MORPHOLOGICAL MODELING OF NEURONS

A Thesis

by

KISHORE MULCHANDANI

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

MASTER OF SCIENCE

August 1995

Major Subject: Computer Science

MORPHOLOGICAL MODELING OF NEURONS

A Thesis

by

KISHORE MULCHANDANI

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

MASTER OF SCIENCE

Approved as to style and content by:

Bruce H. McCormick
(Chair of Committee)

Donald K. Friesen
(Member)

Jianer Chen
(Member)

Donald H. House
(Member)

Richard A. Volz
(Head of Department)

August 1995

Major Subject: Computer Science

ABSTRACT

Morphological Modeling of Neurons. (August 1995)

Kishore Mulchandani, B.S., University of Roorkee

Chair of Advisory Committee: Dr. Bruce H. McCormick

A formal representation of neuron morphology, adequate for the geometric modeling of manually-traced neurons, is presented. The concept of a stochastic L-system is then introduced and the critical distribution functions governing the stochastic generation of dendritic and axonal trees are defined. Experiments with various stochastic L-system models for pyramidal, motoneuron, and Purkinje cells are reported which generate synthetic neurons with promising proximity to neurons in the neurobiology literature. A neuron morphology modeler is designed and implemented which provides a methodology for iterative refinement of the stochastic L-system model.

ACKNOWLEDGMENTS

I thank Dr. Bruce H. McCormick for introducing me to the L-systems, my colleagues Sandeep Tewari, David Batte and Gregory Schmidt for their suggestions, and Brent Burton, for his help in generating images of synthetic neurons. I also thank my sister Lakshmi and her husband R.K. Gianey, my uncle Mr. Vincent Emmanuel, my cousin Dr. Chandra P. Belani, and my friend Brijesh Srivastava for their financial and moral support during the course of my study. And a very special thanks to a special friend, Kristi, for her help and encouragement in some of my stressful moments. Finally, I thank my parents for making me what I am today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

L-systems were first proposed by Aristid Lindenmayer in 1968 as a mathematical theory to model the growth and morphology of simple plants [12]. Turtle geometric interpretation of the strings generated by the L-system grammar allows the geometric modeling of plants. Today L-systems are an accepted tool in plant modeling[12]. Neurons, scaled by 10^5, show a strong resemblance to trees. This similarity led us to harness the power of L-systems to produce geometric models of neurons. Though neurons and glial cells, unlike plants, are composed of a single cell, analogous formal L-system methods are described below to generate neuron-like structures using separate subgrammars for the dendritic and axonal processes.

All models in this work are presently limited to modeling neuron morphologies at the limit of optical resolution. In particular spines are not presently included, except in a superficial way. Furthermore, fiber tracts and fiber terminations on cells are not presently modeled.

A. Objective

1. Generation of a Representative Neuron Population

Our primary objective is to generate a representative population of several neuron types (pyramidal, motoneuron, and Purkinje cells) from stochastic L-system models of these neuron morphologies. The use of stochastic L-systems is mandated due to the non-deterministic nature of the neural morphology. No two neurons, even of same type, are identical; yet they are sufficiently similar to be classified as being

---

The journal model is *IEEE Transactions on Automatic Control.*

co-members of the same type set. This property points to the existence of a common family of rules, or grammar, where the rules are chosen stochastically.

2. Validation of Neuron Morphology

Once visually similar neuron morphologies are generated, a system is needed which can statistically validate these morphologies against available databases of manually-traced cells. Morphological comparison requires that the distributions of the actual and synthetic neurons are compared for differences. These differences, in turn, must be narrowed by choice of the L-grammar and its stochastic distribution functions to enhance the credibility of these L-system models.

3. Database for Neuron Morphology

Both acquisition of manually-traced neurons and multiple modeling simulations of an L-grammar quickly generate sufficient morphological data to require a database management system. We provide an object-oriented schema for representing neuron morphology and a data repository for storing morphological information derived from both manually-traced and synthesized neurons.

4. Environment of the Embedded Neuron Population

There are two reasons why the neuron morphology cannot be modeled independent of the environment.

Firstly, boundary of the neural tissue poses a major problem. If the size, the shape and the relative arrangement of the neurons of an unconvoluted tissue are considered the norm, then it must be possible to correlate these three properties with those in a curved part of that tissue [14].

Secondly, in a developing nervous system, specific nerve connections require that the nerve growth make appropriate orientation responses during its pathfinding [3]. The turning of the growth cone can be influenced by a number of physical and chemical factors in the extracellular environment, including chemotactic substances, mechanical barriers, substratum adhesiveness, and selective affinity for nearby cells [10].

Any manually-traced neurons would have to be corrected for these environmental influences to obtain a neuron morphology suitable for modeling in a uniform logical space. And vice versa, any morphologies generated for a uniform logical space would have to be mapped into the generally non-uniform physical space. We suggest a solution to this problem which involves the use of three-dimensional grids and mapping these grids between the uniform logical space and the physical curvilinear space.

B. Motivation

The applications of these L-system models are severalfold. First, the L-system morphological models bring coherence to existing databases of neurons. They allow us to estimate the new information contributed by additional measurements of the neuron morphology.

Secondly, neurons are packed in a volume-filling manner into neural tissue, posing a barrier to visualization and interpretation of neural tissue scanned at the limit of optical resolution. New developments in confocal microscopy and multi-terabyte mass storage offer the potential for massively parallel tracing of fibers and other neuron processes in neural tissue, even at the scale of an entire rat brain. The interpretation of these massive volumetric data sets will rest on having creditable models of neuron morphology. Furthermore, because of the close juxtaposition of neurons, existing volume visualization methods, such as the marching cubes algorithm or the Levoy algorithm, would portray only a dense and confused volume. Alternatively, what is needed are object-oriented models for these volumetric databases of neural circuitry.

Finally, volume-filling tissue, because of interference between neighboring cell processes, can not be assembled from a collection of independently-generated neurons and glial cells. We believe the only creditable way to graphically generate morphological models of brain tissue is to mimic the growth processes of developmental neurobiology.

CHAPTER II

REPRESENTATION OF NEURON MORPHOLOGY

A. Terminology for Neuron Morphology

The constituents of a neuron are its soma, one or more dendritic arbors and its axon or axonal arbor (Figure 1).

Given a dendritic arbor, three types of nodes are identified: (1) the *soma node* representing the point of attachment of the arbor to the soma; (2) *branch points* at which the dendrite branches; and (3) *terminal nodes*, the endpoints of the arbor. Starting at the soma node a unique path exists to any other node of the dendritic arbor.

A *dendritic segment* is the portion between two consecutive nodes in the dendritic arbor. The *stem segment* has its origin at the soma node and is counted as the first order segment. Daughter segments arising from the first order branch point are called second order segments, and so on. A segment ending in a terminal node is called a *terminal segment*; all others are called *non-terminal segments*.

The *junction* at a branch point is called a *bifurcation* if the parent segment branches into two daughter segments, and a *multifurcation* if the junction gives rise to more than two daughter segments.

Axonal processes, including axonal arbors if present, are described by nodes, segments, and junctions in like manner, as discussed above for dendritic arbors.

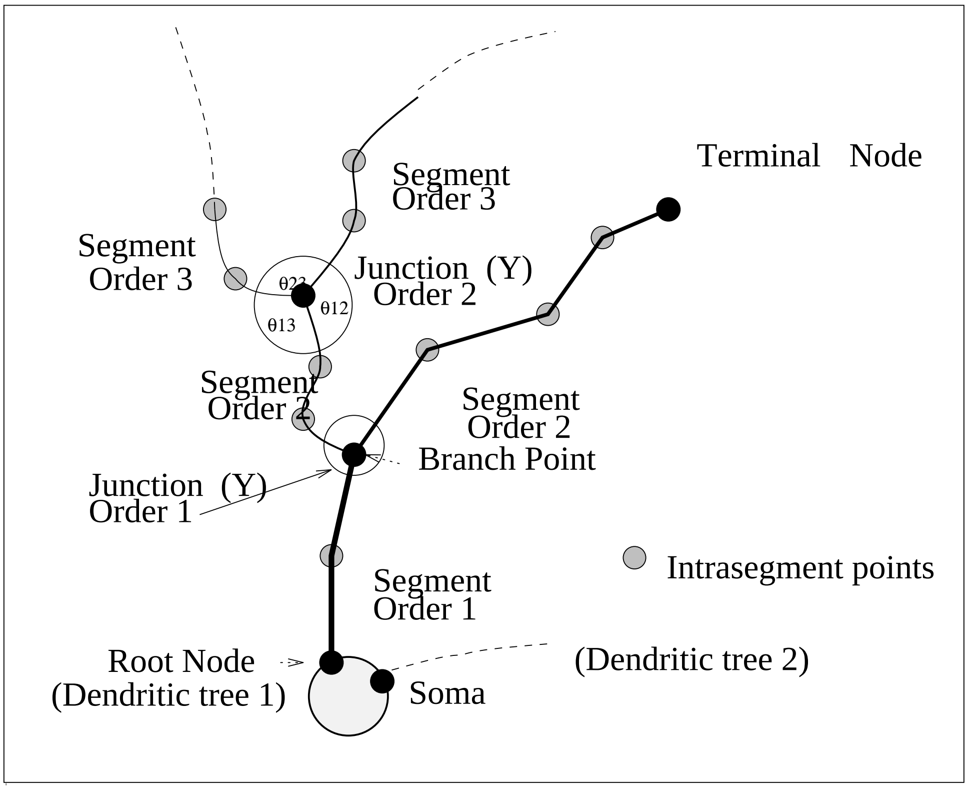


Fig. 1. Dendritic arbor representation

B. Representation of Segments

1. Segments are Represented as Generalized Cylinders

Segments are represented by generalized cylinders (Figure 2) whose central trajectory spans between the initial and final nodes of the segment. The length of the segment is defined as the integrated arc length of its trajectory between these two nodes. The diameter of the segment is in general a function of position along the trajectory. Two simple models are to assume (1) constant diameter, or (2) a tapering diameter characterized by the initial and final diameters.

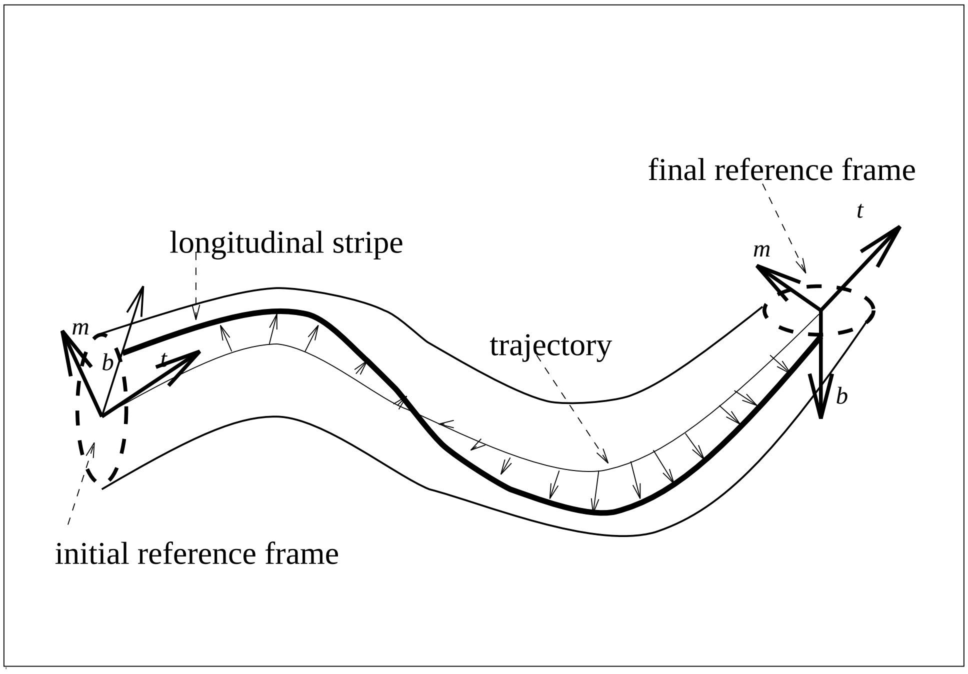


Fig. 2. Segment representation

2. The Segment Trajectory Defines a Local Reference Frame

Segments may be visualized as tubes with a fibrous internal structure (e.g., microtubules) which, like a plant stem, may impart stiffness against twisting. To accommodate this contingency we define a local coordinate system, or frame, with origin at any point $\mathbf{P}$ along its trajectory. The first axis is selected to lie along the unit *tangent vector* $\mathbf{t}$ at a point $\mathbf{P}$; the other two axes lie in the cross-sectional plane at $\mathbf{P}$, whose normal is $\mathbf{t}$. The second axis with unit vector $\mathbf{b}$, called the *binormal vector*, is chosen in this cross-sectional plane to reflect the orientation of the segment, or tube, about its tangent vector. (We can think of the $\mathbf{b}$ vector, as it sweeps along the trajectory, as "painting" a longitudinal stripe along the tube where it pierces the tube.) The third orthonormal vector of the right-handed coordinate system, $\mathbf{m}$, called the *main normal vector*, is given by $\mathbf{m} = \mathbf{b} \wedge \mathbf{t}$, where "$\wedge$" represents a cross product.

For dendritic or axonal segments, how does one define a local frame along the segment trajectory in some natural way from the available information, in the absence of a longitudinal stripe or access to the internal distribution of microtubules? The answer is to evoke a special local cartesian coordinate system, called the *Frenet frame*, linked to a point $\mathbf{x}(s)$ on the segment trajectory, as parameterized by s [7]. The Frenet frame is defined by the triplet $(\mathbf{t}\ \mathbf{m}\ \mathbf{b})$ of orthonormal vectors, where we retain the same names for these vectors as above; in particular $\mathbf{t}$ is the tangent vector (Figure 3). These vectors are now computed from the intrinsic geometry of the trajectory:

$$\mathbf{t} = \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}, \ \mathbf{b} = \frac{\dot{\mathbf{x}} \wedge \ddot{\mathbf{x}}}{\|\dot{\mathbf{x}} \wedge \ddot{\mathbf{x}}\|}, \ \mathbf{m} = \mathbf{b} \wedge \mathbf{t}a \tag{2.1}$$

where ".": means derivative with respect to s, and all required derivatives are assumed to exist. The binormal $\mathbf{b}$, computed from the local curvature, serves to define the "longitudinal stripe" of our illustration. The geometric interpretation of these vectors is straightforward. The vector $\mathbf{t}$ points along the trajectory at $\mathbf{x}(s)$; $\mathbf{b}$ is the normal to the plane containing the osculating circle which passes through the point $\mathbf{x}(s)$; and $\mathbf{m}$ points to the center of the osculating circle. For curves traditionally used in computer graphics, such as Bezier curves, the Frenet frame is readily computable, as also are the curvature κ and torsion τ at $\mathbf{x}$(s).

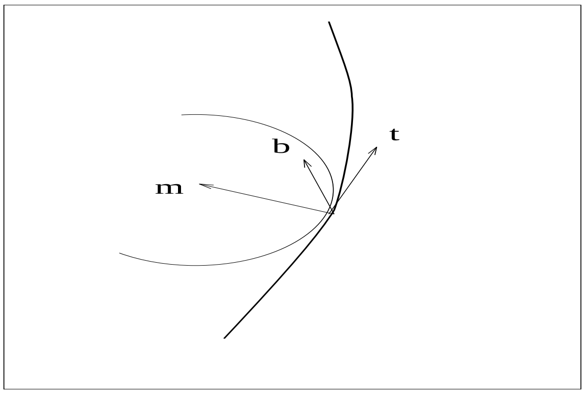


Fig. 3. The Frenet frame in 3D

In summary the reference frame is defined along the segment trajectory. Its computation requires specification and parameterization of the segment trajectory but makes no reference to the additional information necessary to specify the generalized cylinder of the segment.

3. Segment Placement in the Dendritic or Axonal Arbor

The dendritic (axonal) arbor is fully characterized by its segments. A segment, in turn, is represented by (1) its *trajectory*, (2) the additional information to specify its *generalized cylinder*, and finally (3) its *placement* in the dendritic (axonal) arbor. For the trajectory this information consists of the initial and final nodes, the trajectory between these nodes and its arc length, the initial and final reference frames, and the integrated torsion, or twist, of the local reference frame as it traverses the trajectory. The additional information to specify the generalized cylinder consists of the diameter, which may taper along the segment, and the distribution of spines and boutons, if observed, on the surface of the generalized cylinder. Finally arbor placement information consists of segment order (stem or first, second, ...), and whether the segment is terminal or non-terminal, as characterized by the final node of the segment.

C. Representation of Junctions

1. Geometry of the Bifurcation Model

A generic bifurcation is exhibited in Figure 4. The trajectories of the parent segment $\mathcal{S}_0$ and the two daughter segments, $\mathcal{S}_1$ and $\mathcal{S}_2$, meet at a common point called the branch point of the junction. In Figure 4 these segments are shown truncated. The length and diameter of these stubs, however, is irrelevant, as all significant properties of the junction are extrapolated to the branch point.[1] The order of a junction is defined to be that of its parent segment $\mathcal{S}_0$.

Conceptually the bifurcation can be modeled as a physical piece of plumbing, a "Y"-joint, into which three segment "pipes" $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{S}_2$ can be screwed. By screwing in the pipes and tightening, each pipe is given a terminal orientation about its axis, defined by the binormal vector $\mathbf{b}_i$ (i=0,1,2) of the segment at the branch point. Apart from these three terminal segment twists, we can give an invariant geometric description of the junction as a rigid body by specifying the angles θ_{01}, θ_{02}, and θ_{12} between the three segment tangent vectors and its chirility C (-1, 0, 1). Bifurcations with opposite chirility are mirror reflections of one another. Specifically, if the segments $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{S}_2$ are modeled respectively by the wrist, thumb and first finger of the right hand, then the chirility is 1; if $\mathcal{S}_0$, $\mathcal{S}_1$, $\mathcal{S}_2$ lie in a common plane, the chirility is 0;, and if left-handed, -1 (Figure 5).

It proves convenient to express the tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ of the daughter segments in spherical coordinates (θ_1,ϕ_1) and (θ_2,ϕ_2) respectively in the reference

---

[1]Mathematically speaking, the three trajectories characterizing $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{S}_2$ could fail to extrapolate to a common branch point. We ignore this source of error as being insignificant in practice

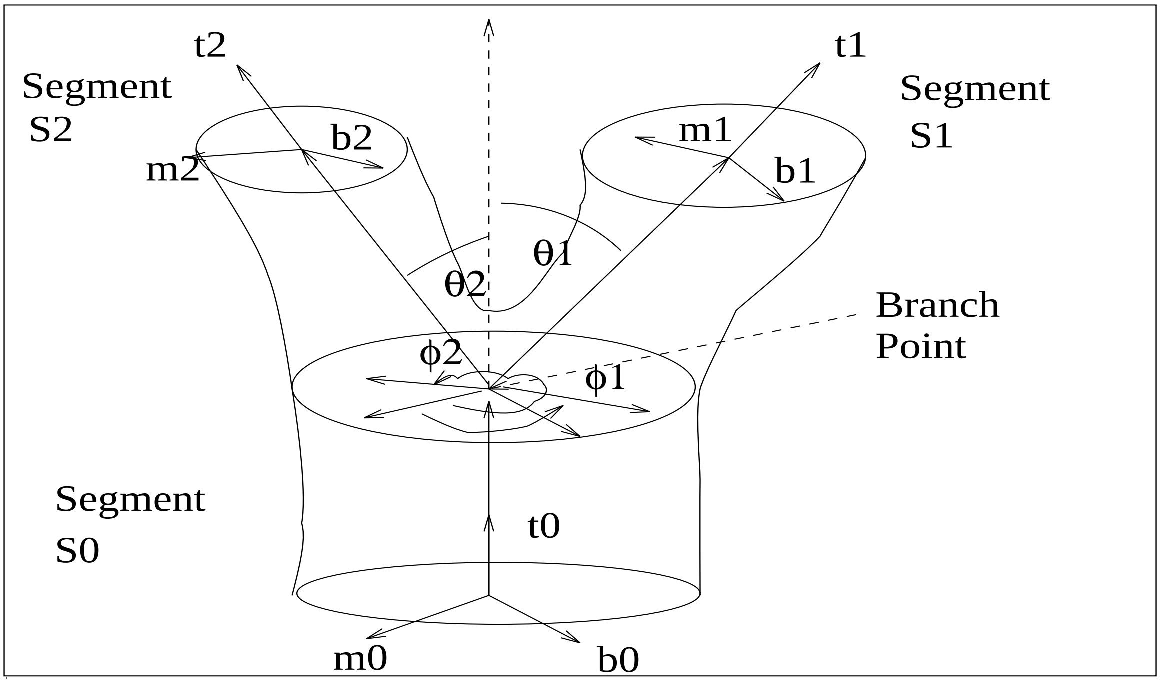


Fig. 4. The generic bifurcation and exit angles

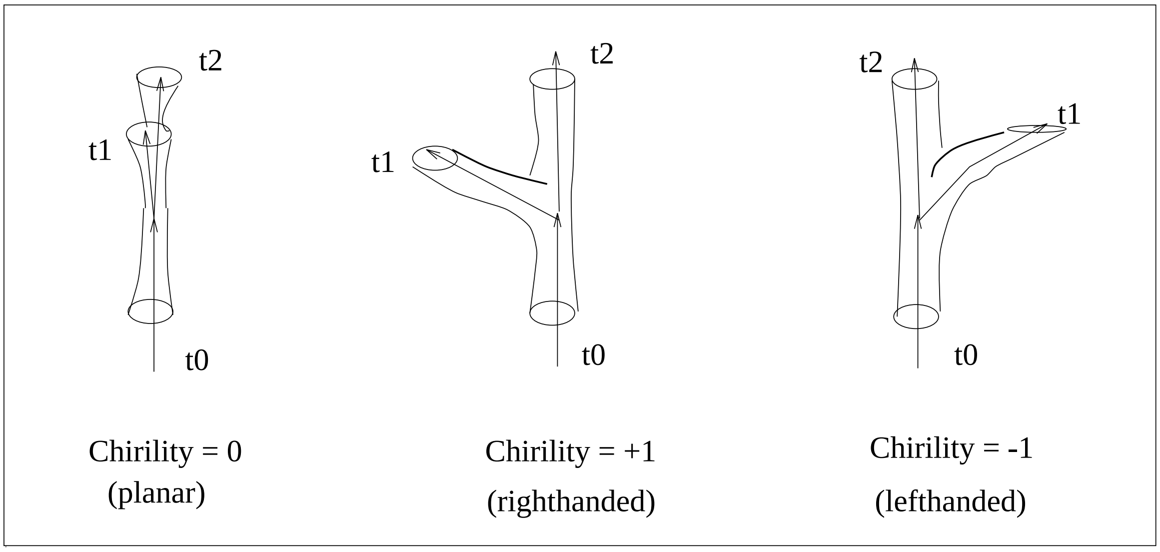


Fig. 5. Chirility

frame defined by the parent segment, with polar axis $\mathbf{t}_1$(Figure 4):

$$\mathbf{t}_0 \quad = \quad (0,0,1) \tag{2.2}$$

$$\mathbf{t}_i \quad = \quad (\sin\theta_i \cos\phi_i, \sin\theta_i \sin\phi_i, \cos\theta_i) \text{ for i=1, 2} \tag{2.3}$$

Hence we derive

$$\cos\theta_{12} = \mathbf{t}_1.\mathbf{t}_2 = \cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2 \cos(\phi_1 - \phi_2) \tag{2.4}$$

and observe

$$C = -\mathrm{sgn}(\mathbf{t}_0.(\mathbf{t}_1 \wedge \mathbf{t}_2)) = \mathrm{sgn}(\sin\theta_1 \sin\theta_2 \sin(\phi_1 - \phi_2)) \tag{2.5}$$

is uniquely defined by the invariant geometry of the junction.

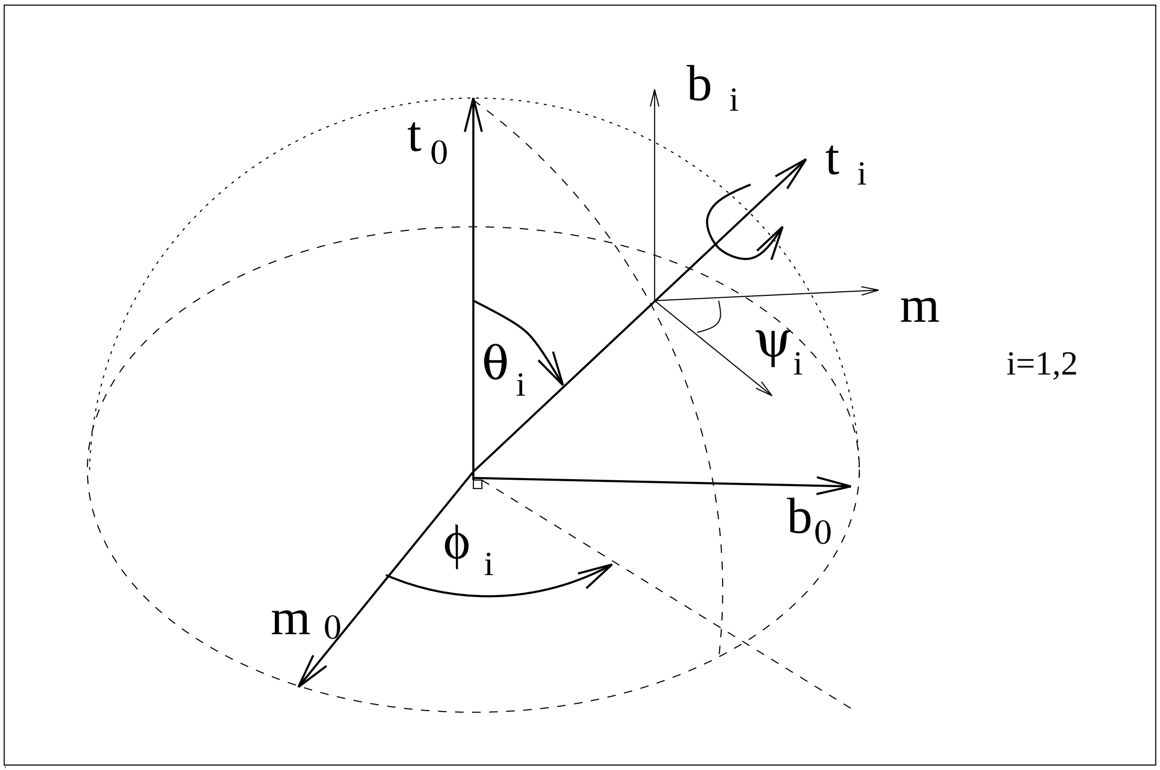


Fig. 6. Spherical coordinates (ϕ_i,θ_i) and twist (ψ_i) describing relative orientation of the reference frame for segment $\mathcal{S}_i$ (i = 1, 2)

Recapping, the bifurcation is fully specified by two 3-dimensional rotations, $\mathcal{R}_1$ and $\mathcal{R}_2$, about the branch point which transform the final reference frame of parent segment $\mathcal{S}_0$ into the initial reference frame of daughter segment $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively. In particular, these rotations specify how the tangent vector $\mathbf{t}_0$ of the parent segment at the branch point is transformed in the initial segment tangents $\mathbf{t}_1$ and $\mathbf{t}_2$ respectively. Segments, however, also have thickness and internal structure (e.g., microtubular structure) which may define an intrinsic orientation of the segment about its tangent. Accordingly the rotation $\mathcal{R}_i$ (i = 1, 2) specifies (1) the rotation of $\mathbf{t}_0$ into $\mathbf{t}_i$, and (2) the final 2-D rotation (twist ψ_i) about $\mathbf{t}_i$ which sets up the initial alignment of the daughter segment $\mathcal{S}_i$ (Figure 4).

Formally, let the rotation $\mathcal{R}_i(\phi_i, \theta_i, \psi_i)$ about the branch point transform the reference frame $F_0(\mathbf{t}_0, \mathbf{m}_0, \mathbf{b}_0)$ of the parent segment into the reference frame $F_i(\mathbf{t}_i,$ $\mathbf{m}_i$, $\mathbf{b}_i)$ (i = 1, 2) of daughter segment $\mathcal{S}_i$. The rotation can be expressed as the product of three successive 2-D rotations:

$$\mathcal{R}_i(\phi_i, \theta_i, \psi_i) = \mathcal{R}_t(\psi_i)\mathcal{R}_b(\theta_i)\mathcal{R}_t(\phi_i) \tag{2.6}$$

where ϕ is the azimuthal rotation about $\mathbf{t}_0$, and θ is the latitudinal rotation about $(\mathbf{t}_0 \wedge \mathbf{t}_i)/\|\mathbf{t}_0 \wedge \mathbf{t}_i\|$, the rotated $\mathbf{b}_0$ binormal. These two rotations carry $\mathbf{t}_0$ into $\mathbf{t}_i$. Finally ψ_i is the rotation (twist) about the $\mathbf{t}_i$ axis which defines the initial orientation of segment $\mathcal{S}_i$. If $\mathcal{S}_i$ has a straight or piecewise linear trajectory this final "twist" about $\mathbf{t}_i$ is not needed, and $\psi_i = 0$.

A single unit quaternion $\mathcal{Q}_\rangle$ can be substituted for the three rotations. (See Appendix B for details on Quaternions).

2. Extension to Multifurcations

Multifurcations, those junctions in which more than two daughter segments participate, are relatively rare. For example, almost all junctions in motoneurons of the rat are bifurcations, while trifurcations account for 1% [16].

We observed above that the rotation $\mathcal{R}_i(\phi_i, \theta_i, \psi_i)$ to rotate from the parent reference frame $\mathcal{F}_0$ to $\mathcal{F}_i$, the reference frame of the daughter segment $\mathcal{S}_i$, could be described as the product of three successive axial rotations. We retain the same convention for multifurcations, though the number of daughter segments is now greater than two.

3. Diameter Dependence and Rall's Ratio

By convention we designate segment $\mathcal{S}_1$ to be the daughter segment of largest diameter. Thus $\mathcal{S}_1$ serves as the principal limb emerging from the parent segment at the junction.

Rall has predicted on the basis of a passive electrical model of dendrite segments that R, called Rall's ratio, given by $(\Sigma(\mathcal{D}^{3/2}_{daughter}))/(\mathcal{D}^{3/2}_{parent})$ should be 1 [15]. For motoneurons R is approximately one [16]. We assume a default value of R = 1 for Rall's ratio in the absence of direct evidence from manually-traced neurons.

D. Representation of Somata

For synthetic neurons, the soma can be modeled by an ellipsoid or a sphere, or for pyramidal cells by a truncated cone. These axially symmetric models require two form factors [16]:

- *soma cross-sectional area* ($\mathcal{SCA}_s$) - from 2D projection onto the sagittal plane, and

- *soma diameter ($\mathcal{D}_s$)* - the diameter of a circle with the same cross sectional area.

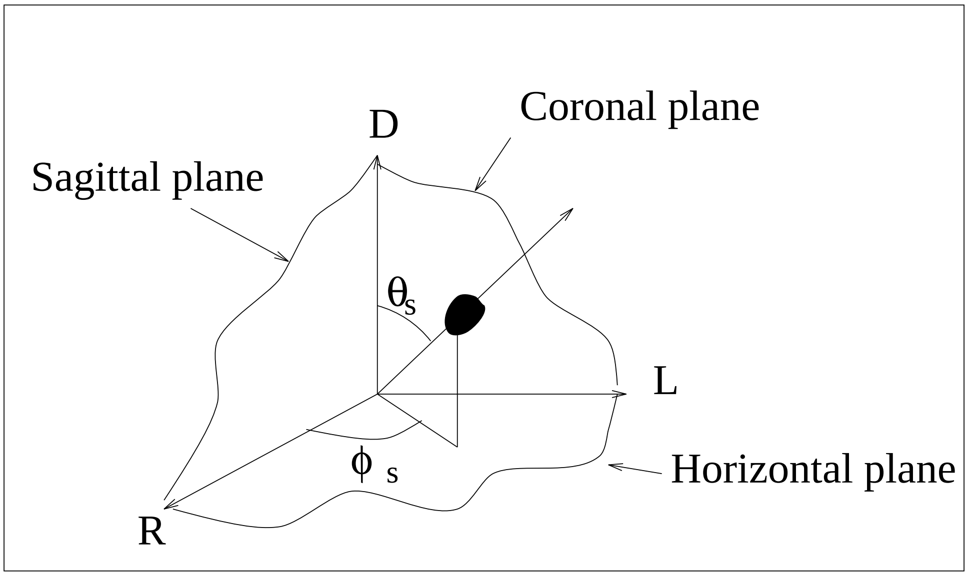


Fig. 7. Position and relative orientation of soma

The soma, as a rigid body, requires a soma position and orientation relative to the rostral (R), lateral (L), and dorsal (D) anatomical axes of the nucleus (Figure 7), or for cortical neurons, relative to the local cortical surface normal.

At higher resolution, and for the bulbous somata of the autonomic nervous system, we model somata by conventional 3D reconstruction [6]. Somata, of course, can be viewed as particularly plump segments and like segments, represented as generalized cylinders about a central trajectory. We suggest that this trajectory initiate at the axon hillock and pass through the centrum of the soma. However, so few soma have been reconstructed in 3D that we are unable to give further guidance at this time.

E. Representation of the Environment

The physical embedding of the neuron may be regarded as a curved space having its own curvilinear coordinate system and variables x, y, and z as its coordinates. On the other hand, we use variables ξ, η, and ζ as the coordinates of the uniform logical space. The coordinate lines in the logical space are the images of the coordinate lines in the physical space and are thus given by the curves where one of the ξ, η, or ζ varies and the other two are held constant [11](Figure 8).

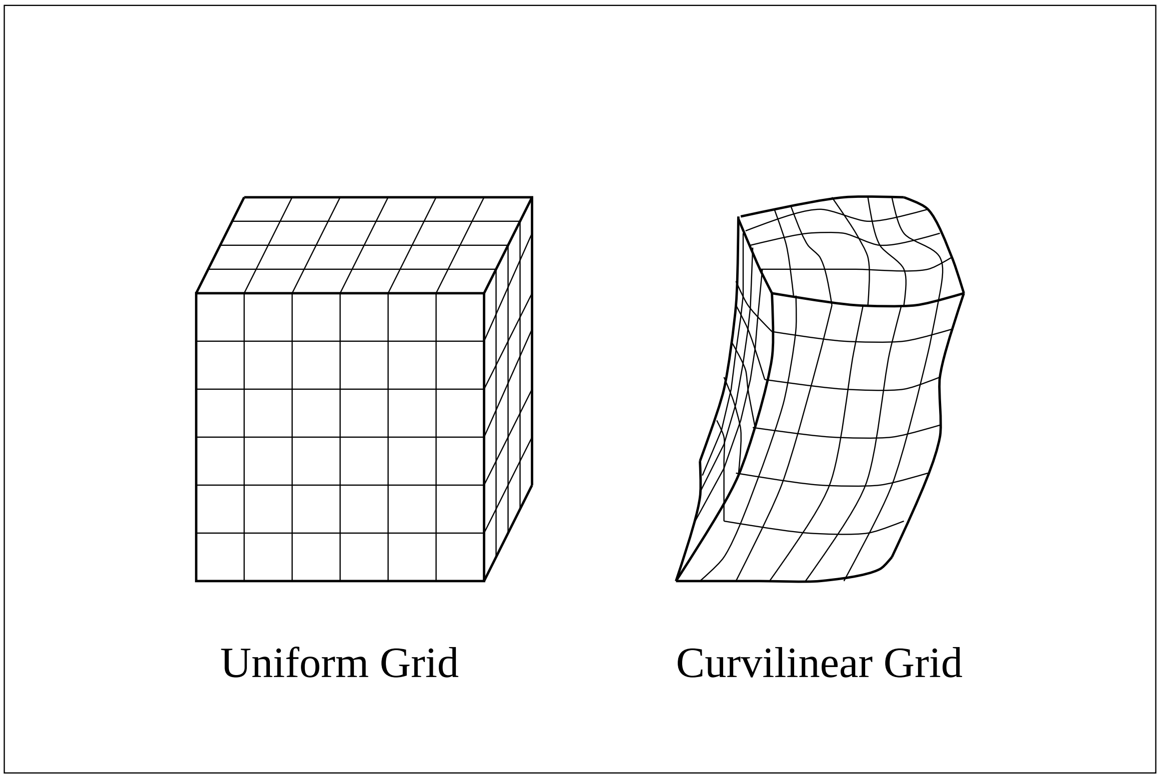


Fig. 8. A uniform grid and a corresponding curvilinear grid in 3D

An advantage of having a uniform grid superimposed in the model space is the simplicity it grants in the specification of positional forces. A neuron in tissue grows under the influence of various physical and chemical forces. These forces could be a result of a mechanical barrier or a chemical gradient. To specify these forces accurately one needs a co-ordinate system. Grids provide such a system. Mechanical barriers and chemical concentration can then be placed at desired positions relative to the neuron in the logical space. The gradient of the chemical concentration potential may then be calculated analytically from their implicit functions described in variables ξ, η, and ζ.

We provide models of growing neuron morphologies in this uniform logical (ξ, η, ζ) space. These morphologies can then be transformed into any physical space using the inverse mapping functions that map the cells of this physical space into the cells of the logical space.

CHAPTER III

L-SYSTEM MODELING

A. L-system Grammars Can Generate Strings of Parenthesized Terminal Symbols

A grammar consists of a partitioned set of symbols (*non-terminal* symbols, including the *start* symbol, and *terminal* symbols) and a set of replacement rules called *productions*. In the context-free grammars considered here, the lefthand side of a production designates a non-terminal symbol: its righthand side designates a finite string of symbols, optionally both terminal and non-terminal. To apply a production in an L-grammar, all non-terminal symbols in the string generated so far which match the non-terminal symbol of the left hand side of the production are replaced *in parallel* by the right-hand-side of the production. Paired brackets, as terminal symbols, may be embedded in the string recursively. Productions are applied until the string so generated contains only terminal symbols.

It is convenient to map a string generated by the L-system into an equivalent tree format. A bracketed list of terminal symbols, where the brackets are paired, can be mapped into a finite tree as follows: the root of the tree is labeled by the start symbol, each non-terminal node of the tree corresponds to a (bracketed) list of terminal symbols, and finally terminal nodes are labeled by terminal symbols of the string while preserving their order in the string.

B. Turtle Rotations are Defined in the Local Reference Frame

The terminal symbols of the L-grammar are interpreted graphically as commands for turtle reorientation and movement. These symbols instruct the turtle to trace a path in the 3-dimensional space. The term “turtle” is traditionally used as the turtle conveys not only a position in space, but also its orientation; that is, the turtle defines a local frame of reference (**t**, **m**, **b**) which we shall identify with the reference frame.

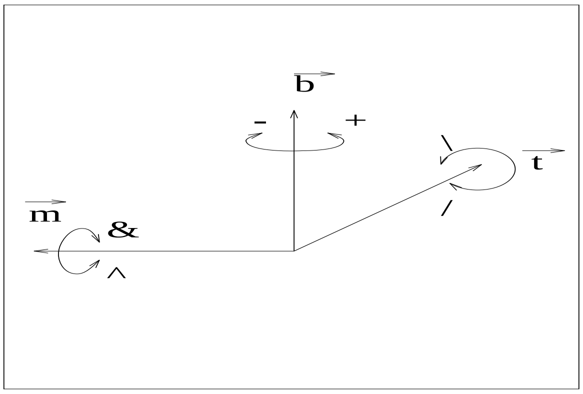


Fig. 9. Rotations of the reference frame

Six terminal symbols are reserved to signify rotations of the turtle in its local reference frame (Figure 9). These are ("\","/") for tangential rotations about **t**; ("&", "∧") for rotations about main normal vector **m**; and ("+", " -") for rotations about the binormal **b**.

C. The Reference Frame Can be Generalized to Piecewise Linear Trajectories

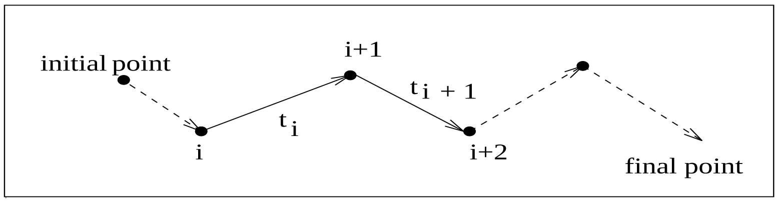


Fig. 10. Piecewise linear trajectory

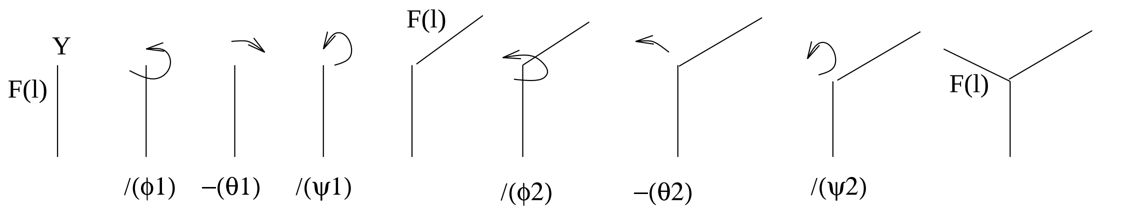
Piecewise linear trajectories occur as approximations to segment trajectories and as describing the path between two nodes in the dendritic or axonal arbor. For piecewise linear segment trajectories (Figure 10), the initial and final points of the trajectory are called external knots: all other knots of the piecewise linear curve are called internal knots. The tangent vector $\mathbf{t}$ is defined as a piecewise constant, which changes discontinuously at internal knots. But $\mathbf{b}$, and hence $\mathbf{m}$, are not computable from Eqn 2.1. Therefore we introduce the following generalization of the reference frame for piecewise linear trajectories. At an internal knot define $\mathbf{b}$ and $\mathbf{m}$ as follows

$$\mathbf{b}_i = \frac{\mathbf{t}_{i-1} \wedge \mathbf{t}_i}{\|\mathbf{t}_{i-1} \wedge \mathbf{t}_i\|}, \;\; \mathbf{m}_i = \mathbf{b}_i \wedge \mathbf{t}_i \tag{3.1}$$

Then define the generalized reference frame along the trajectory as a piecewise constant function: the reference frame from the initial point is translated without rotation to the second knot point, at this knot $\mathbf{t}$, $\mathbf{m}$ and $\mathbf{b}$ are computed by Eqn 3.1 and this reference frame propagated to the next knot, and so on. The final node carries the reference frame computed at its next-to-last knot. For straight trajectories and their generalization, piecewise linear trajectories, the initial orientation of the binormal vector $\mathbf{b}$ cannot be computed from the segment trajectory alone but requires reference to the branch point and parent segment from which the daughter segment arose. We shall define $\mathbf{b}$ to be normal to the plane defined by these two segments. Finally for a stem segment having a piecewise linear trajectory, we assign an initial orientation to the $\mathbf{b}$ vector derived from the coordinate system used to describe the soma.

D. L-systems Can Generate Geometric Models

An L-system grammar generates a parenthesized string of terminal symbols. Each terminal symbol serves to orient or generate a geometric model of a constituent part of the organism. For modeling dendritic or axonal arbors, the principal constituent parts are segments, which are realized as generalized cylinders and may be of several classes, and junctions, from which grow subarbors. Figure 11 illustrates the generation of a




```
S(n,l,d0) -> !(d0)F(l)Y(n,  ϕ1,θ1,ψ1,ϕ2,θ2,ψ2, d1,d2)
Y(n,ϕ1,θ1,ψ1,ϕ2,θ2,ψ2, d1,d2) -> [ /(ϕ1)−(θ1) /(ψ1) S(n+1,l1,d1)][ /(ϕ2) −(θ2)/(ψ2) S(n+1,l2,d2)]
```

Fig. 11. A simple grammar that generates a junction

simple junction by an L-system. In this simple grammar, the segments are straight black lines.

E. Depth-first Generation of the Dendritic Arbor

Terminal symbols which represent segments are realized as generalized cylinders swept out as the turtle traverses the trajectory of the segment.

At terminal symbols which represent junctions (signalled by a "[" symbol in the string representation; and equivalently by a non-terminal node in its arbor representation) the turtle is reoriented as specified by the first of the list of rotations specifying the junction. The daughter segment $\mathcal{S}_1$, and its associated subarbors, are swept out in a depth-first traversal of the tree. The turtle, backtracking at each occurrence of a "]" symbol, finally returns to its prior orientation at the original junction, the reference frame of segment $\mathcal{S}_0$. Then the turtle begins anew to traverse in a similar manner the subarbors blossoming from $\mathcal{S}_2$ (and other segments, if a multifurcation). In this manner the turtle completes a depth-first traversal of the tree and by this process generates a geometric model of the neuron.

F. Stochastic L-systems Generate Populations of Neurons

A stochastic L-system proceeds as above, but the value of segment parameters and the rotation angles emerging from a branch point are now random variables drawn from appropriate distribution functions. These functions are invoked from the grammar rules when they are being applied during the generation phase. The results of these invocations are the values of the parameters that are used by the reorientation symbols, namely, "\", "/", "&", "∧", "+", "-", of the turtle geometry terminal symbol set.

In general these distribution functions will depend not only on neuron type, and possibly its depth in the cortex, but also on the type of dendritic arbor. For example apical and basal dendritic arbors of a pyramidal cell are generated by different subgrammars, and have different distribution functions controlling segment length and branch point exit angles.

CHAPTER IV

OBJECT MODEL FOR MORPHOLOGICAL DESCRIPTION

Our description of neuron morphology is intuitive and brings all the benefits of the object-oriented paradigm, which include portability, extensibility, and reusability.

The *Cell* class has as current attributes its physiological type, its parent organism, its position in the neural tissue, its sources and other text fields carrying information about the experiment and the investigator that carried out the study.

Some important attributes of the *Soma* class for geometric modeling at the level of optical resolution, are its shape, its volume and the number of first-order segments (axonal or dendritic) that arise from it. By measuring points along the soma cross-section borders and stacking up the resulting polygons, one can reconstruct in 3D the soma shape and also compute its volume. The accuracy and precision of these soma attributes are a function of the number of sections used. Though the soma may also be viewed morphologically as a special node or a special multifurcation junction, we have kept it as a separate class in recognition of the special role the soma has in neuron function.

Dendritic or axonal arbors define three other classes: *Segment* and *Junction*, whose instances are interlinked at instances of the *Node* class to form the overall neuron morphology. Each segment is represented as a generalized cylinder by a sequence of points along its trajectory and the diameters at these points. A segment can be a terminal segment or it can bifurcate(multifurcate) to produce a junction and two or more daughter segments. The junction owes its existence to three (or more) segments which share the same branch node. These segments can be identified as the base segment, the primary, secondary (and possibly tertiary) segments. The intermediate angles between the segments are also stored as attributes of the junction.

To facilitate statistical analysis of a sub-arbor of the dendritic or axonal arbors, we provide an *Arbor* class. This class currently stores attributes such as a pointer to its root segment, and its type (dendritic or axonal).

Spines are also known to have different shapes depending on the different synaptic properties. The shape of a spine is too small to distinguish at the current limit of optical resolution and thus our *Spine* class is presently a virtually empty class. We have provided it as a spaceholder class.

Building on these basic classes, we have devised a seven class model (Cell, Soma, Arbor, Segment, Junction, Node, and Spine) which can be used as a canonical model for morphological measurements. The complete morphological information of a neuron is captured as the attributes and relationships of the object instances of these seven classes.

Appendix A provides complete descriptions of these classes in C++.

CHAPTER V

STOCHASTIC GENERATION OF NEURONS

A. Premises

In performing the stochastic generation of neurons, we make several simplifying hypotheses. These are:

- Each dendritic arbor develops independently.
- Segments develop independently, though they may exhibit tropism in the presence of a shared external force field or in conforming to a common nucleus boundary.
- Segment length is a random variable whose distribution function is both class and order dependent.
- Each class of segments of order n is assigned a probability of non-termination $\mathcal{P}_{nt}(\mathrm{n})$. A typical plot of $\mathcal{P}_{nt}(\mathrm{n})$, as derived using Eqn 5.1 and using the segment distribution observed in a motoneuron [16] is illustrated in Figure 12.

$$\mathcal{P}_{nt}(\mathrm{n}) = \text{No. of Segments of order n+1}/(2 * \text{No. of Segments of order n}) \tag{5.1}$$

- Bifurcations and multifurcations develop independently, but are conditional upon the class of the parent and daughter segments.
- The junction probability density function describing the distribution of branch point exit angles (and twists, if any) can be order dependent.

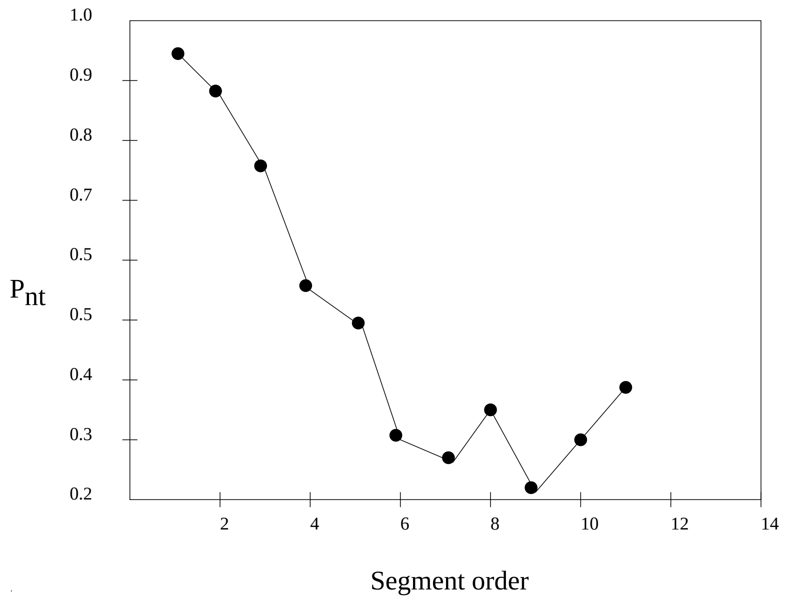


Fig. 12. Typical probability that segment is non-terminal vs segment order

- Both segment and junction distribution functions can depend on the position of the soma in the neural tissue, (e.g., pyramidal cells exhibit scale dependence upon their depth in the cortex).

B. Generation of Segments

The stochastic generation of segments attempts to simulate the growth process of the neurobiological segment but from a coarser grain perspective. We distinguish three models of the environment for this growth:

- *the isotropic model*, where in the absence of environmental forces we assume that segment growth is determined solely by its local reference frame;
- *the force field model*, where an external force field provides a tropism for growth. The apical dendritic arbors of pyramidal cells, for example, exhibit tropism and grow predominantly toward the cortical surface. Using context-sensitive,

parametric L-grammars, we can also generate neurons whose growth conforms to the surface boundary of an encompassing nucleus. The nucleus boundaries can be specified as an implicit function in the model space. The distance from the surface and the angle made with the surface can be used as parameters to reorient the direction of growth. These parameters are adjusted with each generation of rewriting, and so this mechanism provides a feedback to the L-system about whether or not a reorientation is needed for the next stage of growth [12].

- *the 3D grid model*, where a boundary-conforming grid is imposed throughout the body of the brain nucleus or architectonic area. The grid imposes, by interpolation, a curvilinear coordinate system throughout the neural tissue. Segment generation is then prejudiced to conform to this local coordinate system. For example, Purkinje cells develop as if conforming to planar glial trellises.

The force field model and the 3D grid model can be viewed as alternative statements of the same model. We sketch this correspondence briefly below. The force field $\mathrm{f}(\mathbf{x})$ is assumed acyclic, that is $\mathrm{curl}\ \mathrm{f}(\mathbf{x}) = 0$. The force f can then be derived as the gradient of a scalar potential Φ, that is $\mathrm{f} = -\nabla\Phi$. Now we design the potential Φ such that the nucleus boundary is an iso-surface of Φ, and hence $\nabla\Phi$ is the local outward-pointing surface normal. Setting $\mathrm{f} = -\nabla\Phi$ sets up a repulsive force field, which bends ramifying segments of the dendritic arbor to conform to the nucleus surface (e.g., the cortical surface).

C. Generation of Junctions

An individual junction has been represented, as described above, as a list of rotations, where each rotation in turn transforms the reference frame of the parent segment into the reference frame of a daughter segment. The stochastic generation of junctions (of given order and segment class) requires that we specify the conditional probability density for these concurrent frame rotations.

To illustrate the concept, consider a bifurcation for which the incident trajectory bifurcates ("scatters") into daughter trajectories $(\phi_1, \theta_1, \psi_1)$ for $\mathcal{S}_1$ and $(\phi_2, \theta_2, \psi_2)$ for $\mathcal{S}_2$. The differential probability for this event is

$$\mathcal{P}(\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2 | n, \mathcal{S}_1, \mathcal{S}_2)\mathrm{d}\omega_1\mathrm{d}\psi_1\mathrm{d}\omega_2\mathrm{d}\psi_2 \tag{5.2}$$

where $\mathrm{d}\omega_1$ and $\mathrm{d}\omega_2$ are the differential solid angles given by

$$\mathrm{d}\omega_i = \sin\theta_i \mathrm{d}\theta_i \mathrm{d}\phi_i \text{ for i=1, 2} \tag{5.3}$$

The observant reader will recognize that the statistical distribution of junctions, as required for their stochastic generation, directly parallels the S-matrix theory for particle scattering in theoretical nuclear physics [8]. In fact, if segments have an inherent orientation (as defined by their reference frame) then this additional complication plays a role comparable to polarization for particle beams (which also require the definition of a local reference frame for each beam). We shall defer further development of this correspondence between the junction distribution functions and the S-matrix formalism until subsequent papers.

D. Generation of Soma

A soma may be very coarsely approximated by an ellisoid, sphere or a cone. Depending on the level of accuracy desired, it may instead be reconstructed by stacking consecutive boundary contours and using traditional 3D contour reconstruction methods [6]. We do not provide any method for generating these contours in our synthetic model and leave it as a subject of future research. However, we do allow the soma to be reconstructed from any contour data that might become available and displayed in the 3D Neuron Morphology Visualizer as described later.

E. Generation of a Forest

Typically only a sparse "forest" consisting of 0.01-0.1% of the cells in a nucleus or architectonic area are displayed simultaneously, with the choice of cells to be displayed under investigator control. We offer different classes of non-uniform distributions of neurons in 3D space, such as the Poisson, the Poisson sphere and the jittered distributions. A 'seed' for a neuron may be planted in 3D space according to the appropriate distribution function. This allows an investigator to conduct studies with controlled variation in population densities of neurons.

An additional and particularly significant environmental influence must be built into the L-grammars when parallel growth of neurons is being considered. An important inviolable constraint is imposed by the fact that space is not a shared resource. So one must prevent processes of one neuron from encroaching into space occupied by some other during growth. We call this *the intersection problem.* Intersections can be avoided by building in feedback in the L-grammar about the current state of the string generated.

Each production must be applied conditionally, constrained that the result of the production will not result in the violation of the space rights. This would imply that the string generated at the end of each cycle of the string substitution would need to be interpreted geometrically, and any spatial conflicts would mandate a backtracking. Obviously this is a computationally expensive approach. The less expensive approach is to detect the spatial conflicts at the final string interpretation phase and reject any violators then. A variant of this approach has been used in [13] for growing boundary-conforming ornamental plants.

We suggest a similar method. Every new segment being generated is validated against the previously generated list of segments for spatial conflicts. If no conflicts are detected, the list is augmented with this validated new segment.

Table I. Arbor morphological variables

Description	Symbol	Formula/ Interpolation
Number of junctions	N_j(n)	N_j(n)
Prob. of non-termination	P_{nt}	$N_s(\text{n}+1)/(N_s(\text{n}) * 2)$

F. Morphological Variables and their Distributions

To model any junction stochastically, we need values of the number of daughter segments, Rall's ratio, the rotation angles θ, ϕ and ψ for each daughter segment, and the chirility.

For modeling a segment, we require its length, the initial and the final Frenet frames, the parameters (control points) of the B-Spline fit of the segment trajectory, and the least square fit. The computation of the Frenet frame of a segment at any point including the endpoints can be done from its curvature and torsion at those points. For computing the "jitter" in the segment trajectory, the Frenet Frame at all intermediate points on the segment may be required. The diameters at the end-points of the segment give its taper ratio. Diameters at all intermediate points provide the taper ratio per unit length.

A list of variables that determine the morphology of a neuron are listed in Ta-

Table II. Segment morphological variables

Description	Symbol	Formula/ Interpolation
Segment arc length	L_s	L_s
Segment mean diameter	D_s	D_s
Taper ratio of segment	TR	$D_{s_{final}}/D_{s_{initial}}$
Mean taper per unit length	TR_l	D_s(t+1)/Ds(t)
Curvature	κ	Angular velocity of **t**
Torsion	τ	Angular velocity of **b**
Rotation of the Frenet frame	Q	$Quat(FF_{init} \Rightarrow FF_{final})$
Successive linear rotation of Frenet frame	Q_i	$Quat(FF_i \Rightarrow FF_{i+1})$
Parameters of piecewise Bezier or B-spline cubic fit	G_i	Knot Point i

bles I, II and III. Each variable is associated with one class of our object model of neuron morphology as described earlier.

A paradigm shift to a new mode of analyzing neuron morphology is suggested. The developing bifurcations and multifurcations of growing dendritic and axonal arbors bear striking resemblance, in ultra-slow motion, to nuclear scattering events [4]. We recognize this resemblance, and can statistically describe these junction "events" by a variant of the S-matrix theory used in the formal theory of nuclear reactions. These scattering distribution functions, in turn, feed the stochastic L-system models. Conceptually, the block of embedded histological tissue replaces the bubble chamber of high energy physics.

Table III. Junction morphological variables

Description	Symbol	Formula/ Interpolation
Number of segments	N_s	N_s
Angle between segments i, j=0,1,2; i $\neq$ j	θ_{ij}	θ_{ij}
Roll angle between segments i=0, j=1,2	ϕ_j	azimuth angle
Turn angle between segments i=0, j=1,2	θ_j	polar angle
Twist angle between segments i=0, j=1,2	ψ_j	ψ_j
Steradian form factors	ω_i	$\sin\theta_i$
Segment length reduction daughter i	LR_i	L_{s_i} / L_s
Segment diameter reduction daughter i	DR_i	D_{s_i}(n+1)/D_s(n)
Rall's ratio	RR	$(\Sigma(\mathcal{D}^{3/2}_{daughter}))/(\mathcal{D}^{3/2}_{parent})$
Chirility	C	$\mathrm{sgn}(\sin\theta_1 \sin\theta_2 \sin(\phi_1 - \phi_2))$
Chirility factor	C_f	$\sin\theta_1 \sin\theta_2 \sin(\phi_1 - \phi_2)$

CHAPTER VI

GRID GENERATION AND CONFORMAL GROWTH OF SAMPLE NEURON POPULATIONS

A. Introduction

The L-grammar we use consists of terminal symbols that are commands for the turtle. The turtle can turn around its three principal axes and can thus point into the direction in which the primary, or secondary, segments need to be grown. The growth of the primary and secondary segments at the branch point involves calculation of the exit angles by which the turtle needs to turn, roll and then twist before sending off the turtle in this new orientation. Since neurons are unlike chemical molecules, these angles are distribution functions rather than single fixed values (delta function distributions). These angular distributions must be provided by the statistical analysis of the measured neuron morphology. For instance, through a statistical analysis it may be discovered that the branching angle between the primary and secondary segments is a normal distribution with a mean of 60 degrees and a standard deviation of 20 degrees for a given neuron type. Or it may be found that the primary and secondary segments have absolutely no idea of each others growth and bifurcate quite independently from the branch point. Mathematically speaking, their exit angles are uncorrelated, and their probability density function, $\mathcal{P}(\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2) = \mathcal{P}_1(\phi_1, \theta_1, \psi_1)\mathcal{P}_2(\phi_2, \theta_2, \psi_2)$, factors into independent probabilities of each daughter segment. These conclusions can be used in the stochastic L-grammars to provide values for turn, roll and twist of the turtle.

B. Pyramidal, Motoneuron, and Purkinje Cells

Pictures of different neuron types that were generated using the stochastic L-system synthesizer after estimates of the distribution functions were supplied, are shown in Figures 13, 14, 15.

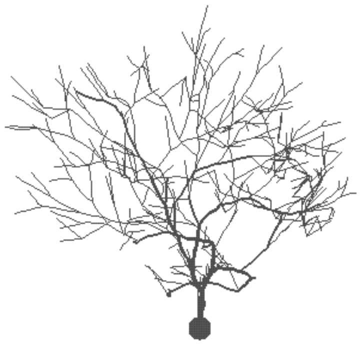


Fig. 13. A Purkinje neuron

Fig. 14. A Motorneuron

The current grammars assume normal distribution functions. The mean and standard deviation of the parameters are supplied through the grammar. For instance, if the the distribution of ϕ_1, ϕ_2, θ_1 and θ_2 are known, then the angle θ_{13} can be computed. However as the angles are extracted from a distribution their values

Fig. 15. A Pyramidal forest

have to be checked against geometrical constraints like $\theta_{01} + \theta_{02} + \theta_{12} \leq 2\pi$ and $\theta_{12} \geq \mathrm{abs}(\theta_1 - \theta_2)$.

The diameters of the daughter segments are calculated from the parent segment's diameter, the distribution estimate for the diameter of the primary segment and Rall's ratio set to 1.

A Purkinje neuron has a relatively planar architecture. Even though its individual segments are not in a perfect plane, the cell somehow maintains a mean planar character as if there were two invisible walls between which this cell is sandwiched. To control the individual twists from destroying this overall planar character and yet avoid perfectly planar dendritic arborization, the turtle is influenced by a kind of tropic effect. The current turtle direction is mirrored with respect to the normal of the plane of the Purkinje cell. This forces the segments of the alternate orders to return back to the plane and hence prevents a propagating deviation from destroying the planarity of the cell.

A typical rat motoneuron has from 6 to 12 first order dendritic segments, with the mean diameter ranging from 4.4 to 6.8μm [16]. The maximum segment order ranges from 8 to 12. Almost all branching is dichotomous. On an average, each dendrite has 27 segments, 13 branch points, and 14 terminations. By calculating the ratio of the number of segments at order n+1 to twice the number at order n, one can coarsely estimate the probability of non-termination. This estimate was used in the stochastic application of productions of the L-grammar to produce the motoneuron shown in Figure 14.

Once a grammar model for a particular neuron morphology is perfected, it can be used to generate a population of neurons. Since the parameters of the grammar are extracted from their distribution functions, no two cells will be identical; yet they will all be similar. Figure 15 shows such a population of pyramidal neurons.

As an application of the force field model, consider the flattening of the apical dendritic arbor of a pyramidal neuron morphology. To achieve this flat character, a vector field along the vertical axis of the cell can be applied at the point when the turtle is generating segments of this flat arbor. If the turtle is made to reorient itself such that it is always headed in the direction perpendicular to the vector field, the required flatness is simulated. This realignment force is an exponential function of the turtle's position from the surface; the effect is negligible away from the surface and strongest at it.

Currently only flat and cylindrical nucleus boundaries are available, but other shapes can be easily implemented.

C. Grids

As discussed earlier, grids represent an important framework of the growth environment, providing the basis for defining the mapping between the logical and the physical spaces. To obtain a neuron morphology in the curved tissue, we suggest that a curvilinear grid be imposed through this tissue. The resolution of the grid will depend on the curvature of the nucleus surfaces being modeled as regions of higher curvature need to be subdivided finer. Higher resolution also guarantees simpler cell shapes, which in turn are easier to map. The grid should have the same number and arrangement of cells as the cells in the uniform grid of the logical space. (Figure 8)

When transforming the logical space into the physical space, a one-one mapping of each cell of the uniform grid in logical space to a cell in the curvilinear grid in the physical space is required. Based on the cell occupancy information, all the segments within a single cell in the logical space are mapped into segments of the corresponding cell in physical space.

A single segment may be completely inside a cell or may span several such cells, depending on the granularity of the grid. In the case where a segment spans across multiple cells, we suggest that the segment be broken into smaller pieces such that each of these pieces is completely inside a single cell.

D. Stochastic Seeding of Neuron Populations

The population morphological data can be generated in two ways:

1. by invoking the same grammar as many times as the number of required cells; or

2. by introducing a higher level production into the grammar which distributes the neurons by walking the turtle in model space and planting the somata of the neurons. This gives a stochastic L-system grammar for a neuron population.

Although the latter approach seems more elegant, system and memory restrictions force the former approach to be adopted if the number of neurons required is fairly large.

Different seeding distributions can be invoked from within the grammar by adding these distribution functions to our neuron generator. Seeding distributions for the Poisson, the Poisson Sphere and the jittered distributions have been built into the system. In case the second approach is used, these distribution functions can be used to walk the turtle to generate the population.

E. Conclusion

The set of techniques developed in this chapter for the morphological modeling of neurons will be formalized in the next chapter in a system, namely, the *Neuron*

Morphology Modeler (NMM). This system brings coherence to the techniques and strategies discussed so far, and provides an implementation design. The remaining part of the thesis defines and expands this design.

CHAPTER VII

NEURON MORPHOLOGY MODELER

Experiments with various stochastic L-system models of neuron morphology have given us neurons with promising proximity in shape to neurons illustrated in the neurobiology literature. Work is in progress to improve this degree of proximity, but more importantly to validate the L-system neuron models against databases of manually-traced neurons. To this end the Neuron Morphology Modeler (NMM), described below, has been devised to provide for iterative refinement of L-system models (Figure 16).

A. Data Level of the Modeler

Central to the scheme at the data level is an object-oriented Neuron Morphology Data Repository (Paradox for Windows, Version 5.0, Borland International, Scotts Valley, CA). This DBMS stores for each neuron a representation of its soma and for each dendritic or axonal arbor a representation of its segments and junctions, as described above.

This DBMS can receive input (at the data level) from the *neuron tracer* (Neuron Tracing System, Eutectic Electronics, Inc., Raleigh, NC) or (at the knowledge level) from the L-system *neuron generator* (a derivative of public-domain software known as *lsys*, copyright (c) 1990, 1991 by Jonathan Leech, University of North Carolina). In this sense the DBMS is indifferent as to whether it is storing data from real manually-traced neurons or L-system-generated synthetic neurons.

Upon command, the DBMS can prepare database files for *the spreadsheet* program (Excel, Version 5.0, Microsoft Corporation). The spreadsheet program allows data editing and graphical summarization and provides the natural input to the next stage of analysis, the knowledge level of the Neuron Morphology Modeler described below.

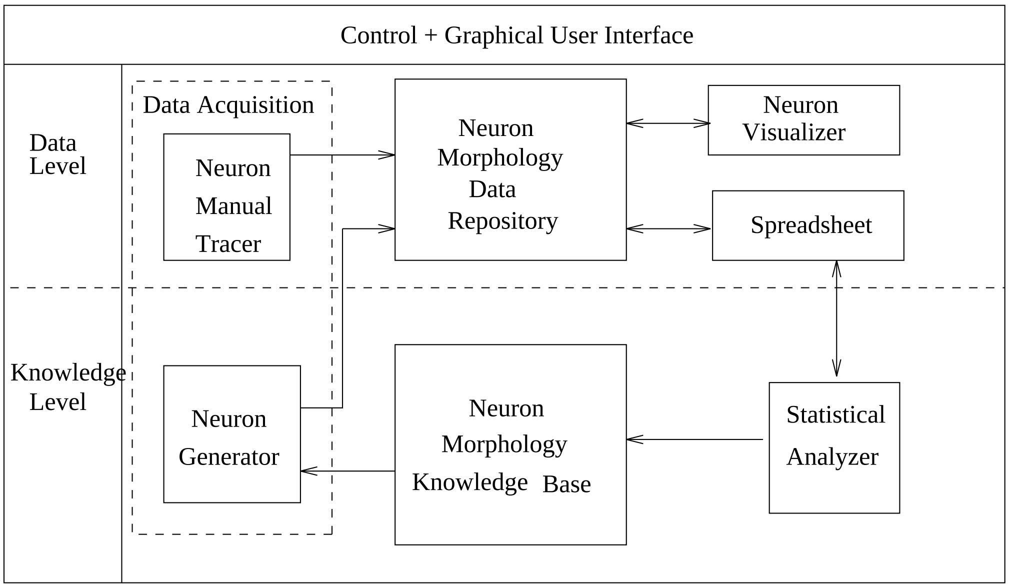


Fig. 16. Neuron morphology modeler

B. Knowledge Level of the Modeler

We now enter the knowledge level (Figure 16), where the spreadsheet generated by the Neuron Morphology Data Repository provide the conventional input to the *statistical analyzer* (Systat, Version 5.03, Systat Inc., Evanston, IL). In particular the statistical analyzer estimates the distribution functions for segment non-termination, segment length and trajectory generation, and the rotations characterizing bifurcations. At present only ad hoc techniques are available for using Systat for this purpose. The parametric statistical dependencies and distribution functions ascertained by the statistical analyzer are used to refine the L-system models of the neuron population.

Next, paralleling the neuron morphology DBMS, we have at the model level the *neuron morphology knowledge base*, which stores the L-system grammars and their associated distribution functions for stochastic generation of synthetic neurons.

And finally, closing the loop, we return to the *neuron generator*, described above.

With experience we anticipate that the system components at the knowledge level will merge into an expert system. The data level NMM provides the external environment for this model-level subsystem. The long term goal, of course, is for the knowledge level subsystem to converge upon stochastic L-system models such that the input from real manually-measured neurons and synthesized neurons can not be statistically distinguished.

CHAPTER VIII

DATA ACQUISITION

As shown in the Figure 16 the neuron morphology data is captured through two different sources, namely from the actual measurement of the manually-traced neurons and from synthetically-generated neurons.

A. Manually-traced Neurons

Neuron-tracing equipment and software are available that allow the neuroanatomists to trace neurons [16], [9]. These typically output a series of points along segment trajectories with the segment width at these points. The output files from these systems can be read into the Neuron Morphology Data Repository, as will be discussed later.

B. Synthetically-generated Neurons

Artificial neuron morphologies are generated using stochastic L-system grammars [1](Figure 14, 15, 13). An L-system grammar generates a parenthesized string of terminal symbols. Each terminal symbol serves to orient or generate a geometric model of the constituent part of the neuron. For modeling dendritic or axonal arbors, the principal constituent parts are segments, which are realized as generalized cylinders, and junctions. A neuron generator is utilized for this purpose.

CHAPTER IX

DATA DEPOSITION

The Neuron Morphology Data Repository is a collection of a relational database and conversion programs that convert between different formats and the object model format. The relational database table format, the manually-traced neuron data format, the L-system synthesized data format, and various spreadsheet formats can all be converted into the object-model format and vice versa. A Neuron Morphology Visualizer is also provided that can read the object model format of a neuron's morphology and displays the neuron in 3D.

A. Relational Database

The object model, a volatile memory model, is made persistent by converting it into a nearly identical relational model for widespread distribution amongst investigators. The links between segments and junctions are replaced by reference fields (Segment and Junction IDs). Objects are downloaded into tables, which instead of volatile pointer links, now carry the IDs of the segments or junctions to which the links were attached. By assigning IDs, it is also possible to refer to unique pieces of the neuron. For example, a researcher can refer to an unusual spine density of a segment by specifying the appropriate segment ID. Similarly junctions, neurons, somas, and spines are assigned IDs and form the keys of their respective relations (or tables).

Eight table formats have been designed: two storing segment-related information, and one each for cells, somas, arbors, junctions, nodes, and spines. The design is flexible and can be easily extended.

```
CELL
        CELL_ID     /* Key */
        TYPE
        PARENT
        ORGANISM
        POSITION_IN_TISSUE
        SOMA

SOMA
        SOMA_ID        /* Key */
        VOLUME
        NUM_SEG_ORIG
        SHAPE_DESC
        NUM_CONTOUR

JUNCTION
        JUNCTION_ID  /* Key */
        SEG_ID_BASE
        SEG_ID_PRIM
        SEG_ID_SEC
        SEG_ID_TER
        JUNCTION_TYPE

NODE
        NODE_ID      /* Key */
        X
```

```
        Y
        Z
        DIAMETER
        TYPE

SEGMENT1
        SEG_ID        /* Key */
        NODE_ID       /* Key */

SEGMENT2
        SEG_ID        /* Key */
        ORDER
        JUNCTION_ID
        NODE_ID_INIT
        TERM_NODE_ID


SPINE
        SPINE_ID       /* Key */
        SEG_ID

ARBOR
        ARBOR_ID       /* Key */
        NO_OF_SEGS
        NO_OF_JUNC
```

CHAPTER X

3D NEURON MORPHOLOGY VISUALIZER

A. Introduction

A menu-driven interactive 3D neuron morphology visualizer has been developed as a component of the Neuron Morphology Modeler discussed earlier (Figure 16). Using this tool a neuron's morphology can be displayed, edited in 3D, and saved for further analysis.

B. Input

The visualizer requires two files for input. The morphological information is packaged into two important object model files. One file, the *segment file*, contains data corresponding to the segments. It has the segment ID, the number of sampled points on the segment, followed by a list of the points, followed by a list of segment diameters at these points. The second file, the *junction file*, contains the junction IDs and the segments that share the same branch point at this junction. Both these files can be created from the *lsys* data output using the *convert* utility program. For details on the formats of these files refer to Appendix D.

C. Features

Some salient features of the visualizer are listed below:

- *Display Mode* - The neuron morphology is recreated from the input files and displayed in 3D. Two modes of display are available, namely, the *stick model* and the *cylindrical model*. In the stick model mode, all the segments are displayed as lines with integral thickness. The stick model gives a more instantaneous response when navigational or editing features (described below) are used. In the cylindrical model mode, the segments are drawn as colored, lighted cylinders with diameters in proportion to the segment diameters. This mode provides a more realistic feel of the neuron morphology.

- *Navigation* - The camera view of the scene is under the user's control. Any of the camera controls, namely, the *field of view*, *the eye position*, or the *gaze direction* may be altered interactively.

- *Segment Query* - A segment can be selected using point-and-click interaction. On a successful selection, the selected segment and the dendritic (axonal) subarbor rooted at that segment are highlighted. All the basic information about the segment, such as, the segment's ID, the IDs of the primary, secondary, and tertiary daughter segments, if any, and the ID of the junction which has as its base the selected segment, are accessed from the memory and displayed to the user.

- *Editing* - Some rudimentary editing facilities are also provided. A selected dendritic subarbor can be detached from its parent segment and attached to a new segment. Certain limitations apply, such as the following:

  - A maximum of three daughter segments can be attached to a parent segment, as only bifurcations and trifurcations are currently supported by the visualizer.

  - If a primary or secondary segment of a junction that is a bifurcation is cut, the junction degenerates to a segment, as a junction by definition must have at least two daughter segments originating at its branch point. Thus, the base segment of this degenerate junction is merged with the remaining primary or secondary segment.

- *Plane of Intersection* - A translucent plane can be moved along the z-axis and intersections of this plane with the neuronal processes are displayed simultaneously in a separate window. This is equivalent of taking sections of the neural tissue parallel to the plane. This paradigm is similar to viewing consecutive sections of neural tissue containing a stained neuron.

- *Selective Save* - Any subarbor of the neuronal arbor can be saved separately into the object model format. This selective save facility, coupled with the editing facility, enables a selective analysis of different parts of the same neuron.

A snapshot of the Neuron Morphology Visualizer appears in Figure 17.

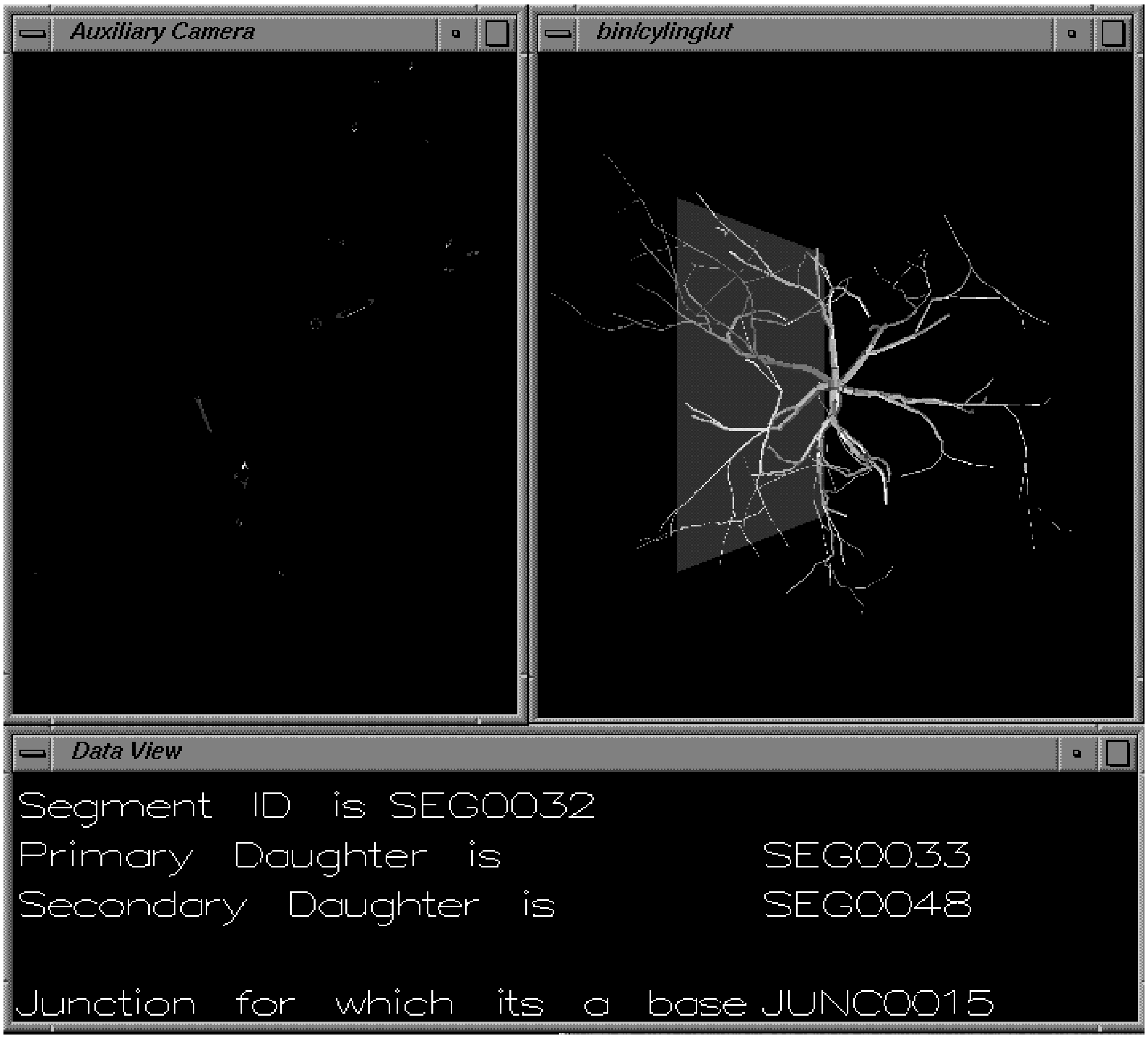


Fig. 17. 3D Neuron Morphology Visualizer

CHAPTER XI

STATISTICAL ANALYSIS AND MODEL VALIDATION

The aim of statistical analysis is to provide a robust basis for the selection of objective values for the set of morphological variables and detect or establish any correlations amongst them. Specifically, we need a to find the distribution functions of the stochastic variables listed in Tables I II, and III, and discover any hidden functional dependencies between these variables. As the number of these variables is large, it is combinatorially infeasible to check for correlations between all possible sets of these variables. Thus, we must first hypothesize for possible correlations, and then look for any consistently high values of statistical dependence.

The statistical analysis can be carried out at two levels in the NMM, at the *Spreadsheet* level or at the *Statistical Analyzer* level.

A. Spreadsheet

The relational database of neuron morphology stores only the directly available measurements such as the segment length and diameter, and some identification related attributes of the object model. Most other attributes need to be derived from those direct measurements. A spreadsheet(Excel, Version 5.0, Microsoft Corporation) is a useful tool for the following reasons:

- The data from a relational table or view can be exported or downloaded as a flat file with separators between the data fields. These flat files are suitable for loading(import) into a spreadsheet.

- A set of builtin functions, coupled with the macros facility provided by the spreadsheet, provide a toolkit for the user to derive other computed morphological variables.

- The spreadsheets also offers an ability to group or split datasets at will. A user can select the required rows and columns by simply dragging the mouse.
- Most common types of plots such as histograms, bar charts, pie charts, and line graphs are provided.
- A spreadsheet data can be exported as flat files to the statistical analyzer.
- Most of all, spreadsheets are ubiquitous and popular software, and require little user training.

However, as spreadsheets are designed for only limited statistical analysis, they fall short at places where the need for more comprehensive statistical analysis is felt.

B. Statistical Analyzer

For carrying out more rigorous analysis, a statistical analyzer (Systat, Version 5.03, Systat Inc., Evanston, IL) is required. This analyzer accepts input from the spreadsheet, or from any flat files that are created by the data repository. It can handle large data files, carry out multivariate analysis and also plot results for better visualization. The parametric statistical dependencies and distributions ascertained by the statistical analyzer can be used to refine L-system models of neuron population.

CHAPTER XII

SIGNIFICANCE

A representational framework for neuron morphology is presented that is adequate both for the quantitative description and stochastic generation of neuron populations. All models are presently limited to modeling neuron morphology as viewed at the limit of optical resolution.

The applications of these stochastic L-system models are several. The L-system models bring coherence to existing neuron databases; they allow us to estimate the new information contributed by the additional neuron morphological measurements.

The methodology is now largely in place for the measurement and morphological analysis of large neuron populations.

REFERENCES

[1] Alan H. Barr, Bena Currin, Steven Gabriel, John F. Hughes, "Smooth Interpolation of Orientations with Angular Velocity Constraints using Quaternions" in *Siggraph 92: Computer Graphics Proc., Annual Conference Series*, pp. 313-320, 1992.

[2] A. Lindenmayer, "Mathematical Models for Cellular Interaction in Development, Parts I and II.", *Journal of Theoretical Biology*, 18:280-315, 1968.

[3] Ann M. Lohof, Mark Quillan, Yang Dan, Mu-ming Poo, "Asymmetric Modulation of Cytosolic cAmp Activity Induces Growth Cone Turning", *J. of Neurosci.*, 12(4):1253-1261, 1992.

[4] Alan Watt, Mark Watt, *Advanced Animation and Rendering Techniques*, ACM Press, New York, 1992.

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

[6] E. Keppel, "Approximating Complex Surfaces by Triangulation of Contour Lines", *IBM J. Res. Develop.*, June 1974.

[7] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego, 1993.

[8] J. M. Blatt, V. F. Weisskopf, *Theoretical Nuclear Physics*, John Wiley & Sons, New York, 1952, Chapter X.

[9] M. Rapp , Y. Yarom, I. Segev, "Physiology, Morphology and Detailed Passive Models of Guinea-pig Cerebellar Purkinje Cells", *J. Physio.*, 474.1:101-118, 1994.

[10] P. C. Letourneau, "Axonal Growth and Guidance" in *Molecular Bases of Neural Development* (Edelman G.M., Gall W.E., Cowan W.M. eds), pp 269-293, Wiley, New York, 1985.

[11] P. M. Knupp, S. Steinberg, *Fundamentals of Grid Generation*, CRC Press, Boca Raton, Florida, 1993.

[12] P. Prusinkiewicz, A. Lindenmeyer, *The Algorithmic Beauty of Plants*, Springer-Verlag, New York, 1990.

[13] P. Prusinkiewicz, M. James, R. Mech, "Synthetic Topiary", in *Siggraph 94: Computer Graphics Proc., Annual Conference Series*, pp. 351-359, 1994.

[14] S. T. Bok, *Histonomy of the Cerebral Cortex*, Elsevier Publishing Company, Amsterdam, Netherlands, 1959.

[15] W. Rall, "Core Conductor Theory and Cable Properties of Neurons". in *The Nervous System: Cell Biology of Neurons*, E.R. Kandel (ed). Section 1:*Handbook of Physiology*, Part 1, Vol. 1. Bethesda: American Physiological Society, pp. 39-97, 1977.

[16] X. Y. Chen, J. R. Wolpaw, "Triceps Surae Motoneuron Morphology in the Rat: A Quantitative Light Microscopic Study.", *J. Comp. Neurol.*, 343:143-157, 1994.

APPENDIX A

CLASS DEFINITIONS

```
class Cell
{
 private:
  char strCellID[MAX_ID];       /* Cell ID */
  char * strParentOrganism;     /* Parent Organism */
  char * strPositionInOrganism; /* Position */
  char * strSource;             /* Source */
  Soma * pSoma;                 /* Soma */

 public:
   .
   .
   .
  Soma* get_soma_ptr(void);  /* returns Soma pointer */

};
class Soma
{
private:
  char strSomaID[MAX_ID];    /* Soma ID */
  Cell  * ptrCell;           /* pointer to parent Cell */
  Seg * apSegRoots[MAXSEGS]; /* array of seg ptrs which are roots */
```

```
  Contour *
       apContSlices[MAXCONTOURS];
                              /* Cross-section Contours */
  int iRootCount;             /* no. of roots */
public:
  float volume();             /* returns volume */
  char  shape();              /* shape description */

};
class Arbor {
  private:
  char id[MAX_ID];                /* Arbor ID */
  char type;                      /* dendritic or axonal */
  Seg *root_seg;                  /* pointer to its root segment */
  public:
    .
    .
    .
   float integrated_length();   /* Sum of all the segments lengths
                                   the arbor */
};
class Seg {
  private:
   char id[MAX_ID];               /* Segment ID */
   Junction *base, *primary,      /* pointers to the junctions */
            *secondary, *tertiary;
```

```
  int order;                  /* the order of the segment */
  int count_of_pts;           /* No. of points (knots)    */
  char node_id_proximal[MAX_ID],
       node_id_distal[MAX_ID]; /* Proximal and Distal Node IDs */
  float diameters[MAX_POINTS]; /* Diameters at the knots    */
  Point pt_list[MAX_POINTS];   /* Coordinates of the knots  */
public:
  .
  .                            /* Various bookkeeping members */
  .
float arc_length();       /* returns chord length of the segment*/
float mean_diamter();     /* returns mean diameter       */
float taper_ratio();      /* returns the ratio of final */
                          /* to initial diameter */
float taper_per_unit_length(); /* return mean taper per
                                  unit length*/
float curvature();        /* returns the curvature */
float torsion();          /* returns the torsion   */
Quaternion rotation_init_final();
                          /* returns the Quaternion that takes */
                          /* the incoming Frenet frame to final*/
Quaternion* rotation(int i);
                          /* returns successive Quaternions */
Point knot(int i);         /* returns the ith knot coordinates */
};
```

```
class Junction {
private:
  char id[MAX_ID];               /* junction ID */
  char branch_pt_node_id[MAX_ID]; /* node id of the branch point */
  Seg *base, *primary;           /* pointers to the segments that */
      *secondary, *tertiary;     /* make up the junction */
public:
      .
      .
      .
 int no_of_segments();   /* returns number of daughter segments*/
 float inter_segment_angles(int i, j);
                       /* returns angle between segments i and j  */
                       /* where i=0,1,2,3 for base, primary,      */
                       /* secondary and tertiary respectively     */
                       /* in radians                              */
 float roll_angle(int i); /* returns the angle phi for ith daughter  */
 float turn_angle(int i); /* returns the angle theta for ith daughter*/
 float twist_angle(int i);/* returns the angle psi for ith daughter  */
 float form_factor(int i);/* returns the form factor for ith daughter*/
 float length_reduction_ratio(int i);
                        /* returns the ratio of length of ith
                        /* daughter to base*/
 float diameter_reduction_ratio(int i);
                        /* returns the ratio of diameter of
```

```
                         /* ith daughter to base*/
 float Ralls ratio();     /* returns the Rall's ratio */
};

class Node {
private:
  char id[MAX_ID];        /* node id */
  char type;              /* Branch node, terminal node, intermed node*/
  Seg* arrSegs[3];        /* poiters to the Segments */
public:
  .
  .
  .

};

class Spine{
 private:
   char id[MAX_ID];       /* spine id */
 public:
  .
  .
  .
};
```

APPENDIX B

QUATERNIONS

A complex number with one real component and three imaginary components is called a quaternion [4].

$$q = a + bi + cj + dk \tag{B.1}$$

In condensed notation q = (s, v)

where:

$$(s, v) = s + v_x i + v_y j + v_z k \tag{B.2}$$

s is thought of as the scalar part of the quaternion and v the vector part with axes i, j, k.

The multiplication of two quaternions:

$$q_1 = (s_1, v_1) \text{ and } q_2 = (s_2, v_2) \tag{B.3}$$

is given by:

$$q_1 q_2 = (s_1 s_2 - v_1 . v_2, s_1 v_2 + s_2 v_1 + v_1 \wedge v_2) \tag{B.4}$$

The conjugate of a quaternion

$$q = (s, v) \text{ is defined as } \bar{q} = (s, -v) \tag{B.5}$$

The act of rotating a vector **r** by an angular displacement (θ, **n**) is the same as performing the operation q ( ) $\bar{\text{q}}$ on the quaternion (0, **r**), where q is a unit quaternion $(\cos(\theta/2), \sin(\theta/2)\mathbf{n})$

For every rotation, two quaternions, +q and -q, lying at the antipodal ends of a hypersphere, correspond to it [1].

Two quaternions multiplied together, each of unit magnitude, will result in a single quaternion of unit magnitude.

There are several reasons to use quaternions to describe rotations. First the quaternion space has the same local topology and the geometry as the set of rotations. Second, the number of coordinates used in describing a quaternion is small (4 numbers, in contrast to the 9 in a 3×3 matrix). Third, the number of constraints on these coordinates is small: the only constraint on a quaternion representing a rotation is that it have unit length; a 3×3 matrix must satisfy six equations to represent a rotation [1].

VITA

Kishore Mulchandani received the B.E degree in Computer Science and Technology from the University of Roorkee, India, in 1989. 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 teaching assistant for the undergraduate courses, Computer Graphics and Analysis of Algorithms.

From June 1989 until December 1992, he worked as a system analyst programmer in the IS industry.

He can be contacted at the following address: 8 Greenleigh Drive, Sewell, NJ 08080, USA.