfaces of Cortical Areas

Reconstruction of cerebral cortex consists of reconstructing the outer surface as well as the inner surface of the cortex. When the reconstruction procedure described above is performed on the two surfaces of the cortex, parametric curvilinear coordinates, u and v, are imposed on the surfaces. These coordinates will be used for the finite element decomposition of the cortex.

CHAPTER III

BIOLOGICALLY-CONSISTENT FINITE ELEMENT DECOMPOSITION

A. Introduction

In this chapter a method of partitioning a solid model into finite elements is proposed. Each finite element has four vertices on both the outer and inner surfaces and defines a six-sided block of reconstructed tissue. To generate grids which conform to the biological constraints as outlined in the introduction, the finite elements are chosen to follow the natural symmetries of the tissue, as defined by its primary native neuron type and thinking how a neuroanatomist would cut tissue locally to make successive sections look as alike as possible.

B. Cortical Finite Element Decomposition

The primary neuron type of the cerebral cortex is the pyramidal cell. The pyramidal cells are arranged in a columner organization which radiates from the inner surface to the outer surface. This can be seen by slicing and staining the tissue perpendicular to the major direction of the sulci and gyri (Figure 1). We use this information to generate the u, v, and w isocurves of the finite element mesh for the cortex. The u-isocurves follow the crest and valley lines of the cortical solid model. The v-isocurves are perpendicular to the u-isocurves and are on the two surfaces. The w-isocurves run from the outer surface to the inner surface.

The basic algorithm is as follows:

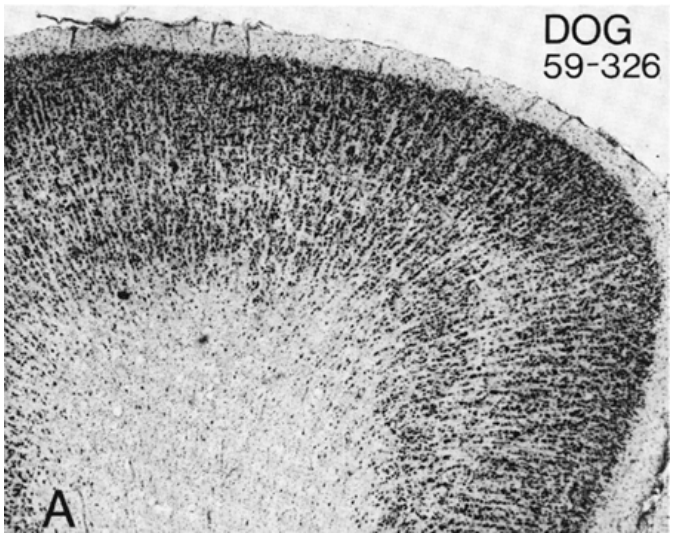

Fig. 1. Stained cortex showing pyramidal cell columns.

1. Find the crest and valley lines on the outer surface.

2. Create a mesh in the u, v plane based on crest and valley lines of outer surface.

3. Reparameterize by mapping the mesh defined in the u, v plane to a new u, v plane.

4. Create a mesh in the u, v plane of the inner surface based on the crest and valley lines of the outer surface.

5. Reparameterize the inner surface by mapping the mesh defined in the u, v plane to a new u, v plane.

6. Create w-isocurves by dropping the intersection points of the u, and v isocurves on the outer surface down to the intersection points of the u and v isocurves on the inner surface.

B.1 Crest and Valley Area Extraction on the Outer Surface

First we need to find points on the outer surface which fall near the crest lines of the gyri and the valley lines of the sulci. According to differential geometry, each point on a surface can be characterized by two principle curvatures, one in the direction of the maximum curvature, k_1, and the other orthogonal to it, k_2. Crest lines are defined by Monga and Benayoun as the successive loci of a surface whose largest principle curvature is locally maximal in the direction of its principal direction (figure []) [15]. They also derive a formula for *extremality*, whose zero crossing determines the crest lines: $\vec{\nabla} k_1 \cdot \vec{t_1} = 0$. Here k_1 is the principal curvature with maximal curvature in absolute value, and $\vec{t_1}$ is the associated principal direction. We can extend this definition of extremality to include the valley lines. Valley lines can be defined as the successive loci of a surface whose largest principle curvature is locally minimal in the direction of its principle direction. The formula for *extremality* is then: $\vec{\nabla} k_1 \cdot \vec{t_1} = 0$, where k_1 is the principal curvature with extremal curvature in absolute value, and $\vec{t_1}$ is the associated principal direction. If we sample the outer surface and calculate the extremality at regular intervals on the u and v parameters, we can find a set of points on the outer surface which is near the crest lines and valley lines (Figure 2). This is done by finding all points passing the following condition: $|\vec{\nabla} k_1 \cdot \vec{t_1}| < \rho$, where ρ is a threshold for the extremality values.

The principle curvature can be calculated using a method introduced by Maillot, Yahia, and Verroust [13]. This method estimates a curvature matrix for each sample point on the surface by taking the differences between surface normals at neighboring

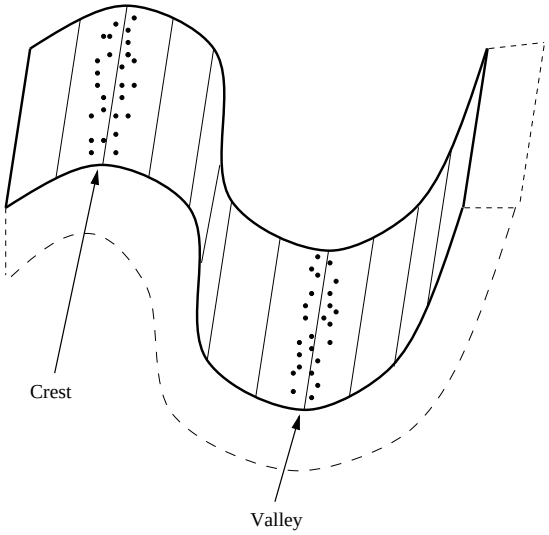

Fig. 2. Extremality points on the surface.

sample points. The curvature matrix calculated is that which best approximates the differences in the normals in a least-squares sense.

B.2 Create a Mesh in the u, v Plane of the Outer Surface

Next we fit loose-fitting planar approximating splines to the u, v coordinates of the points found to be in the area of the crest and valley lines. These lines represent an approximation of the crest and valley lines in u, v space, and will be used as new u-isocurves in the reparameterization of u, v space (Figures 3, 4). Methods for generating planar approximating splines can be found in [5, 12].

New v-isocurves can now be defined by defining planar cubic hermite spline curves which run perpendicular to the new u-isocurves (Figure 5). Hermite curves are defined by two points and two tangent vectors. We can find the two points by sampling the new u-isocurves. The two tangents are simply scalar multiples of the normals of the u-isocurves at the two points.

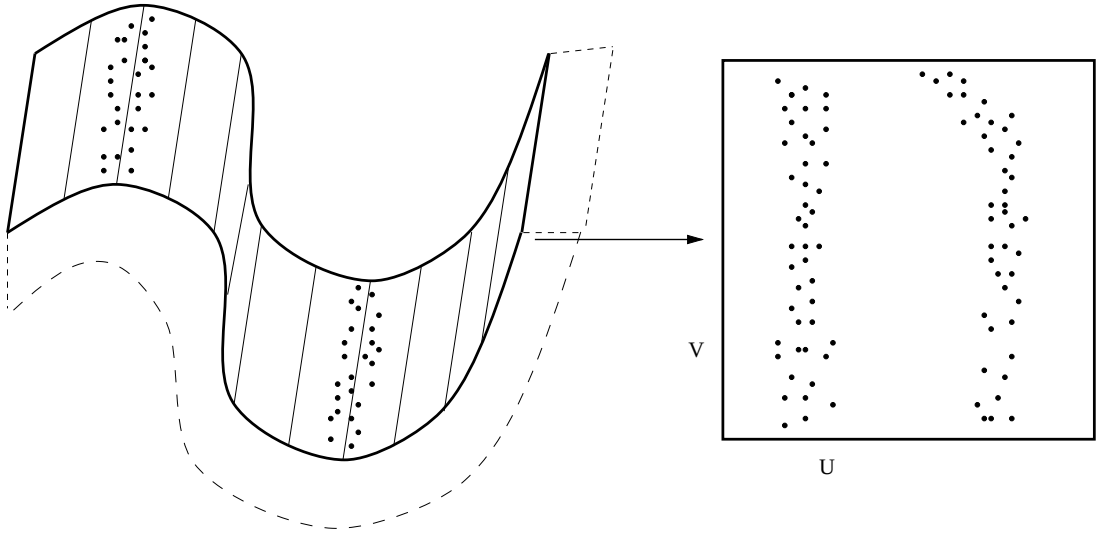

Fig. 3. U, V coordinates of the surface extremality points.

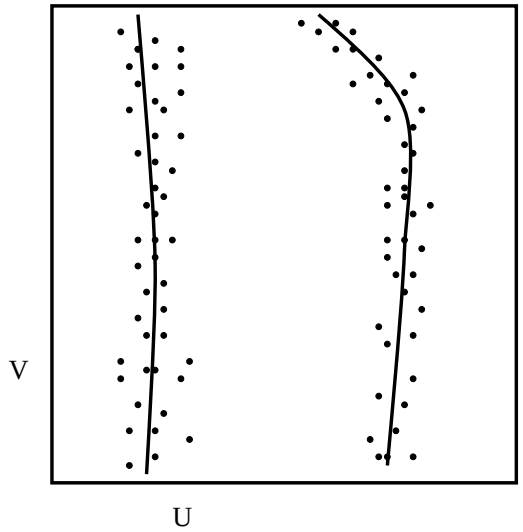

Fig. 4. Splines fit to the extremality points in the u, v plane.

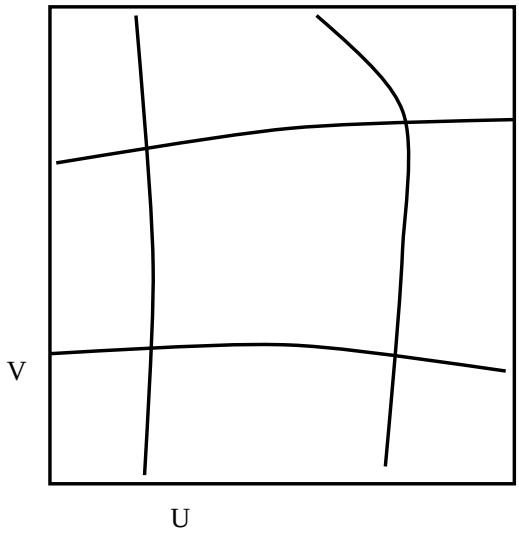

Fig. 5. V-isocurves on the u, v plane.

B.3 Reparameterize the Outer Surface

Using the mesh defined in the previous section, we can reparameterize the outer surface of the solid model. This can be achieved by mapping a new u, v plane into the old u, v plane (Figure 6. This can be done using the same mappings defined by planar grid generation. The boundaries needed by the grid generators are provided by the mesh. The mappings provided by the grid generators define a new curvilinear coordinate system in the old u, v space. A discussion of planar grid generation methods can be found in the following chapter.

B.4 Create a Mesh in the u, v Plane of the Inner Surface

To generate a mesh in the u, v plane of the inner surface, we will project the points which were found on the outer surface to the inner surface (Figure 7). Approximating planar splines can be fit to the u, v coordinates for u-isocurves, and perpendiculars define the v-isocurves, exactly as was done for the outer surface.

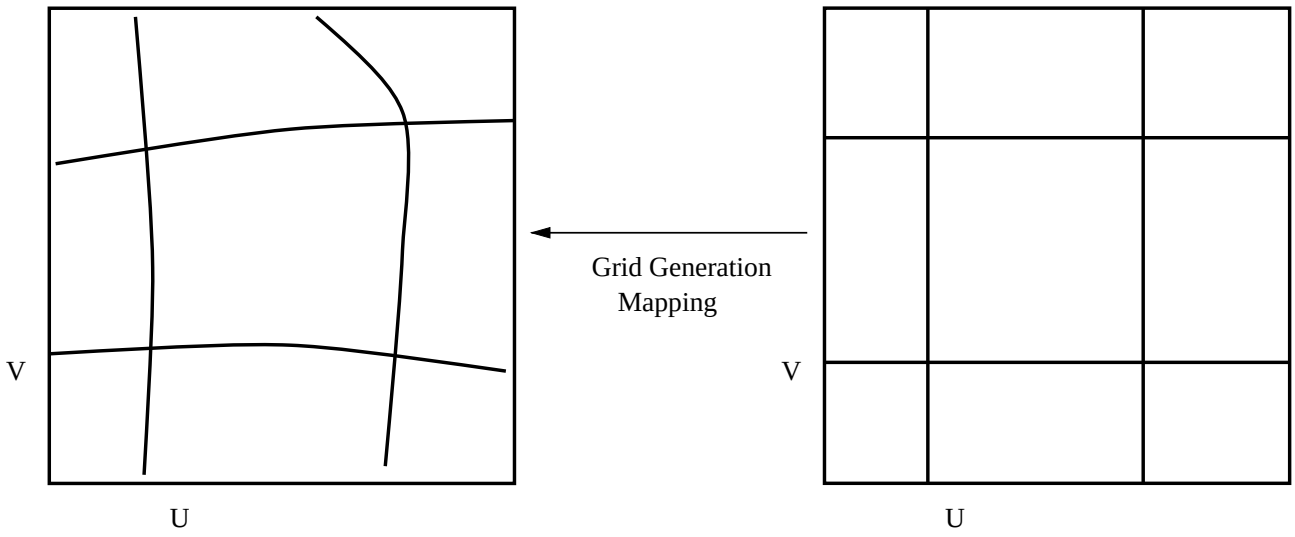

Fig. 6. Reparameterization of the mesh in the u, v plane.

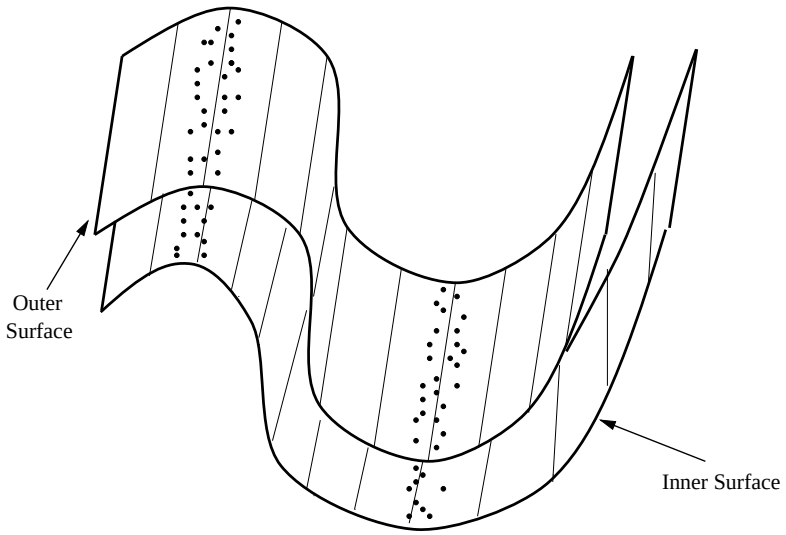

Fig. 7. Projection of the extremality points to the inner surface.

The simplest method of projecting the points from the outer surface to the inner surface is to project along the inverted normal direction of the outer surface. This method, however, is not biologically consistent, and there are problems with crossing normals due to the curvature of the surface. Ideally the points should be projected along paths which coincide with the major vertical axes of the pyramidal cells. The curvature of the major axis of a pyramidal cell is simple and can be expressed using a quadratic curve. Unfortunately the curvature of the axes of these cells depend on the location of the cells in the tissue that the projection is to occur, and this information will generally not be known. The solution to this problem is to be found through experimentation.

B.5 Reparameterize the Inner Surface

The inner surface is reparameterized in the exact same way as the outer surface. A new u, v plane is mapped into the old u, v plane using the grid generation methods described in chapter[].

B.6 Create w-Isocurves in the Solid Model

The final step is to create the w-isocurves which connect the outer surface u, v mesh to the inner surface u, v mesh. The w-isocurves should follow the natural vertical axes defined by the pyramidal cells. The curvature of these axes, as stated above, are simple and can be expressed using quadratic curves. The quadratic curves can be defined using the same method as used in projecting the points from the top surface to the bottom surface. There is, however, the additional constraint that each w-isocurve

must intersect the inner surface at the appropriate u, v isocurve intersection point.

C. Cerebellar Finite Element Decomposition

The primary native neuron type in the cerebellar cortex is the purkinje cell. The planar nature of purkinje cells suggests the definition of the v-isocurves as parallel to the planes of the cells. The u-isosurfaces are defined by the ridge lines of the gyri which primarily run parallel to the normal of to the planes defined by the purkinje cells. The algorithm for defining the isocurves is identical to that of the cerebral cortex with two exceptions: the w-isocurves are defined by the major vertical axis of purkinje cells instead of pyramidal cells, and the v isocurves, as stated before, should be parallel to the planes defined by the purkinje cells.

C.1 Hippocampal Finite Element Decomposition

The hippocampus is a prominent brain formation which will serve as an extreme example of a brain nuclei in which we can test our methods. The hippocampal formation can be described as a long sheet which has been folded several times, similar to a jelly roll . The major cell type of the hippocampus is the pyramidal cell. If the hippocampus is cut across the folds, then the pyramidal cells are aligned in a radiating pattern, similar to the cerebral cortex (Figure 8). Again, the u, v isocurves can be described as a mesh on the outer and inner surfaces with the w-isocurves spanning between them. The u-curves can be defined as curves which run parallel to the axis of the folds, while the v-isocurves are defined as curves which run perpendicular to the direction of the folds. The w-isocurves are created by following the vertical axes

of the pyramidal cells using the same technique as used in the cortical finite element decomposition.

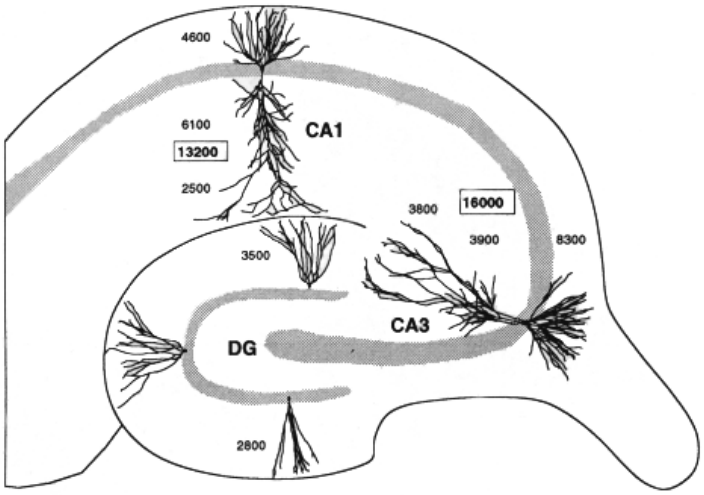

Fig. 8. Pyrimidal cell alignment in the hippocampus.

CHAPTER IV

NUMERICAL GRID GENERATION

A. Grid Generation and boundary conforming coordinate systems

Grid generation methods define boundary conforming curvilinear coordinate systems. Thompson defines a boundary conforming coordinate systems as a "curvilinear coordinate system having some coordinate line (surface in 3D) coincident with each segment of the boundary of a region."[22] Grid generation methods define such coordinate systems within complex physical regions (physical space) by mapping simpler regions (logical space) into the complex regions.

If the unit square in 2D, or the unit cube in 3D, can be mapped to the physical region such that the boundaries of the logical region are mapped to the boundaries of the physical region, then mappings can be produced to generate boundary conforming coordinate systems (Figure 9). The coordinate lines in this coordinate system are generated by the image of uniform coordinate lines mapped from logical space into physical space. The mapping is required to have a non-zero jacobian within the solid, which insures that an inverse mapping exists.

If a set of points are chosen in logical space, then the map can be used to find the corresponding set of points in physical space. In grid generation the points in logical space are generally chosen such that they form a rectangular (or cubical in 3D) grid.

There are two undesirable effects of a mapping that can arise:

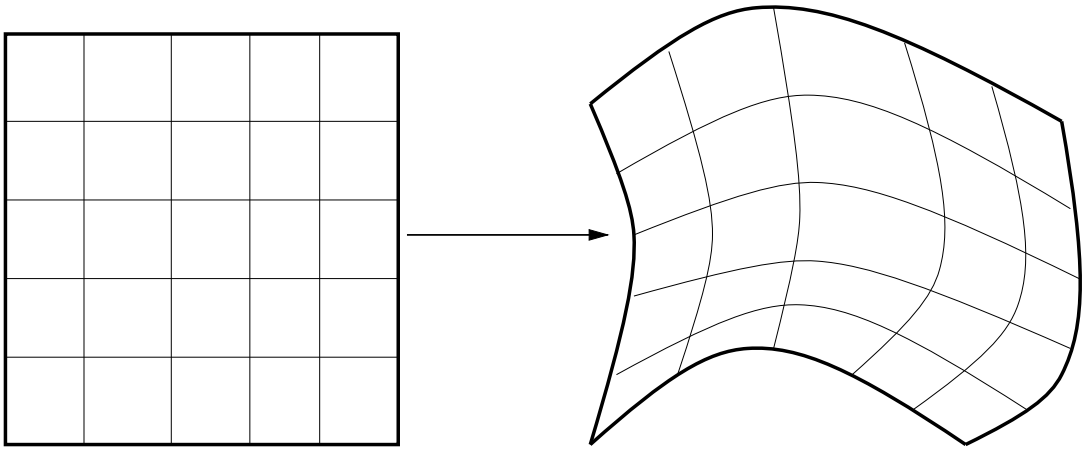

Fig. 9. 2D mapping between logical space and physical space.

1. two points in logical space can map to a single point in physical space, and

2. a point in logical space can map to a point outside the boundary in physical space, thereby producing a non-boundary conforming map. This effect is known as *folding*.

It is therefore important to insure that the mapping is *one-to-one* and *onto*.

B. Boundaries for Grid Generation

Before grid generation can be performed, the boundaries of the physical object must be defined numerically. Grid generation methods require the boundaries to be defined parametrically. If the known boundaries are given implicitly, then the boundaries must be converted to a parametric form. If the boundaries of the physical region are given by a nonsingular parametric map, then we can define a grid on the physical object by extending this map to a map from the interior of logical space to the

interior of physical space. Since grid generation methods depend largely upon boundary specification, Chapter [] addresses the issue of biologically consistent boundary specification and parameterization.

C. Description of Notation

The notation used in this chapter is the same as presented in *Fundamentals of Grid Generation* by Knupp and Steinberg [11].

The variables x, y and z are used as coordinates in physical space, and ξ, η, and ζ are used as coordinates in logical space. For the description of algorithms it is helpful to use a standard vector notation. In logical space $\xi = (\xi_1, \xi_2, \xi_3)$ is used, while in physical space $\mathbf{x} = (x_1, x_2, x_3)$ such that $\xi_1 = \xi$, $\xi_2 = \eta$, $\xi_3 = \zeta$, and $x_1 = x$, $x_2 = y$, $x_3 = z$.

The transformation, or mapping, from logical to physical space to give a system of general coordinates on the physical object can be written as

$$\mathbf{x} = \mathbf{x}(\xi, \eta, \zeta), \quad 0 \leq \xi \leq 1, \quad 0 \leq \eta \leq 1, \quad 0 \leq \zeta \leq 1,$$

and in scalar form

$$\mathbf{x} = x(\xi, \eta, \zeta), \quad y = y(\xi, \eta, \zeta), \quad z = z(\xi, \eta, \zeta).$$

Given a smooth function of (ξ, η, ζ), the three coordinate line tangents are

$$\begin{aligned} \mathbf{x}_\xi &= (x_\xi, y_\xi, z_\xi), \\ \mathbf{x}_\eta &= (x_\eta, y_\eta, z_\eta), \\ \mathbf{x}_\zeta &= (x_\zeta, y_\zeta, z_\zeta). \end{aligned}$$

The Jacobian matrix $\mathcal{J}$ is

$$\mathcal{J} = \begin{pmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{pmatrix}.$$

The determinant J of the Jacobian matrix is

$$J = det(\mathcal{J}) = x_\xi y_\eta z_\zeta + x_\eta y_\zeta z_\xi + x_\zeta y_\xi z_\eta - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi.$$

The covariant metric tensor is

$$\mathcal{G} = \mathcal{J}^T \mathcal{J} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{12} & g_{22} & g_{23} \\ g_{13} & g_{23} & g_{33} \end{pmatrix},$$

where the entries g_{ij} are scalar products of the form

$$g_{ij} = \mathbf{x}_{\xi_i} \cdot \mathbf{x}_{\xi_j},$$

with i, j=1,2,3, $\xi_1 = \xi, \xi_2 = \eta, \xi_3 = \zeta$. The determinant of $\mathcal{G}$ is denoted by g.

D. Grid Generation in Two Dimensions

In using grid generation for brain visualization we are most often interested in volume mapping. This requires 3D grid generation. Examining 2D grid generation, however, can also be useful. Planar grid generation allows us to examine the behavior of grid generation methods in a simpler model.

The basic problem of planar grid generation is: given a simply connected planar region in physical space, find a mapping $\mathbf{x} = \mathbf{x}(\xi, \eta)$ from the unit square (logical space) to the physical region [11].

The solution to the planar grid generation problem begins with the description of the physical region. The boundaries of a planar region are given by four planar curves. In parametric form the boundary equations are

$$\mathbf{x}_b(\xi), \quad \mathbf{x}_t(\xi), \quad 0 \leq \xi \leq 1,$$

$$\mathbf{x}_l(\eta), \quad \mathbf{x}_r(\eta), \quad 0 \leq \eta \leq 1,$$

which describe each part of the boundary. The subscripts on $\mathbf{x}$ stand for the bottom, top, left, and right boundaries of the logical domain.

The easiest parameterization is that of the identity map in which the unit square is mapped onto itself. This parameterization is as follows:

$$\begin{aligned} x_b(\xi) &= (\xi, 0), 0 \leq \xi \leq 1, \\ x_t(\xi) &= (\xi, 1), 0 \leq \xi \leq 1, \\ x_l(\eta) &= (0, \eta), 0 \leq \eta \leq 1, \\ x_r(\eta) &= (1, \eta), 0 \leq \eta \leq 1. \end{aligned}$$

There are four consistency constraints, one for each corner, which also must be satisfied. They are

$$\begin{aligned} \mathbf{x}_b(0) &= \mathbf{x}_l(0), \\ \mathbf{x}_b(1) &= \mathbf{x}_r(0), \\ \mathbf{x}_r(1) &= \mathbf{x}_t(1), \\ \mathbf{x}_l(1) &= \mathbf{x}_t(0). \end{aligned}$$

Once the boundary map has been found, grid generation methods can be used to map the interior of the region. The planar methods described in this thesis are transfinite interpolation, elliptical grid generation, and variational grid generation.

D.1 Planar Transfinite Interpolation

Transfinite interpolation (TFI) is the simplest of the grid generation methods. This method uses interpolation techniques on the boundary values to determine interior grid values on the physical object.

If $K+1$ planar curves $V_k(\xi); k = 0, 1, \ldots, K$ in the ξ direction and $L+1$ planar curves $W_l(\eta); l = 0, 1, \ldots, L$ in the η direction are given, then the transfinite interpolation equation which interpolate a grid between the curves is:

$$\begin{aligned}\mathbf{x}(\xi,\eta) &= \sum_{k=0}^{K}\phi_k(\eta)V_k(\xi) + \sum_{l=0}^{L}\psi_l(\xi)W_l(\eta) \\ &\quad -\{\sum_{k=0}^{K}\sum_{l=0}^{L}\phi_k(\eta)\psi_l(\xi)V_k(\xi)W_l(\eta)\}\end{aligned}$$

where $\phi_k(\eta)$ and $\psi_l(\xi)$ are interpolating polynomials [7].

If we choose first degree Lagrange polynomials, $1-\xi, \xi, 1-\eta, \eta$, as the interpolation functions, then the TFI formula for grid generation between 4 boundary edges is

$$\begin{aligned}\mathbf{x}(\xi,\eta) &= (1-\eta)\mathbf{x}_b(\xi) + \eta\mathbf{x}_t(\xi) + (1-\xi)\mathbf{x}_l(\eta) + \xi\mathbf{x}_r(\eta) \\ &\quad -\{\xi\eta\mathbf{x}_t(1) + \xi(1-\eta)\mathbf{x}_b(1) + \eta(1-\xi)\mathbf{x}_t(0) + (1-\xi)(1-\eta)\mathbf{x}_b(0)\}.\end{aligned}$$

Other interpolation functions can also be used such as splines, Hermite polynomials, and Bernstein polynomials. These interpolation functions allow the slopes of the grid lines to match, resulting in smoother grids.

D.2 Planar Elliptic Grid Generation

Elliptic grid generators are based on the solution of systems of elliptic partial differential equations. The advantage of such methods over the interpolation methods is that they produce smooth grids even when given unsmooth boundaries. The major disadvantage of the elliptic methods is the amount of computational work required to solve the equations.

D.2.1 Inhomogenous Thompson-Thames-Mastin Grid Generation

The Inhomogenous Thompson-Thames-Mastin (ITTM) generator is the most common elliptic grid generator. The ITTM generator grants control over the interior of the grid points by using two weight functions which can be used to attract or repel grid nodes or coordinate lines. The ITTM approach as introduceded by Warsi [24] requires the components of the inverse transformation $\xi = \xi(x,y), \eta = \eta(x,y)$ to satisfy

$$\begin{aligned} \nabla^2_{\mathbf{x}}\xi &= \xi_{xx} + \xi_{yy} = \frac{g_{22}}{g}P, \\ \nabla^2_{\mathbf{x}}\eta &= \eta_{xx} + \eta_{yy} = \frac{g_{11}}{g}Q, \end{aligned}$$

where $\nabla^2_x\xi$ and $\nabla^2_x\eta$ are Laplace operators with respect to the logical space. The control functions P and Q are

$$\begin{aligned} P(\xi,\eta) &= -\sum_{m=1}^{M} a_m \frac{\xi-\xi_m}{\mid \xi-\xi_m \mid} e^{-c_m|\xi-\xi_m|} \\ &\quad -\sum_{i=1}^{I} b_i \frac{\xi-\xi_i}{\mid \xi-\xi_i \mid} e^{-d_i[(\xi-\xi_i)^2+(\eta-\eta_i)^2]^{\frac{1}{2}}}, \\ Q(\xi,\eta) &= -\sum_{m=1}^{M} a_m \frac{\eta-\eta_m}{\mid \eta-\eta_m \mid} e^{-c_m|\eta-\eta_m|} \end{aligned}$$

$$-\sum_{i=1}^{I} b_i \frac{\eta - \eta_i}{\mid \eta - \eta_i \mid} e^{-d_i[(\xi-\xi_i)^2+(\eta-\eta_i)^2]^{\frac{1}{2}}},$$

where M is the number of lines and I is the number of points the grid is to be attracted to and $a_m, b_i, c_m, d_i, \xi_i$, and η_i are parameters.

D.3 Variational Planar Grid Generation

Variational grid generation techniques provide another method of building grids. Using the principles of the calculus of variations, grid generation techniques can be developed that control the length of segments, area of cells, orthogonality of the angles between grid lines, and other grid properties. The principles of calculus of variations are not covered here, but for more information see [26].

The goal in variational grid generation is to find grids that best fit the desired grid properties specified by a functional. The first step is to find functionals whose minimum is a transformation having properties that produce these effects in the grids. The grid is then determined by computing the first variation of the functional. The resulting Euler-Lagrange equations are the partial differential equations which are numerically solved to determine the grid. The variational technique produces grids that are solutions in a *least squares* sense.

The next sections present the weighted length functional, weighted area functional, and the orthogonality functional. The Euler-Lagrange equations for each of these functionals are also given. The final section gives a brief discussion on combining these fundamental functionals to produce functionals which overcome the limitations of the individual functionals.

D.3.1 The Weighted Length Functional

The goal of the length functional is to find a grid such that the lengths $\sqrt{g_{11}}$ and $\sqrt{g_{22}}$ of the ξ and η coordinate lines are proportional respectively to the square root of two given positive logical weight functions $\phi = \phi(\xi, \eta)$ and $\psi = \psi(\xi, \eta)$. The functional which needs to be minimized to achieve this is

$$I_L[\mathbf{x}] = \frac{1}{2} \int_0^1 \int_0^1 \left(\frac{g_{11}}{\phi} + \frac{g_{22}}{\psi} \right) d\xi d\eta.$$

The first variation gives the following Euler-Lagrange equation:

$$\left(\frac{x_\xi}{\phi} \right)_\xi + \left(\frac{x_\eta}{\psi} \right)_\eta = 0.$$

The solution of this partial differential equation produces grids such that the segment lengths are proportional to the weight functions.

D.3.2 The Weighted Area Functional

The goal of the area functional is to find a grid such that the area of the cells in the grid are proportional to some weight function $\phi = \phi(\xi, \eta) > 0$. The functional to achieve this is

$$I_A[\mathbf{x}] = \frac{1}{2} \int_0^1 \int_0^1 \frac{g}{\phi} d\xi d\eta = \frac{1}{2} \int_0^1 \int_0^1 \frac{J^2}{\phi} d\xi d\eta.$$

The first variation gives the following Euler-Lagrange equation:

$$J_\xi \mathbf{x}_\eta - J_\eta \mathbf{x}_\xi - J \frac{\phi_\xi}{\phi} \mathbf{x}_\eta + J \frac{\phi_\eta}{\phi} \mathbf{x}_\xi = 0.$$

Unfortunately the area equations are quasilinear, coupled, and nonelliptic. Therefore solutions to the equations might not exist for some arbitrary weight functions or boundary data.

D.3.3 The Orthogonality Functional

The goal of the orthogonal functional is to find a grid such that the grid lines which run in the ξ and η directions are orthogonal. The functional whose minimum attempts to satisfy the orthogonality condition in a least squares sense is

$$I_O[\mathbf{x}] = \frac{1}{2}\int_0^1 \int_0^1 g_{12}^2 d\xi d\eta.$$

If the minimum is zero then the mapping is orthogonal. If it is greater than zero, then the solution is the closest to orthogonal in a least squares sense. The Euler-Lagrange equation for the functional is

$$(g_{12}\mathbf{x}_\eta)_\xi + (g_{12}\mathbf{x}_\xi)_\eta = 0.$$

The orthogonality equations are also quasilinear, coupled, and nonelliptic. Solutions to these equations are not guaranteed.

D.3.4 Combinations of Functionals

Since there are limitations to the above functionals, minimization of weighted combinations of the functionals can achieve a compromise between the properties controlled by the functionals. The basic form for the weighted combination functional was introduced in [21] and is:

$$I_w[\mathbf{x}] = w_L I_L + w_A I_A + w_O I_O,$$

where w_L, w_A, and w_O are weight parameters such that their sum is one. The functionals whose weight function include $w_O = 0$ are studied by Castillo et al. [3] They are shown to produce smooth nonfolded grids on many physical regions.

E. Grid Generation in Three Dimensions

The basic problem of three dimensional grid generation is: given a simply connected volume in physical space, find a mapping $\mathbf{x} = \mathbf{x}(\xi, \eta, \zeta)$ from a unit cube (logical space) to the physical region[11] (Figure 10).

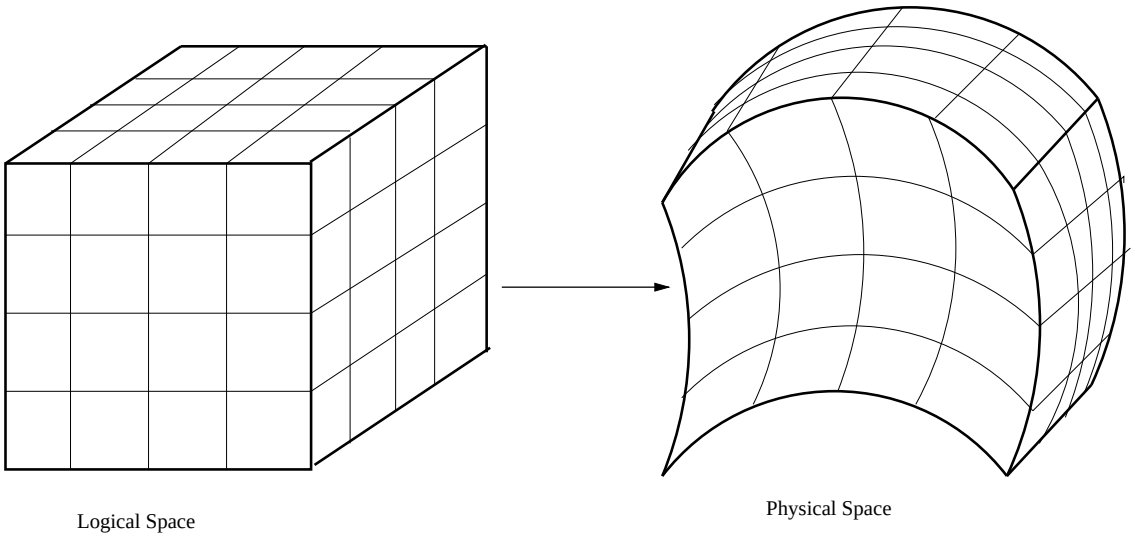

Fig. 10. 3D mapping between logical space and physical space.

Boundary specification of the physical region in 3D is much more complex then in 2D. Six bounding surfaces are required and are given by:

$$\mathbf{x}_W(\eta, \zeta), \quad \mathbf{x}_E(\eta, \zeta), \quad 0 \leq \eta, \zeta \leq 1,$$

$$\mathbf{x}_N(\xi, \zeta), \quad \mathbf{x}_S(\xi, \zeta), \quad 0 \leq \xi, \zeta \leq 1,$$

$$\mathbf{x}_T(\xi, \eta), \quad \mathbf{x}_B(\xi, \eta), \quad 0 \leq \xi, \eta \leq 1,$$

where the subscripts on $\mathbf{x}$ stand for *North, South, East, West, Top, Bottom.*

There are twelve edges:

$$\mathbf{x}_{SW}(\zeta), \quad \mathbf{x}_{SE}(\zeta), \quad \mathbf{x}_{NW}(\zeta), \quad \mathbf{x}_{NE}(\zeta),$$

$$\mathbf{x}_{BW}(\eta), \quad \mathbf{x}_{TW}(\eta), \quad \mathbf{x}_{BE}(\eta), \quad \mathbf{x}_{TE}(\eta),$$

$$\mathbf{x}_{BS}(\xi), \quad \mathbf{x}_{TS}(\xi), \quad \mathbf{x}_{BN}(\xi), \quad \mathbf{x}_{TN}(\xi),$$

and eight corner points:

$$\mathbf{x}_{WSB}, \quad \mathbf{x}_{WST}, \quad \mathbf{x}_{WNB}, \quad \mathbf{x}_{WNT},$$

$$\mathbf{x}_{ESB}, \quad \mathbf{x}_{EST}, \quad \mathbf{x}_{ENB}, \quad \mathbf{x}_{ENT}.$$

The consistency conditions for the boundary surfaces are:

$$\begin{aligned}
\mathbf{x}(0,\eta,\zeta) &= \mathbf{x}_W(\eta,\zeta), \\
\mathbf{x}(1,\eta,\zeta) &= \mathbf{x}_E(\eta,\zeta), \\
\mathbf{x}(\xi,0,\zeta) &= \mathbf{x}_S(\xi,\zeta), \\
\mathbf{x}(\xi,1,\zeta) &= \mathbf{x}_N(\xi,\zeta), \\
\mathbf{x}(\xi,\eta,0) &= \mathbf{x}_B(\xi,\eta), \\
\mathbf{x}(\xi,\eta,1) &= \mathbf{x}_T(\xi,\eta).
\end{aligned}$$

The consistency conditions for the edges are:

$$\begin{aligned}
\mathbf{x}(0,0,\zeta) &= \mathbf{x}_{SW}(\zeta), \\
\mathbf{x}(0,1,\zeta) &= \mathbf{x}_{NW}(\zeta), \\
\mathbf{x}(1,0,\zeta) &= \mathbf{x}_{SE}(\zeta), \\
\mathbf{x}(1,1,\zeta) &= \mathbf{x}_{NE}(\zeta), \\
\mathbf{x}(0,\eta,0) &= \mathbf{x}_{BW}(\eta), \\
\mathbf{x}(0,\eta,1) &= \mathbf{x}_{TW}(\eta),
\end{aligned}$$

$$\begin{aligned}
\mathbf{x}(1,\eta,0) &= \mathbf{x}_{BE}(\eta), \\
\mathbf{x}(1,\eta,1) &= \mathbf{x}_{TE}(\eta), \\
\mathbf{x}(\xi,0,0) &= \mathbf{x}_{BS}(\xi), \\
\mathbf{x}(\xi,0,1) &= \mathbf{x}_{TS}(\xi), \\
\mathbf{x}(\xi,1,0) &= \mathbf{x}_{BN}(\xi), \\
\mathbf{x}(\xi,1,1) &= \mathbf{x}_{TN}(\xi).
\end{aligned}$$

The consistency conditions for the corner points are:

$$\begin{aligned}
\mathbf{x}(0,0,0) &= \mathbf{x}_{WSB}, \\
\mathbf{x}(1,0,0) &= \mathbf{x}_{ESB}, \\
\mathbf{x}(0,1,0) &= \mathbf{x}_{WNB}, \\
\mathbf{x}(0,0,1) &= \mathbf{x}_{WST}, \\
\mathbf{x}(1,1,1) &= \mathbf{x}_{ENT}, \\
\mathbf{x}(0,1,1) &= \mathbf{x}_{WNT}, \\
\mathbf{x}(1,0,1) &= \mathbf{x}_{EST}, \\
\mathbf{x}(1,1,0) &= \mathbf{x}_{ENB}.
\end{aligned}$$

The problem of three dimensional grid generation is considerably more difficult then the planar grid generation problem. It is often difficult to map 3D physical objects to a unit cube. For many objects this limitation can be overcome by subdividing the object into a union of finite elements. We use this method for grid generation of the cerebral cortex (see Chapter).

E.1 3D Transfinite Interpolation

3D Transfinite Interpolation is the most common 3D grid generator used in commercial packages [11]. The 3D TFI equation was introduced by Gordon in [7] and is as follows: if $K+1$ surfaces $\{U_k(\eta,\zeta), k=0,1,\ldots,K\}$, $L+1$ surfaces $\{V_l(\xi,\zeta), l=0,1,\ldots,L\}$, and $M+1$ surfaces $\{W_m(\eta,\zeta), m=0,1,\ldots,M\}$ are given, then

$$\begin{aligned}
\mathbf{x}(\xi,\eta,\zeta) &= \sum_{k=0}^{K}\phi_k(\xi)U_k(\eta,\zeta) \\
&+\sum_{l=0}^{L}\psi_l(\eta)V_l(\xi,\zeta) \\
&+\sum_{m=0}^{M}\theta_m(\zeta)W_m(\xi,\eta) \\
&-\sum_{k=0}^{K}\sum_{l=0}^{L}\phi_k(\xi)\psi_l(\eta)U_k(\eta,\zeta)V_l(\xi,\zeta) \\
&-\sum_{l=0}^{L}\sum_{m=0}^{M}\psi_l(\eta)\theta_m(\zeta)V_l(\xi,\zeta)W_m(\xi,\eta) \\
&-\sum_{k=0}^{K}\sum_{m=0}^{M}\phi_k(\xi)\theta_m(\zeta)U_k(\eta,\zeta)W_m(\xi,\eta) \\
&+\sum_{k=0}^{K}\sum_{l=0}^{L}\sum_{m=0}^{M}\phi_k(\xi)\psi_l(\eta)\theta_m(\zeta)U_k(\eta,\zeta)V_l(\xi,\zeta)W_m(\xi,\eta),
\end{aligned}$$

where $\phi_k(\eta)$, $\psi_l(\xi)$, and $\theta_m(\zeta)$ are interpolating polynomials [7].

If we choose first degree Lagrange polynomials, $1-\xi, \xi, 1-\eta, \eta, 1-\zeta, \zeta$, as the interpolation functions, then the TFI formula for grid generation between 6 boundary surfaces is

$$\mathbf{x}(\xi,\eta,\zeta) = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 - \mathbf{x}_{12} - \mathbf{x}_{13} - \mathbf{x}_{23} + \mathbf{x}_{123},$$

where

$$\mathbf{x}_1 = (1-\xi)x_W(\eta,\zeta) + (\xi)x_E(\eta,\zeta),$$

$$
\begin{aligned}
\mathbf{x}_2 &= (1-\eta)x_S(\xi,\zeta) + (\eta)x_N(\xi,\zeta), \\
\mathbf{x}_3 &= (1-\zeta)x_B(\xi,\eta) + (\zeta)x_T(\xi,\eta), \\
\mathbf{x}_{12} &= (1-\xi)(1-\eta)x_{SW}(\zeta) + (1-\xi)\eta x_{NW}(\zeta) \\
&\quad +\xi(1-\eta)x_{SE}(\zeta) + \xi\eta x_{NE}(\zeta), \\
\mathbf{x}_{13} &= (1-\xi)(1-\zeta)x_{BW}(\eta) + (1-\xi)\zeta x_{TW}(\eta) \\
&\quad +\xi(1-\zeta)x_{BE}(\eta) + \xi\zeta x_{TE}(\eta) \\
\mathbf{x}_{23} &= (1-\eta)(1-\zeta)x_{BS}(\xi) + (1-\eta)\zeta x_{TS}(\xi) \\
&\quad +\eta(1-\zeta)x_{BN}(\xi) + \eta\zeta x_{TN}(\xi), \\
\mathbf{x}_{123} &= (1-\xi)(1-\eta)(1-\zeta)x_{WSB} + (1-\xi)(1-\eta)\zeta x_{WST} \\
&\quad +(1-\xi)\eta(1-\zeta)x_{WNB} + (1-\xi)\eta\zeta x_{WNT} \\
&\quad +\xi(1-\eta)(1-\zeta)x_{ESB} + \xi(1-\eta)\zeta x_{EST} \\
&\quad +\xi\eta(1-\zeta)x_{ENB} + \xi\eta\zeta x_{ENT}.
\end{aligned}
$$

As in the 2D case, other interpolating functions can be used to insure continuity across elements for physical regions subdivided into finite elements as discussed above.

E.2 3D Inhomogenous Thompson-Thames-Mastin Grid Generation

The 3D ITTM generator, like its planar counterpart, is based on the solution of a system of elliptic PDE's. The functions are based on the map $\xi = \xi(x,y,z), \eta = \eta(x,y,z), \zeta = \zeta(x,y,z)$ from physical to logical space.

The 3D ITTM equations are

$$\begin{aligned}
\nabla^2_{\mathbf{x}}\xi &= \xi_{xx}+\xi_{yy}+\xi_{zz} &&= \tfrac{g_{33}}{g}P,\\
\nabla^2_{\mathbf{x}}\eta &= \eta_{xx}+\eta_{yy}+\eta_{zz} &&= \tfrac{g_{22}}{g}Q,\\
\nabla^2_{\mathbf{x}}\zeta &= \zeta_{xx}+\zeta_{yy}+\zeta zz &&= \tfrac{g_{11}}{g}R,
\end{aligned}$$

where P, Q, and R are logical space weight functions.

E.3 3D Variational Grid Generation

The 3D variational grid generators offer the same control over grids as the planar variational generators. The generalized 3D functional is

$$I[\mathbf{x}] = \int_0^1\int_0^1\int_0^1 \frac{H}{w^2(\mathbf{x})}d\xi d\eta d\zeta,$$

with H being a smooth positive function and w a weight function. Length, Area, and Orthogonality functionals can be constructed by choosing H as given in Table I.

TABLE I

Three dimensional variational principles

Principle	Form of H
Length	$g_{11}+g_{22}+g_{33}$
Area	$(\sqrt{g})^2$
Orthogonality	$g_{33}g_{12}^2+g_{11}g_{23}^2+g_{22}g_{13}^2$

The 3D Euler-Lagrange equation is derived by taking the first variation of the functional. It is:

$$\frac{H}{\sqrt{g}}div_\xi\frac{C}{w^2} - div_\xi\frac{\mathcal{JB}}{w^2} = 0,$$

where

$$\mathcal{JB} = \left[\frac{\partial H}{\partial x_\xi}, \frac{\partial H}{\partial x_\eta}, \frac{\partial H}{\partial x_\zeta}\right],$$

and

$$\mathcal{B} = \mathcal{M} + \sqrt{g}\frac{\partial H}{\partial \sqrt{g}}\mathcal{G}^{-1},$$

and

$$[\mathcal{M}]_{ij} = (1 + \delta_{ij})\frac{\partial H}{\partial g_{ij}}.$$

CHAPTER V

GROWTH AND MAPPING OF NEURONS

A. Introduction

Sparse networks of neurons are embedded in the reconstructed and gridded tissue to complete the model. Seed points are randomly generated using a distribution function. Synthetic neurons can then be grown within the curvilinear coordinate space. Optically-traced biological neurons can also be placed within the solid model, but they must be first placed in logical space and then mapped into the model.

B. Growth of Synthetic Neurons in the Solid Model

L-systems are a grammar based modeling system which generate models through parallel growth simulation. In computer graphics, L-systems are an accepted tool for plant modeling. Neurons, when scaled up by 10^5, show a strong resemblance to trees. McCormick and Mulchandani have created L-system grammars which have been used to generate synthetic neurons with promising proximity to neurons described in the neurobiology literature [14, 16, 17]. These synthetic neuron models can be grown inside the solid models using the coordinate system which was defined by the grid generation.

C. Mapping Neurons into the Solid Model

Laser Scanning Confocal Microscopy (LSCM) is an established tool for obtaining high resolution images and 3d reconstructions of biological specimens. LCSM can be used to optically section fluorescent stained brain tissue. The sections which are created can be used to trace or reconstruct the processes of the neurons in the tissue. Neurons obtained through this method are typically a series of points along the segment trajectories with the segment width at these points. To embed these neurons in the solid model, they first must be placed in logical space then mapped into the solid model (Figure 11). The mapping which is used is determined by the method of grid generation which was used to generate the curvilinear coordinate system. If the 3D extension of the de Casteljau algorithm is used, then the neurons can be directly mapped from the boundary information into the solid model using the same algorithm.

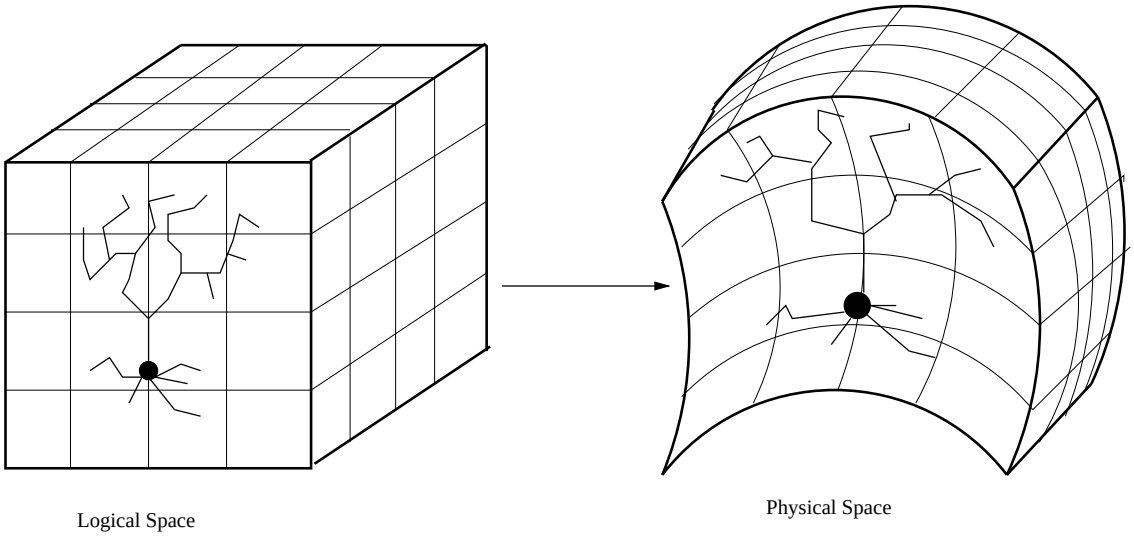

Fig. 11. Mapping a neuron from logical space to physical space.

If any of the other grid generators are used, then a continuous mapping from logical space to physical space does not exist. Another mapping must be constructed to map the neurons from logical space into the solid model. The neurons are mapped by mapping the end points of each segment. Each neuron segment end point which is inside logical space must fall either inside a single grid cell, or on the boundary of several grid cells. We can define a continuous mapping from individual grid blocks in logical space to corresponding grid blocks in physical space using the 3D transfinite interpolation grid generation technique with lagrange interpolation functions (trilinear interpolation) as introduced in section []. The boundaries which are required are simply the eight corner points in logical space and physical space.

CHAPTER VI

GRID GENERATION and NEURON MAPPING TOOLS

A. Introduction

A set of grid generation utilities has been developed to generate two and three dimensional grids. This grid generation system is a collection of modular tools which make it easily extensible.

B. Two Dimensional Grid Generation Tools

B.1 Boundary Input

The input to the 2D grid generation system consists of boundary information. The boundaries for the planar grid generators consist of control polygons for four planar cubic Bezier curves. The first utility, *edg2surf*, reads the control polygons and samples the curves to generate four boundary grids.

B.2 2D Transfinite Interpolation

The next utility, *tfi2d*, takes the four boundary grids and extends them to the interior of the domain using the bi-variate cubic transfinite interpolation equation(chapter []). The resulting grids may be suitable for some applications, if not, elliptic or variational methods may be employed.

B.2.1 3D Inhomogenous Thompson-Thames-Mastin grid generator

The planar ITTM grid generation utility, *ittm*, solves the ITTM equations given in Chapter[] to generate smooth grids on the physical domain. The first step in solving the ITTM equations is to replace the derivatives with second-order central difference equations. The resulting non-linear difference equations are then solved by iteration. A nine point successive overrelaxation (SOR) method is used for the iterative procedure.

The elliptic grid generators and the variational grid generators described below, take the initial grids as input and manipulate the grid points to find a final grid. Both sets of utilities for non-algebraic grid generators are based on the FORTRAN code included in *Fundamentals of Grid Generation* by Knupp and Steinberg [11].

B.2.2 2D Variational Grid Generators

The variational grid generation utility, *var*, generates grids by solving the Euler-Lagrange equations for the weighted area, length and orthogonality functionals. The Euler-Lagrange equations can be solved in a way similar to the ITTM equations. First the derivatives are replaced by second-order central difference equations. Again a successive overrelaxation method can be employed to solve the difference equations for the final grids.

C. Three Dimensional Grid Generation Tools

C.1 3D Boundary Input

The input to the 3D grid generation system consists of either control polygons for twelve cubic Bezier curves for boundary edges, or the control nets for six cubic Bezier surfaces. If the boundaries are specified by twelve edges, then the first step is to find the six bounding surfaces. The utility, *edge2surf*, finds the cubic Bezier surfaces by interpreting the edge boundary control polygons as piecewise linear curves and computing the bilinearly blended Coons patch that interpolates to them. The resulting bilinear Coons patch can be treated as a control net for a Bezier surface. The resulting Bezier surface is actually the bilinearly blended Coons patch to the original Bezier boundary curves [6].

The 3D grids can be derived in one of two ways: application of the 3D transfinite interpolation equation to the boundary grids, or extending the de Casteljau algorithm to three dimensions.

C.2 3D Transfinite Interpolation

The first step in the transfinite interpolation process is to discretize the boundary surfaces into boundary grids. This can be done using the two dimensional de Casteljau algorithm on the surface control nets [6]. The 3D transfinite interpolation utility, *tfi3d*, can then be applied to the boundary grids to extend them into the interior of the domain to create a final grid (Chapter[]).

C.3 3D de Casteljau Algorithm

The traditional transfinite interpolation method produces a discrete mapping between the logical domain and the physical domain. Using a three-dimensional extension of the de Casteljau algorithm, we can create a continuous algebraic mapping from the logical to the physical domain. This is preferred for brain visualization because it is useful for mapping neurons from the logical space to the physical domain.

The first utility, *surf2cub*, creates a three-dimensional control grid for a Bezier volume. It generates the control grid using a three-dimensional extension of the Coons patch technique as explained above. Given the six cubic Bezier surface control nets, a control grid is generated by treating the control nets as bilinear patches. Trilinear interpolation on the six patches creates a trilinear grid. This grid can be used as a control grid for a Bezier volume, which in turn can be used to generate the final grid.

The 3D de Casteljau utility, *cub2grid*, generates three dimensional grids by repeated application of bilinear interpolation. It is functionally the same as transfinite interpolation, but it generates a continuous mapping between the logical space and the physical domain, that is, given any (ξ, η, ζ) in logical space we can find the corresponding (x, y, z) in physical space. To generate the grid in physical space, a grid of points is defined in logical space and then mapped into physical space.

D. Neuron Mapping Tool

The neuron mapping utility, *map*, uses the 3D de Casteljau algorithm to map either synthetic, or manually-traced neurons from the logical space into the physical domain. Inputs are a set of 3D control grid data files, a set of neuron data files, and a description data file.

The description data file lists the control grid data files and neuron data files which are to be used in the mapping process. It defines the organization of the finite element control grids in space. That is, it contains the adjacency information that allows the whole physical space to be reconstructed from the finite elements. In addition to listing the neuron data files, it contains information which orients the neurons in logical space. This information includes position of the neuron soma, and three scalar values for scaling the neuron geometry along the three curvilinear axes.

The neuron data files define neurons as a set of simple line segments and a point and radius for the soma. Each line segment consists of two 3D points for end points and a width for the thickness of the segment. The mapping of the neurons from logical space into physical space is carried out by the 3D de Casteljau algorithm by simply mapping each of the segment end points. Likewise, the soma point is also mapped using the de Casteljau algorithm. The widths of the segments, and the radius of the soma are not effected by the mapping. Segments that cross finite element boundaries are recognized and are properly mapped by utilizing the control grid adjacency information. This insures that the proper control grids are used for each end point.

CHAPTER VII

VISUALIZATION OF 3D GRIDS AND EMBEDDED NEURONS

A. Introduction

An interactive 3D grid and neuron visualization tool has been developed. Using this tool multiple grids and neurons to be mapped within the grids can be loaded and examined interactively to determine the quality of the mapping. The visualizer utilizes the OpenGL graphics API for 3D display and a Motif interface (Figure 12).

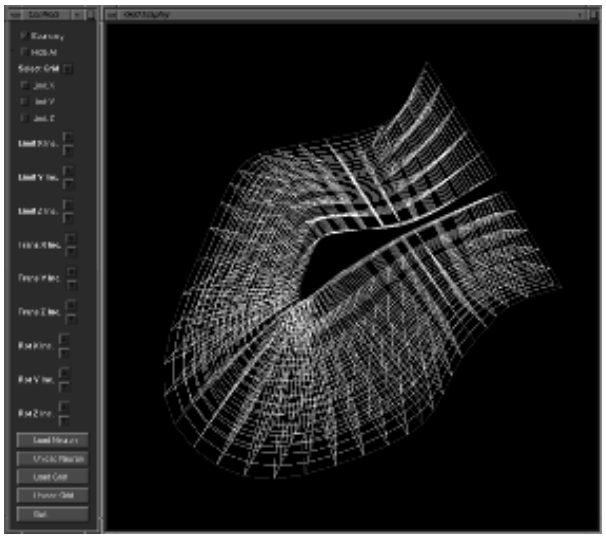

Fig. 12. 3D grid and neuron vizualization tool.

B. Input

The visualizer can read two types of files for input: grid data files and neuron data files. The grid data files are generated by the 3D grid generation tools described in Chapter[]. They contain a set of 3D points representing the nodes in the grid.

The neuron morphology is stored in a neuron data file. The neurons are defined in the data file as a set of line segments with diameters and a soma point with a diameter. These files can be generated by the Neuron Morphology Modeler developed by Mulchandani [17].

C. Features

Some salient features of the visualizer are listed below:

- *Addition and deletion of grid data files* - The visualizer allows the addition and deletion of grids through the use of dialog boxes. This allows the display of the entire gridded physical object being modeled, possibly requiring multiple grids. The grids are displayed in 3D as points connected by line segments.

- *Addition and deletion of neuron data files* - The visualizer also allows the addition and deletion of multiple neurons. This allows an entire population of neurons that have been mapped into the physical object to be visualized. The neurons are displayed as simple stick figures to allow quick interactive display on graphics workstations.

- *Display of grid boundaries* - The display of the grids often adds confusion to the display of the neuron populations. If the grids are simply not displayed, the user does not get any sense of the environment into which the neurons are mapped. To alleviate this problem the visualizer allows the user to remove the grid mesh and replace it with the bounding edges of the grid.

- *Selection and manipulation of individual grids* - The 3D grids are often very dense. This poses a problem for examining the quality of the grids. The visualizer allows the user to interactively select individual grids and manipulate the portion of grid being displayed. The grids display can be limited to isosurfaces, columns, or individual cells in all three axes. The slices, columns, and cells can then be interactively swept through the grid.

- *Navigation* - The camera view of the scene is under the users control. The camera location and gaze direction can be altered interactively.

CHAPTER VIII

RESULTS

A. Introduction

In this chapter results generated with the softwares are presented.

B. Reconstruction

A small section of human cerebral cortex is reconstructed from a data set containing images collected from the brain of a 76 year old normal female human cadaver. The images were obtained from the UCLA Laboratory of Neuro Imaging World Wide Web site. The brain was cytosectioned through the horizontal plane in 100 $/mum$ increments on a heavy duty cytomacrotome. The cytomacrotome was equipped with a high resolution camera for digital image capture of the serial images $(1024^2, 24 - bit)$. The images were scaled down $(512^2, 24 - bit)$ for distribution over the internet. 42 serial slices were used for the reconstruction.

B.1 Contour Extraction

Contours were collected from each of the images using Elastic Reality (Avid Technologies). Figure [] shows Elastic Reality with an entire slice with a contour around the piece of interest. Figure [] is a closer view of the same section and contour.

B.2 Reconstruction with Smoothing Splines

The contours were sampled, and smoothing spline surfaces were fit to the samples using the FORTRAN surface fitting routines written by Dierckx [5]. Interpolating surfaces are first fit to the data. Figure [] shows a view of the resulting reconstruction showing the outer surface of the cortex. Figure [] shows a view of the resulting reconstruction showing the inner surface of the cortex. The reconstruction using interpolating spline surfaces is very rough. This most likely demonstrates the "wiggly" noise that is picked up in the surfaces when no smoothing is used. Next smooth approximating surfaces are fit to the data. A smoothing factor $S = 1$ is used. Figures [] and [] show the same two views of the reconstructed tissue as before, only with the smooth approximating surfaces. The resulting reconstruction is much better than the interpolated surfaces. Figures [] and [] show the same two views with Gouraud interpolation shading added. This clearly demonstrates the smoothness of the reconstructed tissues. Finally, figures [] and [] are larger views showing the outer and inner surfaces of the resulting reconstructed tissues

C. Finite Element Decomposition

The method which is proposed in chapter [] has not been implemented. The finite element decomposition which was used here was a chord length sampling of the parameter domain. A grid was constructed for both surfaces in parameter space by choosing lines of constant u and v at even increments. The grid lines in parameter space define grid curve lines on the surface. The w lines are linear segments connecting

the u, v intersection points on the outer surface to the u, v intersection points on the inner surface. Figure [] and [] show the finite element grid lines on the surface of the reconstructed tissue. There was no attempt to construct elements which follow the crest lines in the tissue. The finite element lines which follow the crest and valley lines in the two figures were purely accidental.

D. Grid Generation

The 3d transfinite interpolation grid generator was used to generate grids in the reconstructed human cerebral cortex and rat hippocampus. Figure [] shows the boundary edges of a single finite element from the human cerebral cortex. Figure [] shows the finite element with a grid embedded within.

A slice of rat hippocampus was reconstructed from an illustration taken from *The Rat Nervous System* [18] using the same reconstruction techniques as above. Figure [] shows the boundaries of the finite elements. Figure [] shows the finite elements with the grids embedded within them.

E. Mapping Neurons

Finally, a sparse network of neurons is mapped into the reconstructed tissues. Figure [] shows a set of pyramidal cells mapped into the the cerebral cortex finite element. Figure [] shows a set of pyramidal cells mapped into the rat hippocampus finite elements.

CHAPTER IX

SIGNIFICANCE AND FUTURE WORK

Neurons are densely packed into neural tissue and this presents a barrier to the visualization and interpretation of globally-stained neural tissue scanned in at the limit of optical resolution. Advances in confocal microscopy and fluorescent staining of individual neurons allow the tracking of their fibers and neural processes in neural tissue. The interpretation and modeling of these neuron data sets rests on having a biologically-consistent framework in which the neuron populations can be visualized. The goal of this work is to provide effective techniques for the visualization and interpretation of neuron data sets within a volume of reconstructed neural tissue.

A. Future Work

Some of the ways in which this work can be extended and expanded are listed below.

- The contours which were generated for this work were constructed manually using piecewise cubic bezier curves. This process can be automated with the use of active contouring methods such as "snakes".

- The reconstruction methods using the approximating B-splines were attempted only on a small region of tissue. The B-spline techniques must be extended to be more robust to reconstruct larger tissue specimens, such as an entire brain.

- The proposed method for biologically-consistent finite element decomposition should be implemented and tested.
- Brain tissue must be section, stained, and analyzed to find the major axes of the primary cells of the tissue types for the finite element decomposition.
- The 3D ITTM grid generator and the 3D variation grid generators should be implemented and tested.
- A longer term objective of this research is to temporally extend these finite element models to visualize gyrogenesis of the cerebral cortex during ontogeny – from neonate to adult.

B. Summary

In this work, smooth approximating B-spline surfaces are used to reconstruct brain tissue from scanned slice data. A proposed technique decomposes the reconstructed tissue into finite elements in such a way that preserves symmetry within the tissue. Grid generation methods are used to provide a curvilinear coordinate system within the finite elements. The grids and mappings provided by the grid generators allow synthetic and manually traced neurons to be embedded inside the finite element model of the tissue. Finally, tools which visualize these grids and neuron data sets were implemented.

REFERENCES

[1] R. Bolle and B. Vemuri, "On three-dimensional surface reconstruction methods," *IEEE Trans. Pat. Anal. Mach. Intell.*, vol. 13(1), pp. 1-13, January 1991.

[2] J. Brinkley, "Knowledge-driven ultrasonic three-dimensional organ modeling," *IEEE Trans. Pat. Anal. Mach. Intell.*, vol. 7(4), pp. 431-441, July 1985.

[3] J. Castillo, J. Steinberg and P. Roache, "Parameter estimation in variational grid generation," *Appl. Math. and Comp.*, vol. 28, no. 2, pp. 1-23, 1988.

[4] C. de Boor, *A Practical Guide to Splines.* Springer, 1978.

[5] P. Dierckx, *Curve and Surface Fitting with Splines,* Oxford, England: Oxford University Press, 1995.

[6] G. Farin, *Curves and Surfaces for CAGD: a practical guide.* San Diego, CA: Academic Press, Inc, fourth edition, 1995.

[7] W. Gordon, "Spline-blended surface interpolation through curve networks," *Journal of Mathematics and Mechanics*, vol. 18, no. 10, pp. 931-952, 1969.

[8] A. Goshtasby, "Surface reconstruction from scattered measurements," *SPIE*, vol. 1830, pp. 247-256, 1992.

[9] T. Hastie and W. Stuetzle. "Principal curves," *JASA*, vol. 84, pp. 502-516, 1989.

[10] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, vol. 1, pp. 321-331, 1987.

[11] P. Knupp and S. Steinberg, *Fundamentals of Grid Generation.* Boca Raton, FL: CRC Press, 1994.

[12] P. Lancaster and K. Salkauskas, *Curve and Surface Fitting: An Introduction.* New York, NY: Academic Press, 1990.

[13] J. Mailliot, H. Yahia, and A. Verroust, "Interactive Texture Mapping," *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, New York, 1993, pp. 27-34.

[14] B. McCormick and K. Mulchandani, "L-system Modeling of Neurons," in *Proc. Visualization in Biomedical Computing.* SPIE, 2359 pp. 693-705, 1994.

[15] 0. Monga and S. Benayoun. "Using partial derivatives of 3d images to extract typical surface features." *rapport de recherche INRIA*, March 1992.

[16] K. Mulchandani and B. McCormick, "A Framework for Modeling Neuron Morphology," in *Computational Neuroscience: Trends in Research 1995*, J. M. Bower ed. San Diego, CA: Academic Press, Inc, 1996. pp. 453-458.

[17] K. Mulchandani, "Morphological Modeling of Neurons," Masters thesis, Department of Computer Science, Texas A&M University, 1995.

[18] D. Amaral and M. Witter, "Hippocampal Formation", *The Rat Nervous System*, G. Paxinos ed., San Diego, CA: Academic Press, Inc, 1995.

[19] F. Schmitt, B.A. Barsky, and W. Du, "An adaptive subdivision method for surface fitting from sampled data," *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, New York, 1986, pp. 179-188.

[20] F. Schmitt, X. Chen, W. Du, and F. Sair, "Adaptive G1 approximation of range data using triangular patches," in P.J. Laurent, A. Le Mehaute, and L.L. Schumaker, editors, *Curves and Surfaces,* Academic Press, 1991.

[21] S. Steinberg and P. Roache, "Variational curve and surface generation," *Journal of Computational Physics*, vol. 100, pp. 163-178, 1992.

[22] J. Thompson, ed. *Numerical Grid Generation.* New York, NY: North-Holland, 1982.

[23] Toga, A, "Three-dimensional reconstruction," *Three-Dimensional Neuroimaging*, New York, NY: Raven Press, 1990 .

[24] Z. Warsi, "Basic differential models for coordinate generation," in *Numerical Grid Generation*, J. Thompson, ed., pp. 41-78. New York, NY: North-Holland, 1982.

[25] N. Weatherill, ed. *Grid Generation.* Lecture series, von Karman Institute for Fluid Dynamics, 1994.

[26] R. Weinstock, *Calculus of Variations.* New York, NY: Dover Publications, 1974.

VITA

David Batte received a B.S. in Computer Science from Southwest Texas State University in 1992. He worked on his bachelors for four years.

He is currently a graduate student in the Computer Science Department at Texas A&M University, and is working for an M.S. degree since January 1993. He is also a graduate student researcher at the Texas Transportation Institute at Texas A&M University. His duties include research and development of software systems for the visualization of transportation systems.

He can be reached at the following address: 1011 Pin Oak Drive, Belton, TX 76513, USA.