STOCHASTIC GENERATION OF BIOLOGICALLY ACCURATE BRAIN NETWORKS

A Thesis

by

ARAVIND ALURI

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

December 2005

Major Subject: Computer Science

STOCHASTIC GENERATION OF BIOLOGICALLY ACCURATE BRAIN NETWORKS

A Thesis

by

ARAVIND ALURI

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

Approved by:

Chair of Committee,	Bruce McCormick
Committee Members,	Yoonsuck Choe
	Donald House
	Nancy Amato
Head of Department,	Valerie Taylor

December 2005

Major Subject: Computer Science

ABSTRACT

Stochastic Generation of Biologically Accurate

Brain Networks. (December 2005)

Aravind Aluri, B.E.(hons), Birla Institute of Technology and Science Pilani, India

Chair of Advisory Committee: Dr. Bruce McCormick

Basic circuits, which form the building blocks of the brain, have been identified in recent literature. We propose to treat these basic circuits as "stochastic generators" whose instances serve to wire a portion of the mouse brain. Very much in the same manner as genes generate proteins by providing templates for their construction, we view the catalog of basic circuits as providing templates for wiring up the neurons of the brain. This thesis work involves a) defining a framework for the stochastic generation of brain networks, b) generation of sample networks from the basic circuits, and c) visualization of the generated networks.

To Sri. Bapu Lakshmi Narayana Aluri for being my teacher, father, and friend.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Bruce H. McCormick for his guidance, encouragement, and patience. This thesis is a result of his vision and ideas. My sincere thanks to Dr. Choe for his technical input, regular feedback, and constant support. Also, thanks to everyone in the Brain Networks Laboratory here at Texas A&M for their cooperation.

Jens Eberhard provided the source code for his NeuGen software and helped with its extension design. Thanks Jens for your time and help. I would also like to thank Padraig Gleeson for providing us access to his NeuroConstruct software.

I express my gratitude to Nitin Ved and John George for their help with the mechanics of writing this thesis, and to Gopinath Vageesan for his constant support.

Thanks to the Computer Science Department at Texas A&M University especially to Dr. Lawrence Petersen for supporting me as a teaching assistant throughout my graduate studies.

Lastly, for their unconditional love and support, I am grateful to my family: parents, brother, and sister.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Introduction

"To extend our understanding of neural function to the most complex human physiological and psychological activities, it is essential that we first generate a clear and accurate view of the structure of relevant centers, and of the human brain itself, so that the basic plan and the overview can be grasped in the blink of an eye". - Santiago Ramón y Cajal (Nobel Laureate, 1909) [36].

We intend to contribute towards Cajal's vision of generating a clear and accurate view of the structure of relevant centers of the brain. In the recent literature, multiple *basic circuits*, which form the building blocks of the brain, have been identified. We propose to treat these basic circuits as *stochastic generators* whose *instances* serve to wire a portion of the mouse brain. Very much in the same manner as genes generate proteins by providing templates for their construction, we view the catalog of basic circuits as providing templates for wiring up the neurons of the brain.

B. Motivation

The neuronal connectivity of human and other mammalian brains is so far largely uncharted. Indeed, anatomically correct network models of the brain do not exist at present for the mammalian brain of any species; there is simply not enough three-dimensional (3D) neuro-anatomical data available concerning mammalian brain microstructure and, specifically, its neuronal location and connectivity. Provided we can

---

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

generate synthetic brain networks from a small number of basic circuits, we can cast these neurons into a web-based database of synthetic brain microstructure. This is the *direct* (or *synthetic*) brain construction process. We can then turn the table to the *indirect* (or *reciprocal*) process, and develop algorithms to find basic circuits directly from the web-based database of synthetic brain microstructure. If we are able to do that, then at a future date, and not as part of this thesis, we should be able to apply these same discovery algorithms to extract basic circuits from an empirical web-based database of brain microstructure (*basic circuit mining*). Based on empirical datasets acquired from new three-dimensional microscopes, such as the Serial Block Face Scanning Electron Microscope (SBF-SEM) [11] and the Knife-Edge Scanning Microscope (KESM) [27], the reconstructed empirical datasets can be continuously mined to find new basic circuits.

C. Outline of the thesis

Fig. 1 represents the central idea behind this thesis. We propose that brain networks are often, but not exclusively, arranged in a lattice structure defined by five groups of input morphological parameters. Lattice structures are known to play a prominent role in the brain regions such as cerebellum, retina, striate cortex, hypocampus, and most recently in the hyporeticulum. However, such regularity is presently unknown to exist in other brain regions. The five input parameter groups that collectively determine the stochastic generation of biologically accurate brain networks are:

1. *cell type*, which describes the neuron's morphology, spatial location and its statistical parameters

2. *cell packing*, that describes the spatial distribution of a particular cell type

3. *connection map*, that specifies the synaptic assembly between neurons

4. *lattice/group*, which determines how the repeating microstructure is locally arranged, and

5. *solid model of the brain region*, which dictates the global structural organization of a given portion of the brain (*e.g.*, the manifold underlying the cerebellum)

We theorize that when these groups of parameters are fed into the Network Stochastic Generator (see Fig. 1) , it generates a morphologically-accurate model of the brain with tunable levels of detail (XwebDB and YwebDB). However, non-local connectivity questions are not treated in the presented formalism. XwebDB, which is a full-blown representation of the brain network, is projected into a simplified YwebDB, where neurons are represented by (often truncated) multi-compartmental models. The projected brain network model is then visualized or is run as a simulation in one of the popular neuro-simulators, such as GENESIS [4] and NEURON [18].

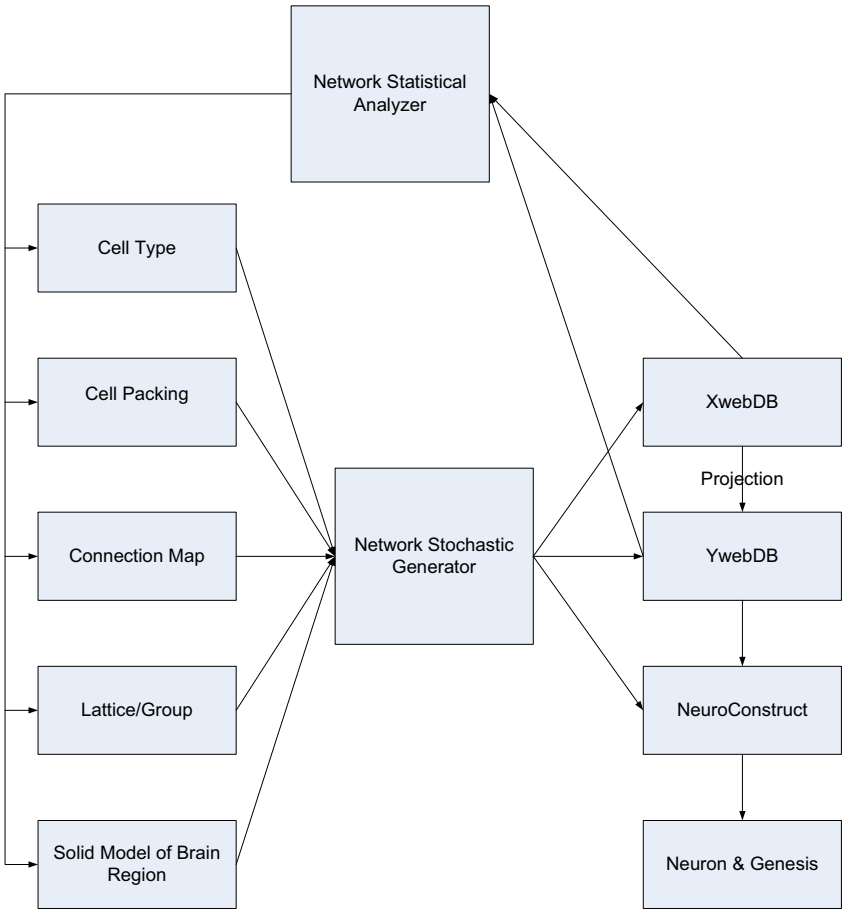


Fig. 1. Outline of the thesis

GENESIS and NEURON have both been developed and refined over a period of years. We complement the capabilities of these simulators by providing anatomical accuracy to the model brain network using new software tools such as NeuroConstruct [15] and NeuGen [12], rather than augmenting the model's functional aspects. NeuroConstruct is a software abstraction layer on top of the GENESIS and NEURON simulators which provides 3D visualization capabilities. NeuGen is a software tool for generation of morphologically realistic neurons and networks in 3D. We added capabilities to the existing NeuGen code to help it fit into the role of the "Network

Stochastic Generator".

The "*Network Statistical Analyzer*" can be used to mine through the biologically observed XwebDB neurons and their YwebDB projections to derive statistical estimates of the parameters groups, which can in turn be used to produce more accurate brain network models.

D. Background and rationale

This section covers:

1. literature and prior work done by other investigators in this area,
2. our research base, and
3. how this thesis adds to the existing pyramid of knowledge.

1. Canonical or basic circuits

In the recent literature, several basic circuits, which form the building blocks of the brain, have been identified. Most of the known basic circuits are documented in Shepherd's book [37]. The book documents approximately 60 basic circuits, and their variants.

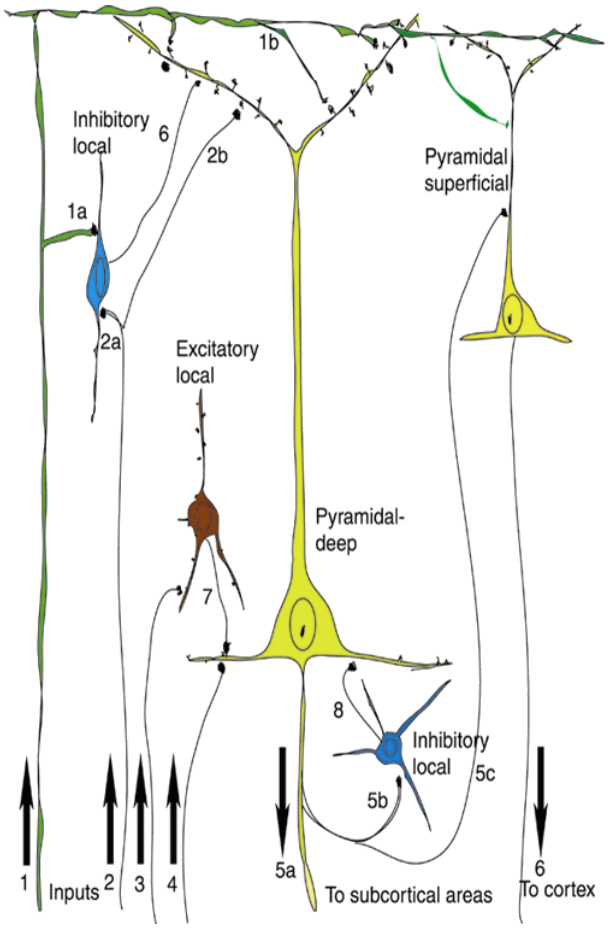


Fig. 2. An example of a basic circuit in the brain [13]

Fig. 2 represents the dominant basic circuit in the cerebral cortical networks. Four neuronal cell types (pyramidal-deep, pyramidal-superficial, inhibitory local, excitatory local), distinguished by their depth of soma in the cortex and the span of their dendritic and axonal trees are shown. Several fibers shown (axons or their collateral branches) have their somata reside outside the view of the model. Four types of afferent fibers are shown: those that target distal dendrites, either (1) by direct connection (Input 1, synapses 1a, 1b) or (2) mediated by an inhibitory local cell (Input

2, synapses 2a, 2b); and those that target proximal dendrites of the pyramidal-deep cell, either (3) mediated by an excitatory local cell (Input 3, synapse 3) or (4) by direct connection (Input 4, synapse 4). Efferent fibers (axons 5a or collaterals 5b, 5c connect pyramidal cells (pyramidal-deep, from the same or adjacent columns) making synapses on inter-neurons (inhibitory local), pyramidal cells (pyramidal-deep), or sub-cortical areas 5a. Inter-neuron axons 6, 7, and 8 connect pyramidal cells [13].

Basic circuits in the mammalian brain are characterized by:

1. the morphology and neurotransmitter type of their participating neurons in a cortical area or nucleus, and

2. their characteristic interconnection pattern.

Typically, there are 3-6 or occasionally more cell types in each basic circuit. Fig. 2 shows one such basic circuit, first described by Cajal in 1905, now believed to be replicated at least 100,000 times in the mouse cerebral cortex [36].

Other databases like Southampton Neuroscience Group (SoNG) [8], Laboratory of Neuroinformatics (Weill Medical College of Cornell University) [14], the Mouse Brain Web [26], BrainML [5], Kötter's book [21] and CoCoMac [22] supplement the information provided in [37].

2. Visualization

Visually, the circuit classes discussed in the Section 1 are represented as multi-compartmental (or stick-figure) models with ideas borrowed from NeuronDB [32]. An example stick figure of a neuron is illustrated in Fig 3.

NeuGen provides means for visualizing brain networks using more complicated (i.e., having many more compartments) stick-figure models than those shown in Fig. 3

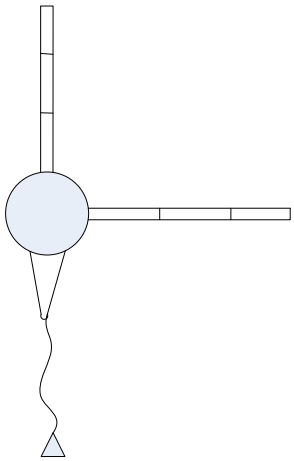


Fig. 3. An example stick-figure representation of a neuron

, using an open-source visualization tool called OpenDX [35]. NeuGen also generates the physiological equivalent of the synaptic assembly using the NEURON simulator.

3. Stochastic generation and network formation

To construct more realistic models compared to simple stick-figures, we need to capture the basic morphological parameters for neurons in the biologically observed neural network, characterizing each neuron's dendritic and axonal arbors and soma. Once we populate a given region in the brain with a stochastically realistic collection of soma, we can construct synthetic neurons using heuristic models of axon and dendrite growth [1][28].

We can then begin the assembly of the synthetic brain network by identifying potential synapses using concepts from [19] and [23]. We first identify pairs of "nearby" neurons that potentially form synapses, thus quickly culling most neuron pairs from further consideration. After we identify pairs of neurons (typically from axon to dendrite/soma) as potentially connectable, we examine them with repeated proximity

tests to identify all appropriate pairs of segments of the desired type within some threshold distance of one other. NeuGen provides one such approximate implementation of the *proximity labeling* algorithm [1]. However, a more efficient algorithm was developed by Lien et al. Using their Probabilistic Road Map (PRM) in our Network Stochastic Generator is a possible future extension of this thesis, and would draw upon the previous work by Lien, Morales and Amato [23].

Our approach for the stochastic generation of synthetic brain networks is to fine tune and extend the code from NeuGen. This allows us to reuse NeuGen's visualization and NEURON model generation features. To this end, we expanded NeuGen's capabilities (provided by Jens Eberhard, University of Heidelberg) by extending the existing class definitions and building new classes to represent basic circuits in the brain. These classes (built in C++) act as building blocks for the generation of the cortical network.

4. Network processing and analysis

The darkened arrows and boxes in Fig. 4 indicate the contribution of this thesis to the interaction diagram of the various components under construction at the Brain Networks Laboratory (BNL). This diagram can be mapped back to Fig. 1. The dotted lines in the figure represent the contribution of this thesis.

E. Summary

In this chapter we presented the motivation, the relevant background, and the outline of this thesis. The rest of the thesis is organized as follows: Chapters II through

[1] recent reports indicate an improvement in the algorithm's performance from $O(n^2)$ to $O(n \log(n))$

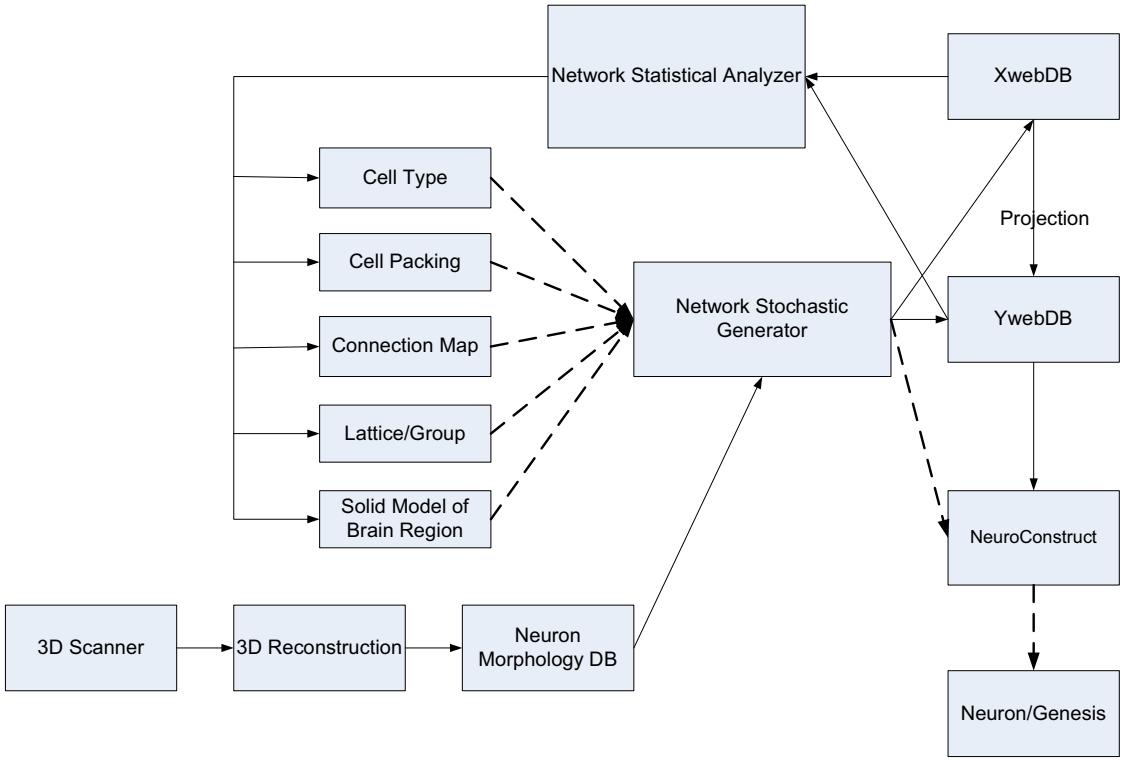


Fig. 4. Flow chart of the proposed brain network processing and analysis at BNL [26]

VI elaborate on the input parameters described in Fig. 1, to the "Network Stochastic Generator". Chapter VII describes the options for projecting XwebDB on to YwebDB. The implementation details of our *Circuits* module in NeuGen are covered in Chapter VIII . We conclude with a summary of the suggestions for future extensions to this thesis in Chapter IX.

CHAPTER II

CELL TYPES

A. Introduction

Cell Type describes a class of similar cells (neurons/glia) by their common morphology and electrophysiology, the statistical distributions of their morphological and electrophysiological parameters (drawn from a population of such cells), and their spatial location (e.g., cortical layer). The cell morphology determines the distribution of segments and spines needed to represent each axonal/dendritic process. On the other hand, the cell electrophysiology determines membrane properties, like ion channels, the neurotransmitter, and neuroreceptors.

The *cell type* data allows one to design stochastic models that can generate realistic virtual neurons, based on biologically plausible "rules" and biophysical determinants. A quantitative morphological analysis of virtual neurons generated from a cell type specification ideally should be statistically indistinguishable from a population of biological neurons from which the model parameters were measured. *Cell types*, accordingly, facilitate the creation of accurate computational models of neuron populations.

B. Data sources

The data sources for *cell type* information can be classified into two broad categories: those that concentrate on morphological data and those that emphasize electrophysiological data.

1. Focus on morphology

This subsection refers to data sources that concentrate on the morphological aspect of *cell type*.

Ascoli [2] at George Mason University maintain the *Virtual Neuromorphology Electronic Database* of the morphology of around 500 neurons in the SWC format. Ascoli's lab is specifically interested in the description and generation of dendritic morphology, and in its effect on neuronal electrophysiology. Their web site also provides references to data from Cannon's Southampton archive [8].

"Synaptic Organization of the Brain" by Shepherd [37] has qualitative morphology information and analysis about the various cell types in ten primary regions of the mammalian brain. The book "Cortex: Statistics and Geometry of Neuronal Connectivity" [6] by Braitenberg et al., provides anatomical data on the mouse in a form useful for modelers, and introduces some novel methods of quantitative neuroanatomy.

In a recent paper, Binzegger et al. [3] gave a quantitative description of the circuits formed in cat area 17 (primary visual cortex) by detailing the morphology of 39 types of neurons. The morphometrical data includes the laminar distribution of dendritic trees, synaptic boutons, and the number of synapses formed by a given type of neuron.

2. Focus on electrophysiology

This subsection refers to data sources that emphasize the electrophysiological aspect of the *cell type*.

The NeuronDB [32], maintained by Senselab at Yale, provides detailed channel data of approximately 40 unique types of neurons. NeuronDB provides a searchable database of various types of neuronal properties and also provides tools for compari-

son of properties across different types of neurons and compartments. Senselab also maintains models submitted by various authors in their modelDB [30]. ModelDB provides an accessible location for storing and efficiently retrieving computational models of neurons.

C. Cell type description

This section is subdivided into two parts. The first part outlines some of the formats and methodologies currently available to describe neuronal morphology and the second part does the same for neuronal electrophysiology.

1. Focus on morphology

a. L-Neuron

"L-Neuron" [1] is a software package for the generation and description of dendritic morphology. The program uses mathematic formalism of Lindenmayer systems. L-neuron can save the generated neurons as compartmental model files that are compatible with the GENESIS and NEURON simulators.

b. MorphML and NeuroML

MorphML [31] provides an extensible Markup Language (XML) application for neural morphology data exchange. This application is consistent with the standard W3C XML Schema, and therefore can be used across different modeling and simulation environments. MorphML can be used alone or included within NeuroML documents.

NeuroML [33], on the other hand, is a collection of related XML projects for modeling different aspects and levels of neural systems. At the cell morphology level, the *Cell Standards* project, which draws ideas from morphML, *Cell builder* in NEURON

and *.p files* in GENESIS, is currently under progress.

c. CVapp

CVapp [7] is a morphology viewer/editor written in Java for converting between file formats and to validate contributed cells. The software can be used to verify and correct neuronal branch morphology. The tool can also be used for reading files in its own (SWC) or Neurolucida format, saving structures for use in the NEURON or GENESIS modeling packages and for visualization of the models.

d. NeuGen

NeuGen [12], developed by Jens Eberhardt at the University of Hiedelberg, is a tool for the generation and description of neuronal morphology of realistic neurons and neural networks in 3D. The input configuration for NeuGen contains parameters for the different neuron cell type morphologies. The configuration parameters for a base neuron class (as shown in Table I) are inherited to all its subclasses (specialized neurons). Table II shows the configuration of one such specialized neuron type (L4stellate). The parameters in Table I and Table II are self-explanatory from the comments that accompany the parameters.

Table I. Configuration file for a Neuron in NeuGen

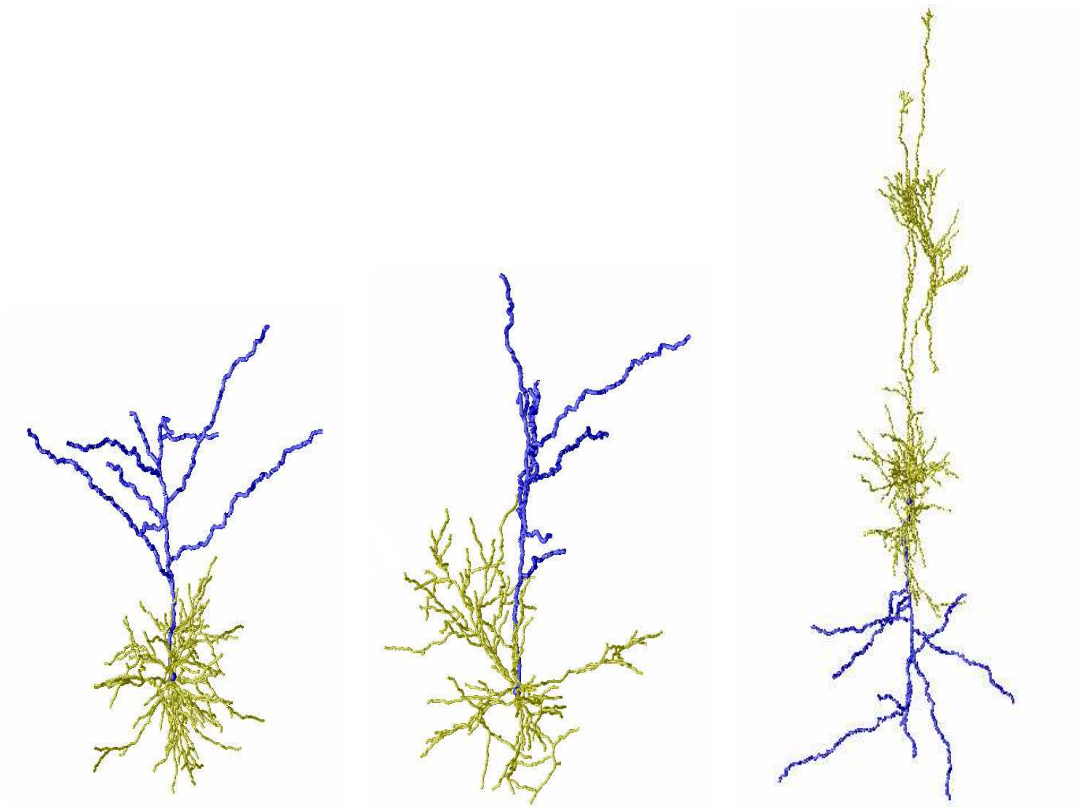
```
neuron
        soma
            rad 5             # soma radius
        deviation             # in multiples of soma radius
            x 1.0
            y 1.0
            z 1.0
        axon
            nparts 100        # number of segments
            nbranch 10.0      # number of branches
            len_param
                z 80.0        # in multiples of soma radius
            rad               # axon radius
                max 0.5
                min 0.1
        # basic parameters for dendrites
        dendrite
            rad               # radius
                max 1.5
                min 0.25
            gen_0            # for branch generation 0
                len_param    # in multiples of soma radius
                    x 20
                    y 0
                    z 0
                branch_angle    # branching angles
                        min 30
                        max 60
                nparts_density 0.25  # segments per um of geometric distance
                nbranch_param 3
                siblings
                    len_param    # in multiples of soma radius
                        x 10
                        y 0
                        z 0
                    siblings
                        nbranch_param 2 # max 10 sibling generations
        nden 12             # number of primary dendrites (apical + basal)
        napiden 0           # number of apical dendrites
        synapse
            rad 1.0         # synapse radius
```

Table II. Configuration file for an L4stellate neuron in NeuGen

```
L4stellate          # for L4 stellate neurons
            dendrite
                rad
                   max 1.5
                   min 0.5
                gen_0
                   len_param
                      x 15
                      y 0
                      z 0
                   branch_angle
                        min 30
                        max 60
                   nparts_density 0.25
                   nbranch_param 4
                   siblings
                      len_param
                          x 10
                          y 0
                          z 0
                      nbranch_param 2
                      siblings
                          nbranch_param 0
                          siblings
            deviation         #in multiples of soma radius
                x 0.5
                y 0.5
                z 0.5
            axon
                nparts 90
                nbranch 15.0
                len_param
                   z 75.0
```

The visualization routines of NeuGen allows one to write the necessary files for a three-dimensional rendered display of the neurons. The cells are visualized in 3D using

the Open Visualization Data Explorer (OpenDX) which uses the native OpenDX file format. Examples of NeuGen neuron type visualizations are shown in Fig. 5.



(a) L4stellate (b) starpyramidal (c) L23pyramidal

Fig. 5. Visualization of neuron types in NeuGen using OpenDX. Blue color denotes the axon arbor and yellow color denotes the dendritic arbor [12], [35].

2. Focus on electrophysiology

a. NEURON and GENESIS

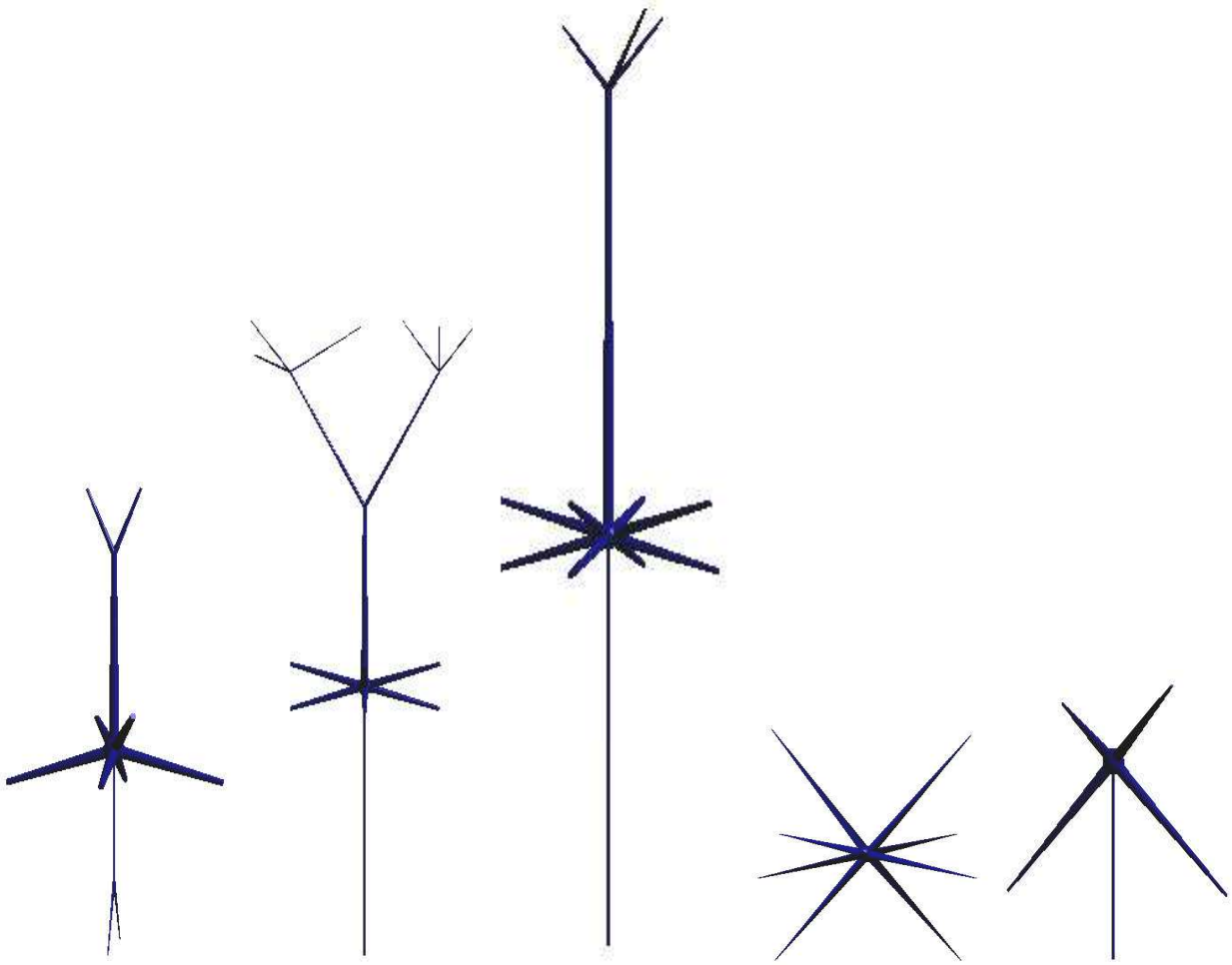
Cell Builder [34], the Graphical User Interface (GUI) tool in the NEURON simulator, helps in constructing and managing models of individual neurons. One can specify

a model's branching pattern, assign biophysical properties, and import and export models in a variety of formats.

In GENESIS, one can describe neuron morphology and electrophysiology by generating the ".p and .g" files using the "GENESIS Scripting Language" [4].

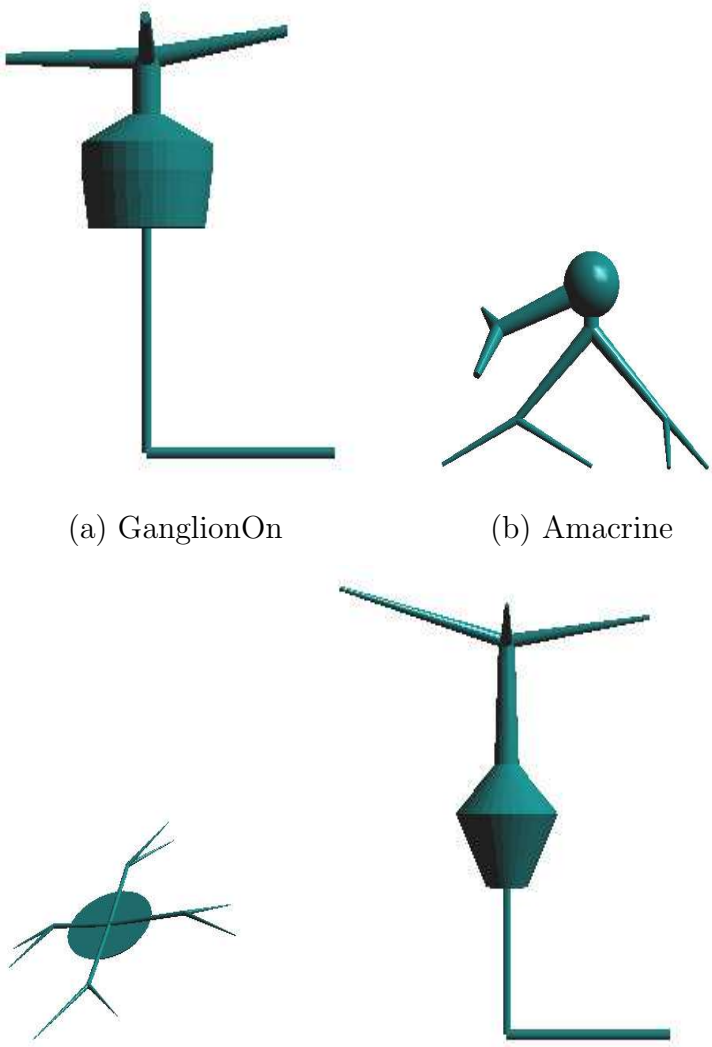
b. NeuroConstruct

NeuroConstruct [15] is a Java-based software tool that can be used to simplify the modeling of neuron networks. The tool allows cell models and channel/synaptic mechanisms created with one of the simulators (GENESIS or NEURON) to be imported, visualized, and incorporated in multi-cell networks of neurons. Apart from good electrophysiological capabilities, NeuroConstruct provides a good interface for morphology description. Figures 6 , 7 , and 8 show some examples of neuron types constructed using the NeuroConstruct GUI. We built these neuron models using morphological estimates from [37].



(a) L23pyramidal (b) L5pyramidal (c) L6pyramidal (d) Stellate (e) Smooth

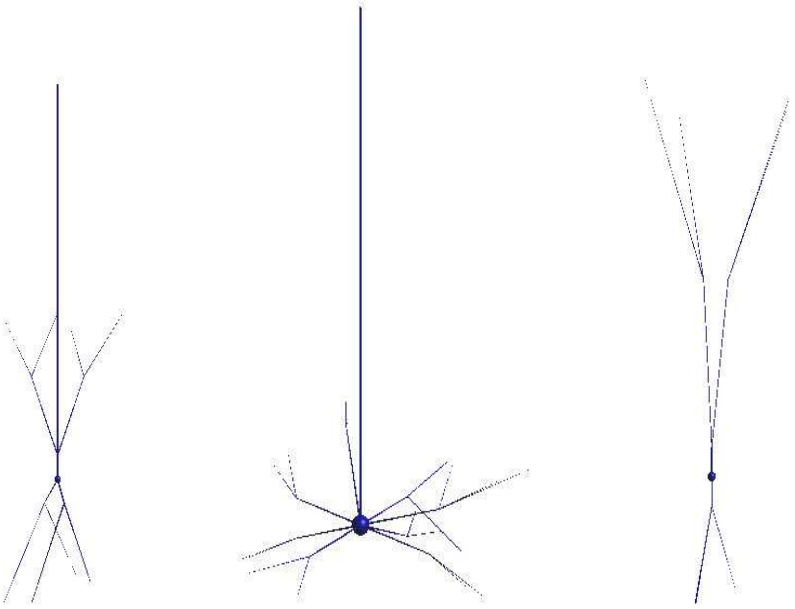
Fig. 6. Visualization of cortical neuron types in NeuroConstruct [15]



(a) GanglionOn (b) Amacrine

(c) Horizontal (d) GanglionOff

Fig. 7. Visualization of retinal neuron types in NeuroConstruct [15]



(a) Xcell (b) Ycell (c) interneuron

Fig. 8. Visualization of thalamic neuron types in NeuroConstruct [15]

D. Summary

To summarize, *cell type* is a very important abstraction for brain model description because these parameters describe cell morphology and electrophysiology, in addition to the spatial location of the cell. Both NeuroConstruct and NeuGen provide interfaces to describe cell morphology and electrophysiology. Fig. 5 shows neuronal types we generated using NeuGen and Figures 6 , 7 , and 8 show the neuron models we built using NeuroConstruct. Our NeuroConstruct cell models can be easily used by other modelers by importing these morphology descriptions into their projects.

CHAPTER III

CELL PACKING

A. Introduction

Cell Packing describes the spatial distribution of a particular type of cell (neuron/glia). Accurate characterization of this distribution is needed to build realistic models of brain networks. Specifically, cell packing refers to distribution of cells and not their connectivity. Commonly used examples, such as uniform and random distributions, are limited in their ability to capture the spatial distribution of cells in sufficient detail. In this chapter, we propose "Voronoi diagrams" as a possible modeling tool to overcome these limitations.

B. Background

1. Simple packing patterns

Cell packing is commonly associated with a specific cell type in a given layer. For example, cell packing is commonly associated with cortical structures in a given layer. Identical cell packing parameters do not necessarily apply to all cell types in a given brain region. Existing brain modeling tools do not provide adequate support for cell packing formats required for building neuronal networks. The only exception, to our knowledge, is NeuroConstruct [15], developed by Padraig Gleeson, University College London. NeuroConstruct supports five different types of cell packing, namely:

1. *simple regular packing*: places cells regularly in 3D, one on top of another
2. *one-dimensional regular spaced packing*: places a fixed number of cells at regular spacing in one dimension

3. *random packing*: places cells randomly in a given region

4. *cubic closed packing* (CCP): arranges cells in a body centered or face centered cubic pattern

5. *hexagonal layer packing*: places cells in a single layer in hexagonal formation

In 2D Euclidean space, the *hexagonal packing* (honeycomb where each circle is surrounded by 6 other circles) has the highest density. The density (proportion of area filled by the circles of equal diameter) of this arrangement is $\frac{\pi}{\sqrt{12}} = 0.9069$. However, in 3D Euclidean space, *cubic close packing* and *hexagonal close packing* have the highest density (proportion of space filled by the spheres). In both of these arrangements, each sphere is surrounded by 12 other spheres, and both arrangements have an average density of $\frac{\pi}{\sqrt{18}} = 0.74048$ [10].

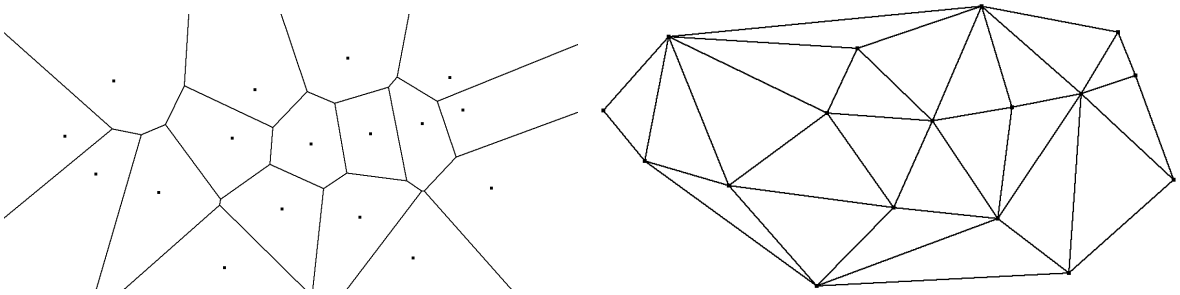
2. Voronoi diagram

Definition of Voronoi Diagram : Let S be a set of n sites in Euclidean space of dimension d. For each site p of S, the Voronoi cell $V(p)$ of p is the set of points that are closer to p than to other sites of S. The Voronoi diagram $V(S)$ is the space partition induced by Voronoi cells [10].

Definition of Delaunay Triangulation: The Delaunay triangulation of S is the geometric dual of the Voronoi diagram of S : two sites of S are linked by an edge in the Delaunay triangulation if and only if their cells are incident in the Voronoi diagram of S [16].

Voronoi diagram (also called Voronoi tessellation or Voronoi decomposition), along with Delaunay triangulation (see Fig. 9), is a very useful tool for representing spatial relationships and growth phenomena. The study of Voronoi diagrams:

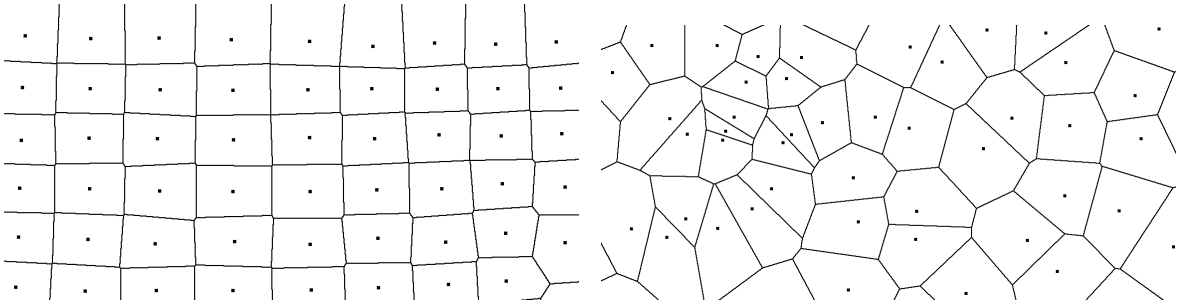
their mathematical properties, their computation, and their generalizations remains a central theme in computational geometry.



(a) Voronoi (b) Delaunay

Fig. 9. An example Voronoi diagram and its corresponding Delaunay triangulation

Fig. 10 illustrates how the distribution of edges in a Voronoi diagram is much more uniform for Face Centered Cubic (FCC) cell packing than for random cell packing.

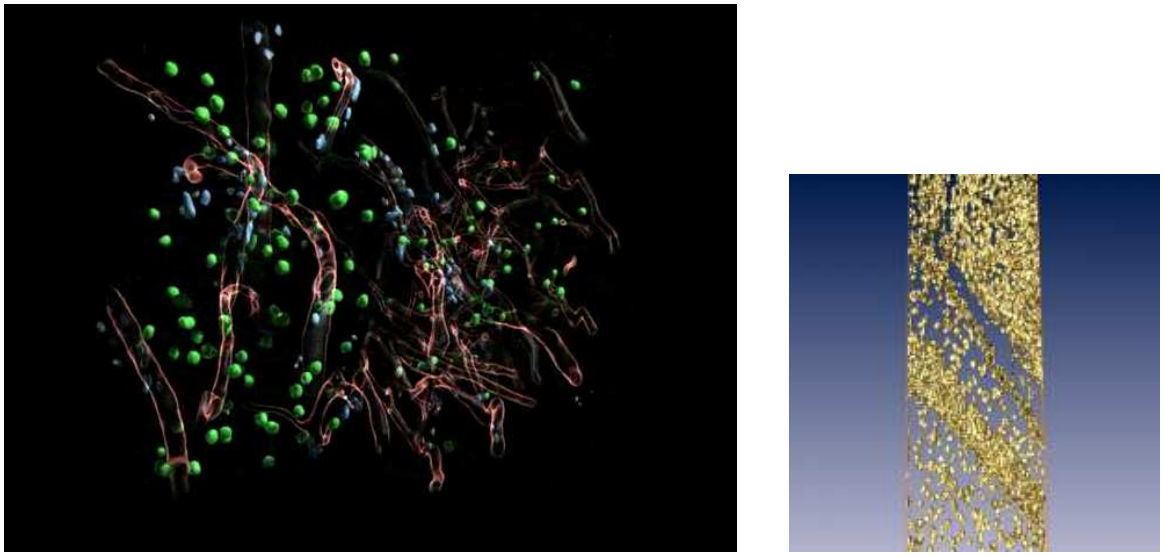


(a) FCC cell packing (b) Random cell packing

Fig. 10. Structured vs. unstructured Voronoi diagrams respectively

C. Using Voronoi diagrams in brain networks

Voronoi diagrams have been used in a very broad spectrum of applications ranging from astronomy to compiler design. In this section, we suggest how Voronoi diagrams can be used for biologically accurate modeling of brain networks. For example, we consider volumetric datasets of Nissl-stained mouse brain tissues from the Brain Networks Laboratory [26].



(a) Blood vessels (b) Hippocampal cells

Fig. 11. 3D reconstructions of nissl data from BNL [25]

Fig. 11(a) shows blood vessels (red/transparent), endothelial cells (purple), and soma (green) reconstruction from Nissl data and Fig. 11(b) shows the 3D visualization of Nissl-stained hippocampal cells. In Fig. 11(b) , one can clearly see layering of neurons in the hippocampus region. These neuron layers are analogous to the layers in a wedding cake. To separate out neurons by layer, one needs to define its normal and

extract a thin slab of cells perpendicular to the normal. The cells in a layer can then be collapsed into a plane perpendicular to the normal of that plane. The Computational Geometry Algorithms Library [9] provides open-source implementation of algorithms which can be used to study the geometric properties of each layer separately using 2D Voronoi diagrams. Fig. 12 illustrates how Voronoi diagrams turn out irregular if the extracted slab crosses more than one layer.

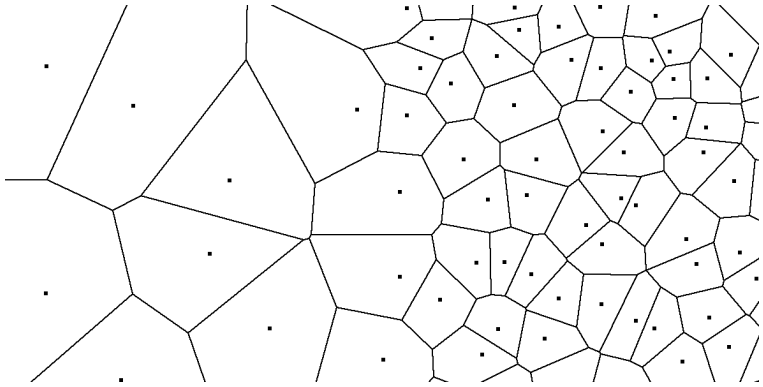


Fig. 12. Irregular Voronoi diagram

Neurons within a layer are in general not all of the same kind. Structured distribution of one cell type in a region doesn't imply that other cell types in that region follow the same distribution. Hence, one needs to distinguish between neuron types by coloring the cell bodies based on some parameter (e.g., proximity to blood vessels). For example, Fig. 11(a) shows endothelial cells with purple and all the other kinds of cells (neurons and glial cells) with green. The distinguishing feature of endothelial cells is that they stick onto the blood vessels (red/transparent) while other cells do not. Another approach to distinguishing cell types is to plot a histogram of a parameter (e.g., the logarithmic volume) of the cell bodies to identify potentially separable classes. Sometimes, if cells from the same layer are classified into different colors based on some characteristics (e.g., soma volume), one of the cell populations

might exhibit regularity while the other sub-population might not. The position of a cell is usually, but not exclusively associated with the center of the soma. For example, the position of a rod or a cone in the retina is defined not by the cell body but by the outer cell structure in the retinal sheet. Similarly, the location at which the dendrite branches determines the position of a bipolar cell in the cerebellum while the rest of the cell just hangs beneath.

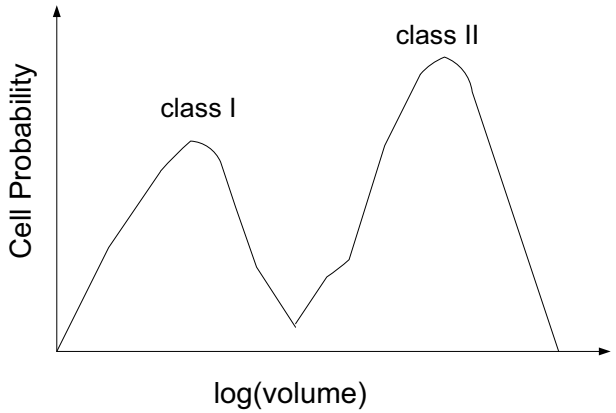


Fig. 13. Graph of cell probability versus log(*volume*) of the cell soma.

There are two easily distinguishable classes of neurons in Fig. 11(a) which show up as two separate clusters in Fig. 13 . Each of these classes can be plotted as a separate 2D Voronoi diagram for individual analysis. These separate diagrams can be overlapped finally to obtain the original (multi-colored)distribution.

D. Summary

Voronoi diagrams are extensively used in a wide range of applications. In this chapter, we proposed some ideas for the application of Voronoi diagrams to modeling cell packing in brain networks. The prospects of quantitatively classifying cell packing by this paradigm are promising.

CHAPTER IV

CONNECTION MAP

A. Introduction

Connection map specifies the synaptic assembly between neurons, including the identification of synaptic type (e.g., excitatory/inhibitory). Synapses are cell-cell contacts found at the end of cell's thread-like extensions through which the cells communicate. In electron microscopy these synapses normally can be identified on purely morphological grounds. In light microscopy, however, synaptic assembly is generally restricted to finding "putative synapses" to wire up the neurons. Small diameter fibers and spines are normally invisible.

B. Background

1. Synapses

Synapses, from the Greek words meaning to "clasp together", are the contact points where one neuron communicates with another. The stereotypical and most abundant synapse in the central nervous system is the asymmetric synapse occurring between an axon and a dendritic spine. Other synaptic relationships exist and involve different parts of the neuron. For instance, axo-axonic, axo-somatic, axo-dendritic, dendro-somatic, and dendro-dendritic synapses can occur and provide alternate mechanisms for functional communication between neurons [17].

2. Light microscopy (LM) vs. electron microscopy (EM)

Structural and functional classifications of axons, dendrites, and their synapses are still emerging. The use of electrophysiology, light microscopy (including laser scan-

ning), serial electron microscopy, and 3D computer-aided reconstruction facilitate the study of neurons and the intricacies of their synaptic assembly within the brain.

Light microscopy allows one to identify only larger processes and then only a subset of the synapses on these visible processes. Moreover, axon diameters, as measured in LM, are too inaccurate to reliably determine axonal delays. In contrast, electron microscopy allows one to identify all neuronal processes and their synapses, but over a very limited spatial volume. Using EM, one can also see fine details of the interior structures of cells (nucleus, mitochondria, etc.). This requires sampling at 10 nm resolution, which is not possible using LM (limited to a sampling resolution of 200 nm). Electron (nano-scale) and light (micro-scale) microscopy can be used at their respective scales to paint the overall picture of the brain connectivity and establish rules for identifying putative synapses as likely biologically valid.

3. Synaptic assembly

Assembly (wiring up) of the whole small animal brain network requires identification of putative synapses because the time required for comprehensive electron microscopy of the brain is prohibitive.

a. $O(n \log n)$ proximity labeling algorithms

Lien et al. identify pairs of "nearby" neurons that potentially form synapses, thus quickly culling most neuron pairs from further consideration [23]. After they identify pairs of neurons (typically from axon to dendrite/soma) as potentially connectable, they examine them with repeated proximity tests to identify all appropriate pairs of segments of the desired type within some threshold distance of each other. After neuron segment pairs in the reconstructed mouse brain have been identified as potentially connectable, they employ a series of increasingly precise proximity tests, based

on known or measured synaptic affinities, to find all pairs of appropriate segments within some threshold distance of each other.

b. Deriving physical connectivity from neuronal morphology

In [19], Kalisman et al. present a model that allows prediction of the probability for the formation of appositions between the axons and dendrites of any two neurons based only on their morphological statistics and relative separation. Statistics of axonal and dendritic morphologies of single neurons are obtained from 3D reconstructions of biocytin-filled cells, and a statistical representation of the same cell type is obtained by averaging across neurons according to the model. A simple mathematical formulation is applied to the axonal and dendritic statistical representations to yield the probability for close appositions.

C. Tools for building a connection map

1. Synapses in NeuGen

NeuGen [12] provides an approximate implementation of the *proximity labeling algorithm* between synthetically generated neurons. NeuGen creates synapses by distributions and/or distance. It uses the NetCon class of NEURON to create an interface to the simulation program. At the time of this thesis, the algorithm ran in $O(n^2)$ where n is the number of segments in the collection of neurons. Hence, the algorithm did not scale well for large values of n. However, Eberhardt et al. have recently reportedly achieved a better complexity of $O(n \log(n))$ by parallelizing the algorithm.

2. Synapses in NeuroConstruct

In NeuroConstruct, synapses are specified as connections between a number of points on cells in one cell group to points in another cell group. Some of the factors that need to be specified in NeuroConstruct are:

1. source cell group: (pre-synaptic cells)
2. target cell group: (post-synaptic cells)
3. synaptic properties: (which cell process contains the synaptic mechanism)
4. the delay
5. firing threshold and weight
6. whether the axons/dendrites grow to meet the post-synaptic connection point
7. method of searching for a connection point (random, closest, etc.)
8. maximum and minimum lengths of the connection
9. various constraints on the number of connections to make between the cells groups

Fig. 14 shows a synaptic assembly of the cortical region built using NeuGen and Fig. 15 shows a synaptic assembly of the cerebellar region built using NeuroConstruct.

D. Summary

To summarize, synaptic assembly is a central component of brain network modeling. In this chapter, we described some tools for modeliing synaptic assembly and proposed possible extensions to overcome the tools' deficiencies. None of the existing tools,

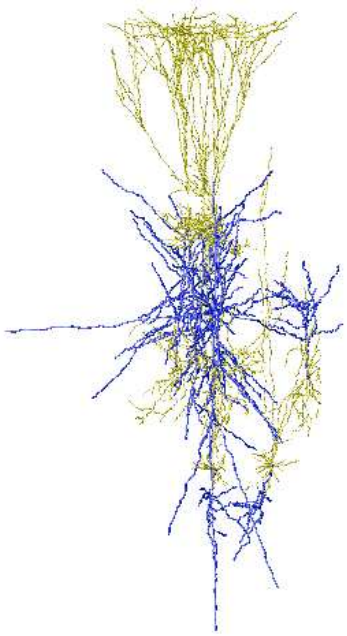


Fig. 14. Representation of cortical synaptic assembly in NeuGen

to our knowledge, provide a good way of wiring up the network. NeuGen did not scale well since it used an $O(n^2)$ algorithm (Recent modifications have reportedly improved the performance to $O(n \log(n))$). NeuroConstruct provides no means for automatic generation of putative synapses. Hence algorithms like those by Lien [23] and Kalisman [19] need to be integrated into the existing tools to make these tools more useful for realistic modeling.

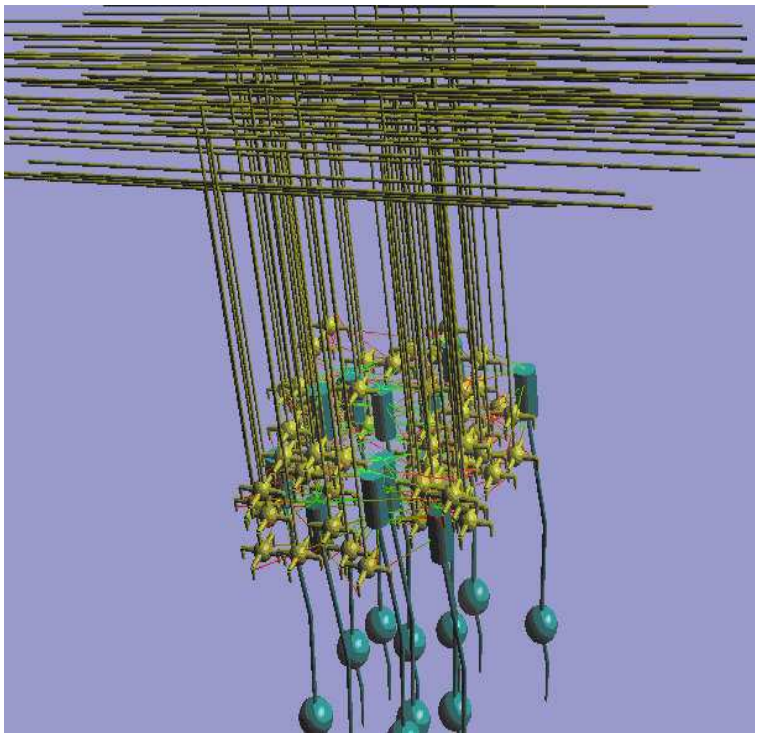


Fig. 15. Representation of cerebellar synaptic assembly in NeuroConstruct

CHAPTER V

LATTICE/GROUP

A. Introduction

Lattice/group determines how the repeating microstructures in the brain are locally arranged. A lattice is a regular, periodic configuration of points, particles, or objects throughout an area or a space. "In mathematics, a lattice (group) in R^n is a discrete subgroup of R^n which spans the real vector space R^n. Every lattice in R^n can be generated from a basis for the vector space by considering all linear combinations with integral coefficients" [38].

B. Background

1. Lattice geometry

Lattice geometry can be represented as (1) the set of all points in n-dimensional Cartesian space with integral coordinates, i.e., the space of n-dimensional vectors (x_i) whose components are integers, together with (1) a set of edges joining adjacent points. The points are the lattice "sites" and the edges are the lattice "bonds". A vector (x_i) is said to be integral if all its components are integers [29].

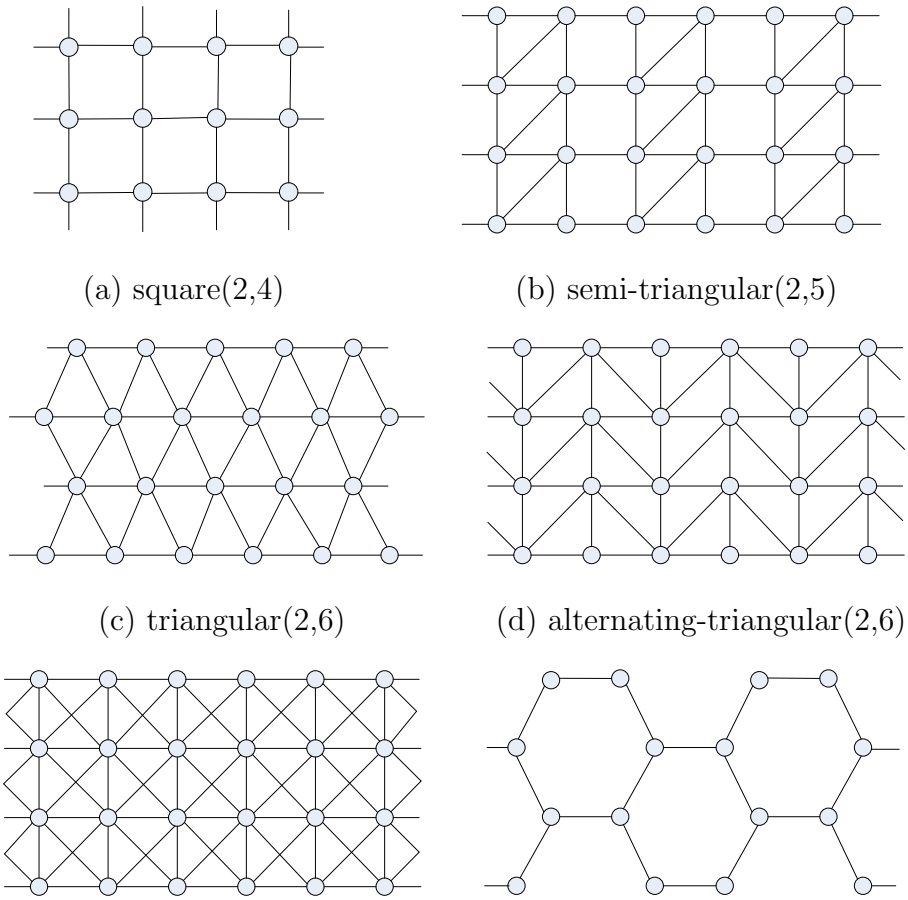
Lattice geometry can be defined on this basis by the use of a finite number (usually only one or two) of finite sets of integral vectors $\{y_i\}$, called "direction vectors", such that $-1 \leq y_i \leq +1$ for all i. See Section 2 for examples.

2. Examples of lattice geometry

The *dimensionality* of a lattice is the dimensionality of the Cartesian space used to define that lattice. Thus, a square lattice is 2-dimensional, a cubic lattice is 3-

dimensional and so on. The *coordination number* of a lattice is the size of the sets of direction vectors required for its definition.

Fig. 16 lists all the possible lattice structures in the 2D Cartesian space. For example, honeycomb(2,3) indicates that the structure has dimensionality 2 and a coordination number 3.



(a) square(2,4) (b) semi-triangular(2,5)

(c) triangular(2,6) (d) alternating-triangular(2,6)

(e) double-triangular(2,8) (f) honeycomb(2,3)

Fig. 16. List of all lattice structures in the 2D Cartesian coordinate system

In the 2D Cartesian space, the set of direction vectors for the square lattice in Fig. 17 are: (1,0), (-1,0), (0,1), (0,-1). Thus the neighbors of the site (1,1) are (2,1),

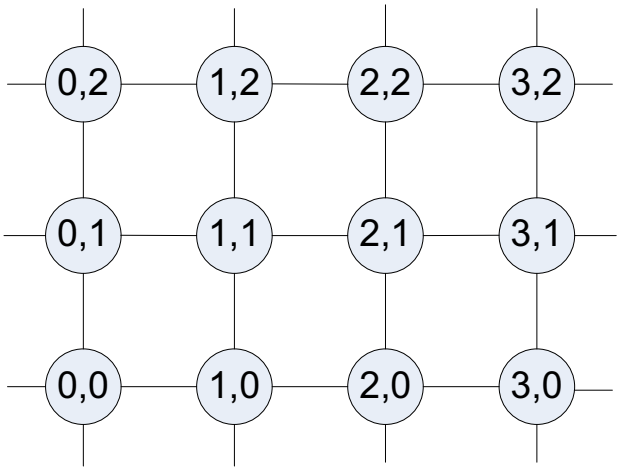


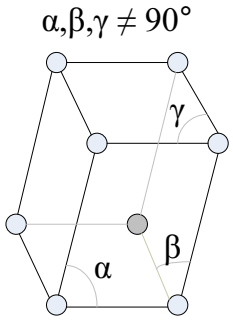
Fig. 17. Direction vectors for a square lattice in 2D Cartesian space

(0,1), (1,2), (1,0) respectively and the neighbors of the site (2,1) are (3,1), (1,1), (2,2), (2,0). Similarly, the three-dimensional cubic lattice has the direction vectors: (1,0,0), (-1,0,0), (0,1,0), (0,-1,0), (0,0,1), (0,0,-1) and the hypercubic lattice (4D) has the direction vectors: (1,0,0,0), (1,0,0,0), (0,1,0,0), (0,-1,0,0), (0,0,1,0), (0,0,-1,0), (0,0,0,1), (0,0,0,-1)

In three dimensions there are fourteen unique regular lattices called *Bravais lattices*. A Bravais lattice is a set of points constructed by translating a single point in discrete steps by a set of basis vectors [10].

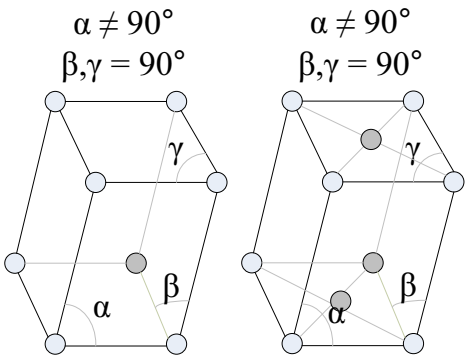
Figures 18 through 24 show fourteen 3-dimensional lattices each classified by its crystal system. In Fig. 18 a, b and, c represent the lengths of the edges and α, β and γ represent the angles between the edges.

From Fig. 16 and Fig. 18 through Fig. 24, we see that the numbers of available regular lattice models in 2D and 3D Cartesian space are limited to six and fourteen respectively. Section 3 explains how this can be used to our advantage in mining lattice structures in brain networks.



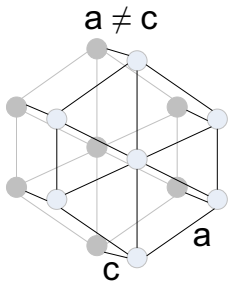
(a) triclinic

Fig. 18. List of 3D Bravais lattices: triclinic



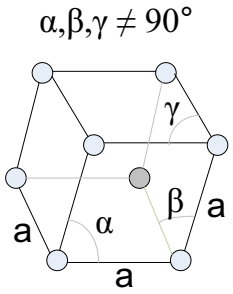
(a) simple (b) centered

Fig. 19. List of 3D Bravais lattices: monoclinic



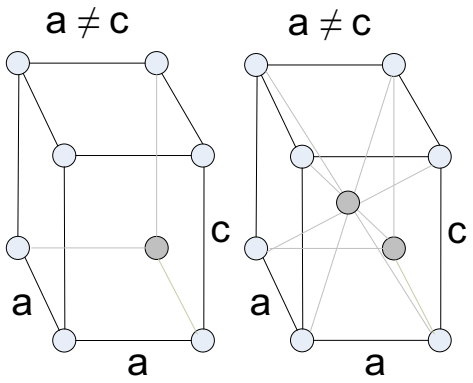
(a) hexagonal

Fig. 20. List of 3D Bravais lattices: hexagonal



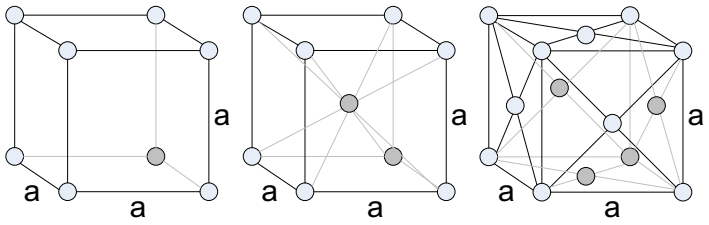
(a) rhombohedral

Fig. 21. List of 3D Bravais lattices: rhombohedral



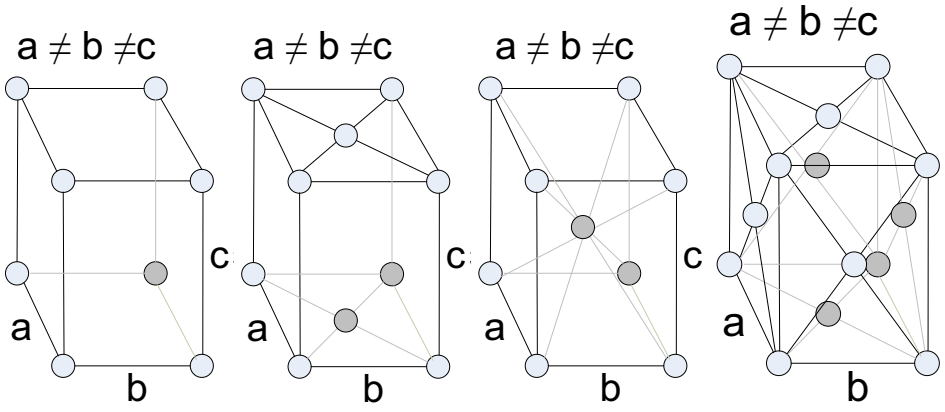
(a) simple (b) body-ctr'd

Fig. 22. List of 3D Bravais lattices: tetragonal



(a) simple (b) body-ctr'd (c) face-ctr'd

Fig. 23. List of 3D Bravais lattices: cubic



(a) simple (b) base-ctr'd (c) body-ctr'd (d) face-ctr'd

Fig. 24. List of 3D Bravais lattices: orthorhombic

C. Lattice structures in the brain

1. Lattice representation of brain networks

We propose that lattice theory can be used as a modeling tool for capturing the stochastic regularity in brain regions. A lattice has a minimum repeat module whose position forms the basis of the regular structure. Models of brain networks, where possible, should be modularized by the explicit introduction of basic circuits. This modularization imposes a stochastic lattice structure on the brain network, which should subsequently simplify parameter search and parallelization of functional computation. Shepherd's book [37] describes basic circuits for the ten better understood regions of the brain. The cell types for each basic circuit are identified; generally, cell packing parameters and lattice/group are not specified. Moreover, few, if any, wiring diagrams are given for the associated pathways, such as cortico-cortical connections, cerebellar peduncles, etc.

Fig. 25 illustrates how a single Purkinje cell forms a repeating module in the cerebellum. The lattice structure comes from the knowledge of the connectivity in-

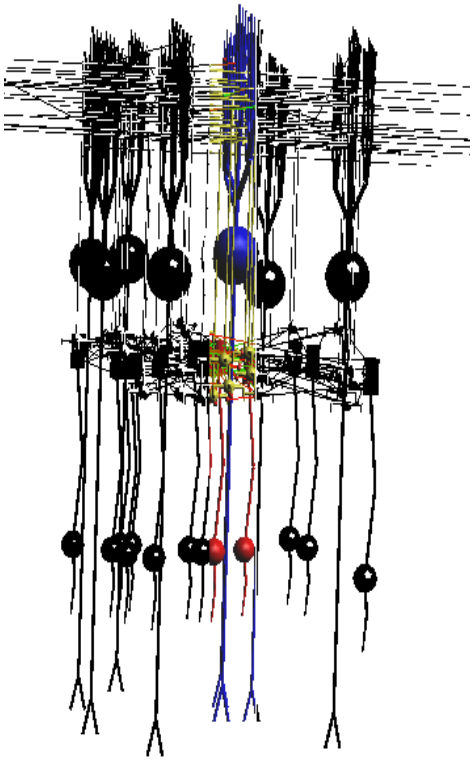


Fig. 25. Repeat module of a Purkinje cell seen in NeuroConstruct

formation.

2. Limitations in applying lattice theory to brain networks

Lattice structures are expected to show up only in selected regions of the brain that have a regular repeating structure (e.g., cerebellum). In the other brain regions, cell packing patterns might not show up at all. In that case, a lattice description is not particularly apt. Also, issues like cell migration (e.g., in the developing cerebral cortex) might wash out prenatal lattice organization. Our solution for this situation is to define the degree of conformance of a brain region to a lattice structure on a scale of *zero* to *one* (zero being a totally random distribution and one being a perfect

lattice structure). Moreover, lattices may not be the only kind of repeating structure present in a brain region. More complicated repeat patterns can be searched for using new tools like GraphEdit (suggested by Dr. Amato). However, it needs to be experimentally determined how these tools compare against the lattice search methods.

Another limitation to the straight-forward application of lattice theory to brain networks is that, in many regions the lattice structure is only locally applicable because the physical boundaries of the brain region is sufficiently curvilinear that only piecewise adherence to lattice structure can be ascertained. However, Chapter VI addresses this issue in more detail.

3. Mining lattice structures in brain regions

The Brain Networks Laboratory [26] plans to perform an exhaustive search of the various regular lattice structures (2D and 3D) in a given brain region to mine out lattice patterns. The number of available regular lattice models in 2D and 3D is very limited. As described in section 1, there are only six (fourteen) distinct regular lattice structures in 2D (3D) respectively. To our knowledge, no basic circuits in the vertebrate brain having 3D lattice structure have been identified. The conformance of a brain region to a lattice structure can be estimated by analyzing the equivalent Voronoi picture of the region (refer to Chapter III). Voronoi representations can be used to find parameters that characterize regularity.

D. Summary

In this chapter we proposed applying lattice theory to modeling and basic circuit mining in brain network. Regular lattice patterns are present all around us in the

physical world: in crystals and in the biological world: in the segmental structures commonly seen in plants and animal anatomy. So, it is naturally reasonable to expect such regularity in some regions of the brain, as indeed seen in the cerebellum, retina, striate cortex, hypocampus, and most recently in the hyporeticulum.

CHAPTER VI

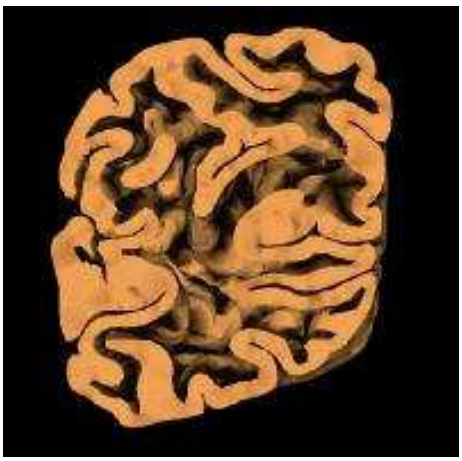
SOLID MODEL OF THE BRAIN REGION

A. Introduction

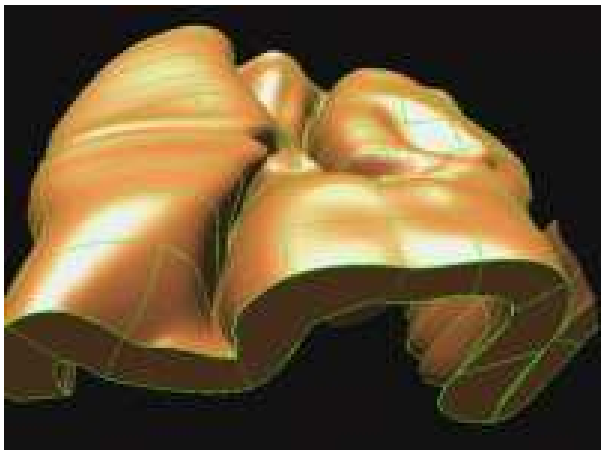
Solid model of the brain region dictates the global geometric organization of a given portion of the brain (e.g., the manifold, the convoluted sheet of tissue underlying the cerebrum). In mathematics, a manifold is a geometric space that looks locally, but not globally, like the Euclidean space R^n (e.g., surface of a sphere).

B. Solid modeling of brain regions

Most regions of the brain have complicated and convoluted geometries imposed by their development. For example, Fig. 26 presents a 3D reconstruction of the human neo-cortex.



(a) Front view of human cortex



(b) Finite-element model for part of the human neocortical shell

Fig. 26. 3-dimensional reconstruction of the human cortex [26]

None of the input parameters defined thus far in earlier chapters is capable of describing the global geometry of brain regions. The lattice models discussed in Chapter V can at best describe the local structure. Hence, a "solid model of the brain region" is required as an input parameter to the "Network Stochastic Generator" (Fig. 1) to reproduce the longer-range geometric structure in the brain region. Its boundary manifold(s), as derived from the solid model of the region, provides the needed longer-range geometric information.

C. Defining topology using manifolds

Manifolds are of interest in the study of geometry, topology, and analysis. Many properties of the Euclidian space carry over to manifolds. Manifolds arise naturally in a variety of mathematical and physical applications as "global objects". From the brain perspective, manifolds represent global versus local brain architecture.

D. Summary

In this thesis we do not quantify the manifolds that best describe each of the brain regions. Such manifolds, represented by piecewise tri-cubic splines, are under construction for the mouse brain in the 3D Mouse Brain Atlas [20]. This chapter serves as a place holder for future extensions of this thesis and to point out the void that needs to be filled in for us to be able to build realistic models of the entire brain regions.

CHAPTER VII

PROJECTION

A. Introduction

An empirically derived brain network of a mouse or other small animal is at best a "model" in that it can be constructed only from the superposition of data from multiple brains and staining technologies. The methodology to project from an observed brain network (cell distributions, their neuron morphology, and interconnection) to the topology of a modeler's network (multi-compartmental models, event-driven spike processing) is still primitive. For the brain networks we need a theory for Level of Detail (LoD), much as exists in computer graphics for objects seen up close to the same object viewed at a distance. Such a theory exists in part for morphology projection of brain networks, but it needs to be extended to the electrophysiological domain.

B. Background

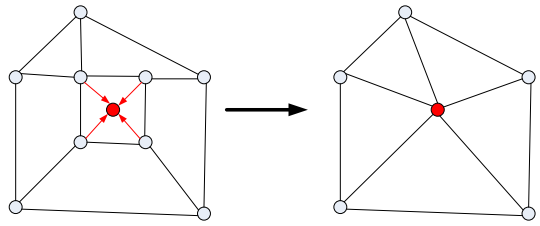
1. Projection

In mathematics, a projection p is a linear transformation that remains unchanged if applied twice: $p(u) = p(p(u))$ (i.e., it is idempotent). Such transformations project any point in the vector space to a point in the subspace that is the image of the transformation. Transforming the point (x, y, z) in three dimensions to $(x, y, 0)$ and generalizations of this transformation in other dimensions are examples of projection. Such projections are not reversible since information is lost during the linear transformation p [39].

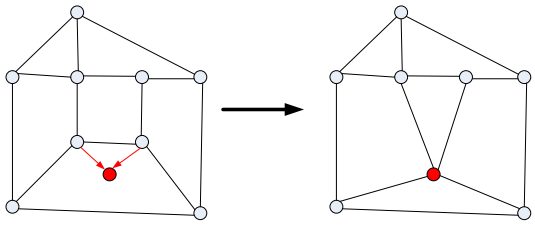
2. Level of Detail (LoD) for 3D graphics

In 1976, Clark first described the benefits of representing objects within a scene at several resolutions [24]. This concept, commonly known as managing level of detail (LoD), exploded into existence in the early 1990s with many algorithms, papers, and software tools were devoted to generating and managing such multi-resolution representations of objects automatically. These methods simplify the polygonal geometry of small, distant, or otherwise unimportant portions of the model, seeking to reduce the rendering cost without a significant loss in the scene's visual content [24].

Particularly relevant to brain modelers (among LoD algorithms) are the vertex-merging schemes operated by collapsing two or more vertices of a triangulated model together into a single vertex, which in turn can be merged with other vertices. Merging a triangle's corners eliminates it, decreasing the total polygon count as shown in Fig. 27(a).



(a) vertex merging



(b) edge collapsing

Fig. 27. Examples of "vertex merging" and "edge collapsing" algorithms

Edge-collapse algorithms that always merge two vertices sharing an edge, preserve local topology (see Fig. 27(b)), but algorithms permitting general vertex-merge operations can modify topology and aggregate objects, enabling drastic simplification of complex objects and assemblies of objects.

Both vertex merging and edge collapsing algorithms can be applied in the context of brain networks. The morphology of a neuron can be represented using a tree representation, which is a special case of a general graph representation for which the above algorithms are applicable (see Fig. 28).

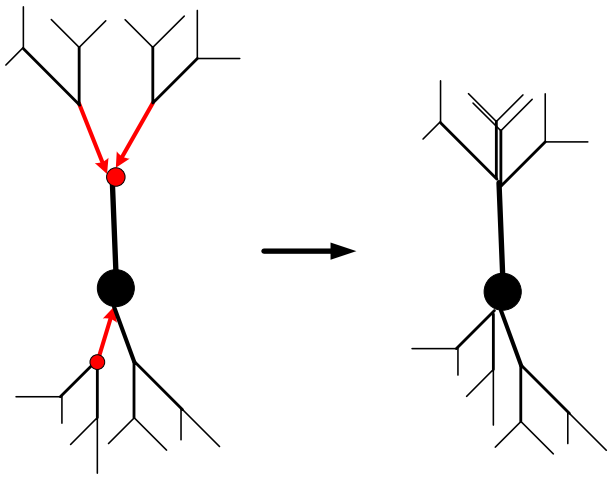


Fig. 28. Vertex merging and edge collapsing algorithms applied in the context of neuron morphology

C. Projections in brain networks

1. Criteria for good projection in brain networks

Surprisingly little work is available on what constitutes a good projection for a brain network. The two main criteria that need to be considered when projecting neurons are 1) geometric and 2) physiological. We need a mathematical framework that en-

ables us to connect the morphological and electrical structure of the neuron to its function. Based on insights gained through investigation of detailed neuron models (for example, from LM reconstructions at the BNL), we can attempt to devise methods to systematically reduce the complexity of the neuron model while preserving its essential input/output properties (branches, synapses and channels). For example, the projection from XwebDB to YwebDB (refer Fig. 4) depends on the definition of Y; i.e, the level of detail required in Y and also whether morphology, physiology or both is important. These reduced, yet biologically realistic, neuron models then serve as the basic unit for models of large networks of neurons. Such projections are required to search for regular lattice patterns. Generally, it is not computationally feasible to perform such searches on a full-blown model.

2. Compartmental models

NeuronDB provides a general framework for defining the physiology and morphology (to a very limited extent) of different neuron types using the canonical forms shown in Fig. 29 .

Fig. 29(a) is a cell with no axon or basal dendrites. Fig. 29(b) is a cell with single dendrite or multi-polar cell and Fig. 29(c) is a cell with apical and basal dendrites. The abbreviations for the canonical parts of a neuron are: D, dendrite; S, soma (cell body); AH, axon hillock (initial segment of the axon); A, axon; and T, axon terminal. The simplest case is a single dendrite, represented as an equivalent cylinder consisting of a chain of three compartments designated as (p) proximal, (m) middle, and (d) distal with respect to the cell body.

There are some existing tools that project detailed structures into simplified compartmental models. Fig. 30(a) shows a detailed multi-compartmental model of a cerebellar purkinje cell, created with GENESIS by De Schutter et al. [4]. Visual-

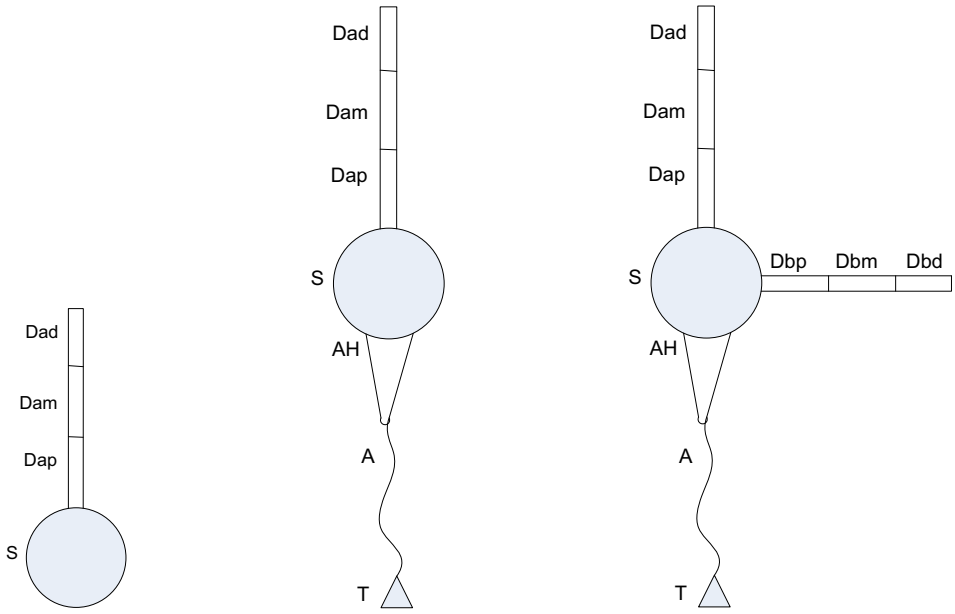
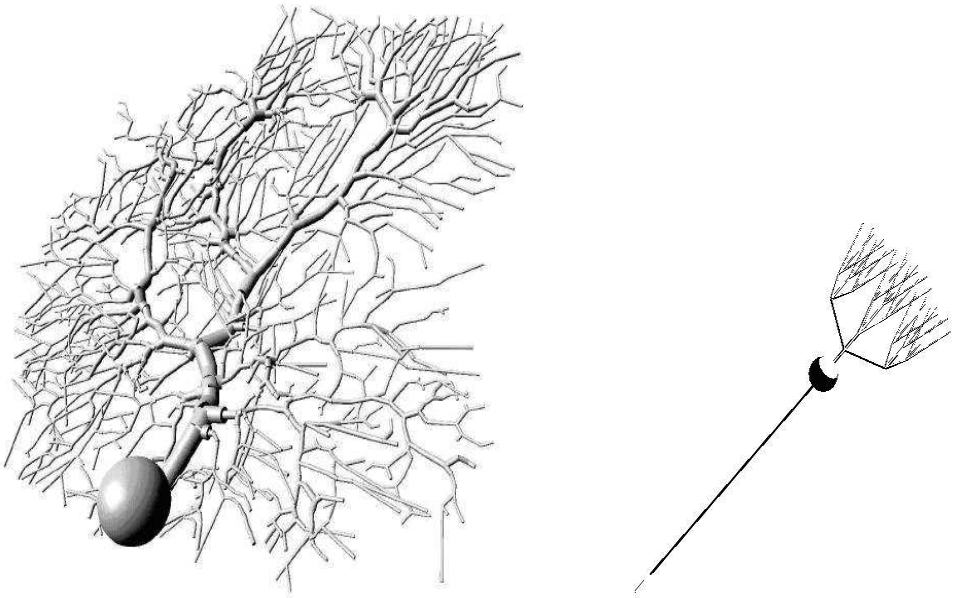


Fig. 29. Canonical forms as described in NeuronDB [32].

ization is by Leigh using the GENESIS visualizer program. The experimental data describing the cell morphology was provided by Rapp et al. [4]. Fig. 30(b) represents a much simpler representation of a purkinje cell that looks like a pitchfork with fewer compartments (generated using NeuroConstruct).

Fig. 31 is a NeuroConstruct visualization of a network for cells in the cerebellum. The choice for projection of a cell type is not purely a function of the cell type but also of its synaptic assembly with other neurons in the basic circuit. For example, in Fig. 31 the cerebellum module uses a pitchfork projection of the purkinje cells to accommodate a representative number of bipolar cells.



(a) Complex Purkinje [4]

(b) NeuroConstruct projection of Purkinje

Fig. 30. Complex and simplified Purkinje cell representations

3. LoD in brain networks

In Fig. 32 , the anatomical model of an individual neuron (Fig. 32(a)) is projected onto its stick figure representation (Fig. 32(b)), and wired into a network of other neurons (Fig. 32(c)) through the projection of the inter-neuronal connections (synapses).

D. Summary

Level of Detail (LoD), in computer graphics has broad applications and is a well researched area. However, the concept has not been systematically applied in the context of brain network modeling. In this chapter, we propose some ideas in this direction. Some of the existing LoD algorithms can be extended so they can be used for neuron morphology and physiology projection. Such algorithm development would

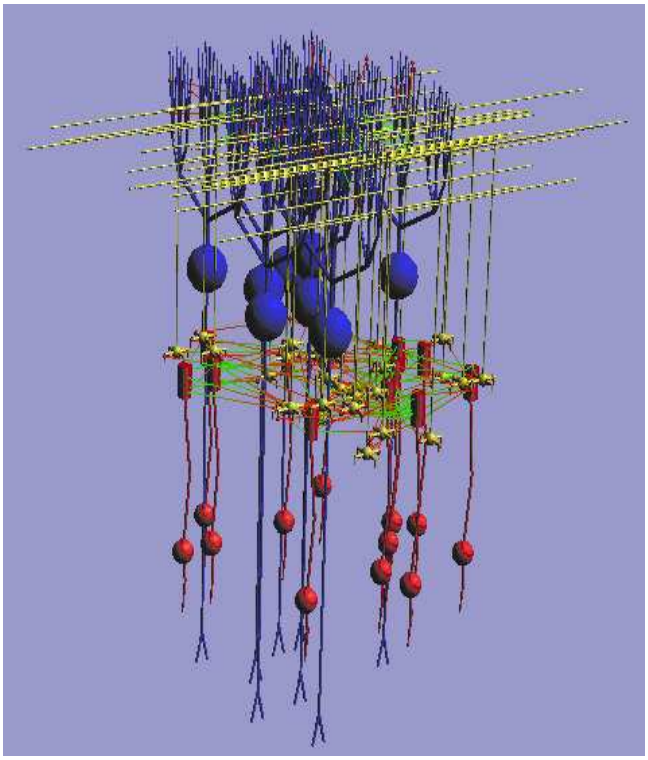
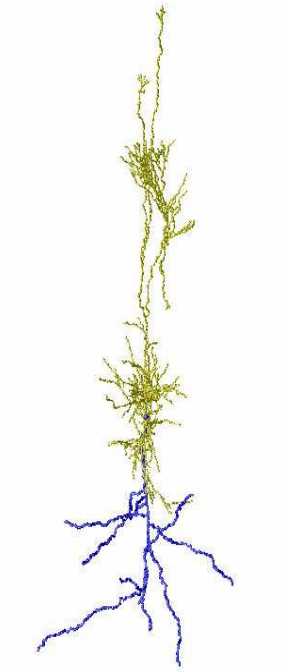
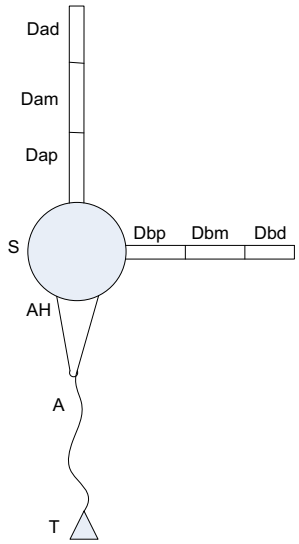


Fig. 31. NeuroConstruct visualization of the cerebellum [15]

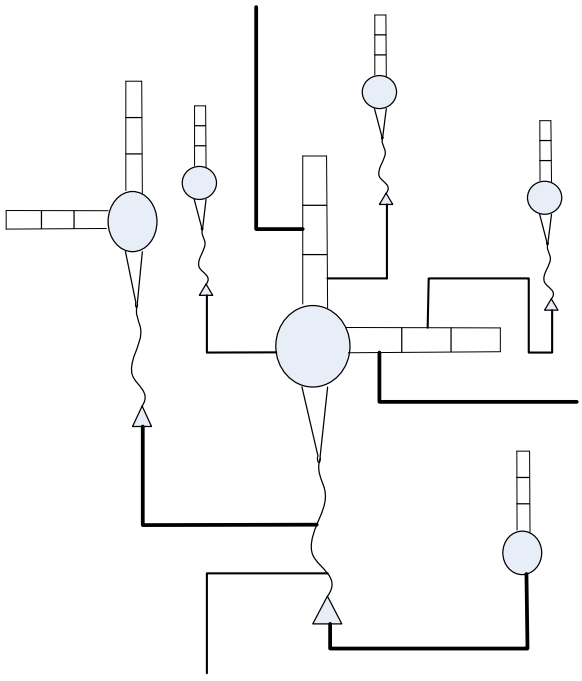
constitute an important future extension to this thesis.



(a) Anatomical model [12]



(b) Compartmental model. Adapted from [32]



(c) Neuronal network. Adapted from [37]

Fig. 32. LoD in brain networks

CHAPTER VIII

IMPLEMENTATION

A. Introduction

In this chapter, we discuss the implementation details of our *Circuits* module in NeuGen [12], illustrate how one can build small networks of simplified neurons using NeuroConstruct [15], and how detailed morphologies described in other tools can be imported into NeuroConstruct.

B. Circuits in NeuGen

We extended NeuGen to support basic circuits. To this end, we made some design changes and modified the NeuGen software. The following subsections cover these design changes and modifications.

1. Extension to *Neuron* class

We extended the *Neuron* class in NeuGen to support new neuron types that occur in basic circuits. For example, NeuGen currently supports four types of neurons namely L4 spiny stellate neurons, L2/3 pyramidal cells, L5 pyramidal cells, and L4 star pyramidal cells. We added support for a new neuron type, the "smooth neuron", which is a representative of the chandlier, basket, and other inhibitory cell types present in the cortex.

Fig. 33 illustrates the current neuron class hierarchy in NeuGen. The cell types: Spiny Neuron, Starpyramidal Neuron and Smooth Neuron inherit directly from the neuron class. L23pyramidal and L5pyramidal Neuron types inherit form the Pyramidal Neuron class which in turn inherits from the Neuron class. We currently use

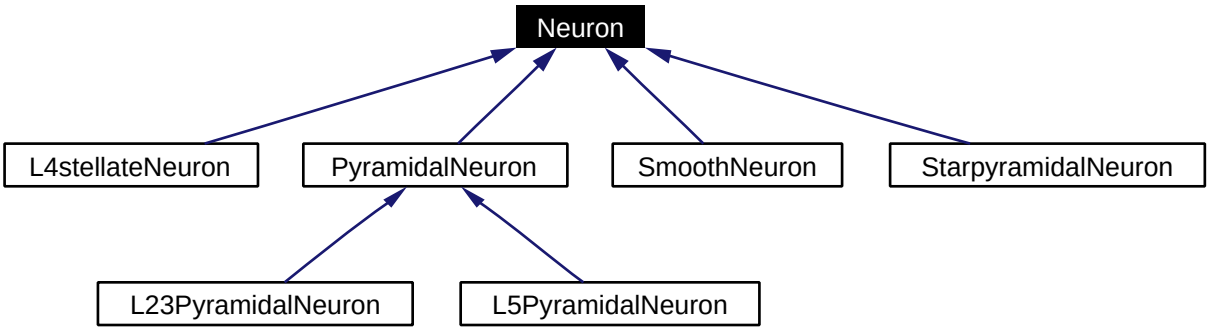


Fig. 33. *Neuron* class hierarchy in NeuGen [12]

dummy parameters for the Smooth Neuron. But, one needs to make changes only to the *param.neu* file to model this neuron type better.

2. Definition of *Circuit* class

We implemented a new *Circuit* class in NeuGen to support basic circuits. The *circuit* class is at a hierarchial level between the *Net* class and the *Neuron* class. Fig. 34 represents the class design of the *Circuit* class. The *Circuit* class contains a list of neurons, connections (synapses), and is defined by a set of parameters known as *CircuitParams*. Section 4 describes the circuit parameters in detail.

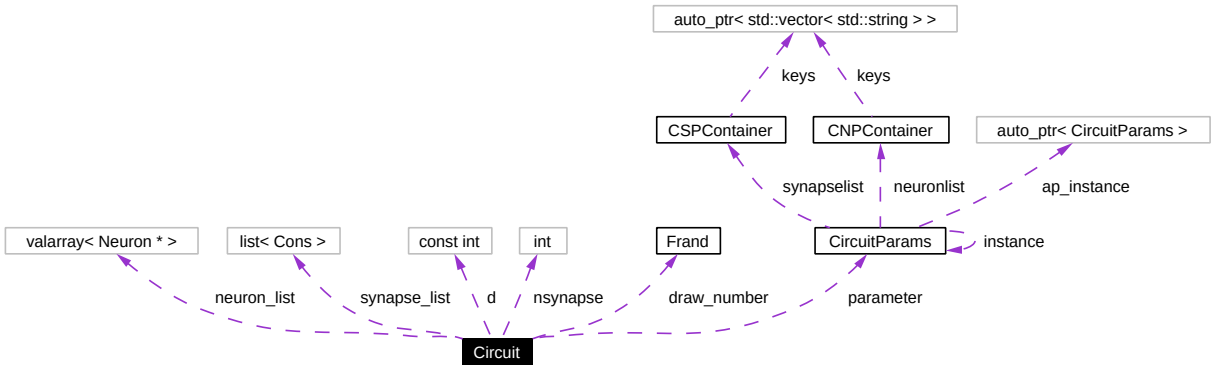


Fig. 34. *Circuit* class implementation in NeuGen

We modified the existing *Net* class such that it now has a list of *Basic Circuit*s and a list of *Neuron*s. The *Basic Circuit* in turn consists of a list of its constituent *Neuron*s. The new *Net* class hierarchy is illustrated in Fig. 35 .

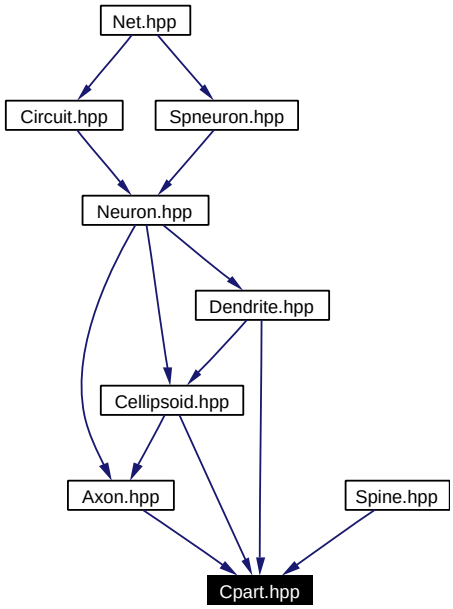


Fig. 35. The new *Net* class hierarchy in NeuGen. The *Circuit* class has been added at a hierarchial level between the *Net* class and the *Neuron* class

Fig. 35 shows the dependencies between the header files to illustrate the new network class hierarchy in NeuGen. *Spneuron.hpp* file contains definitions of specialized neurons types like those described in Section 1. The file *Cellipsoid.hpp* contains definition of the soma and the file *Cpart.hpp* contains definition of the segments. The files *Dendrite.hpp*, *Axon.hpp*, and *Spine.hpp* contain definitions of the dendrites, the axons, and the spines respectively.

3. Definition of *CircuitParams* class

We defined new classes to represent the circuit parameters. Fig. 36 shows the class hierarchy of *CircuitParams* class which contains the list of synapses in the *CSPContainer* class and the list of neurons in the *CNPContainer* class. The neuron parameters are defined by the *CircSomaParam* class and the synapse parameters are defined by the *CircSynapticParam* class. Fig. 37 illustrates the details of these classes.

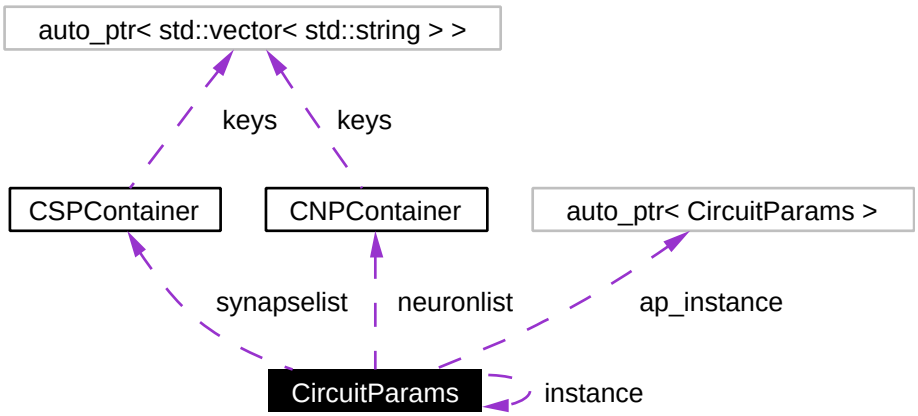
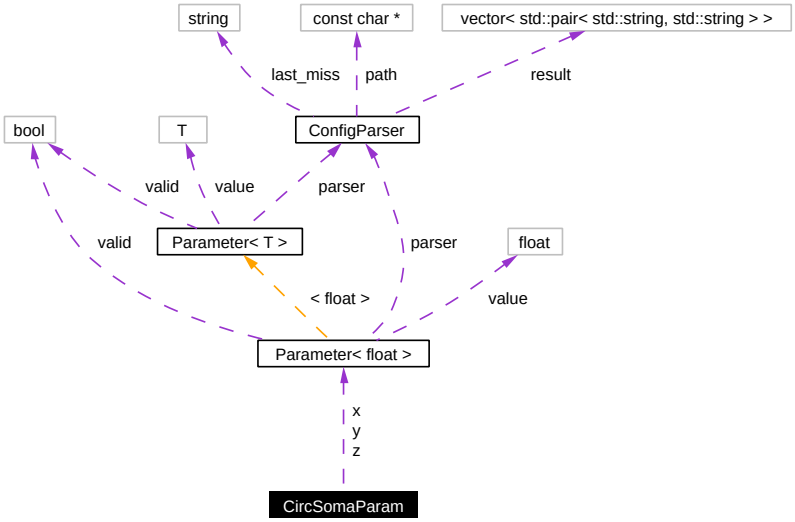
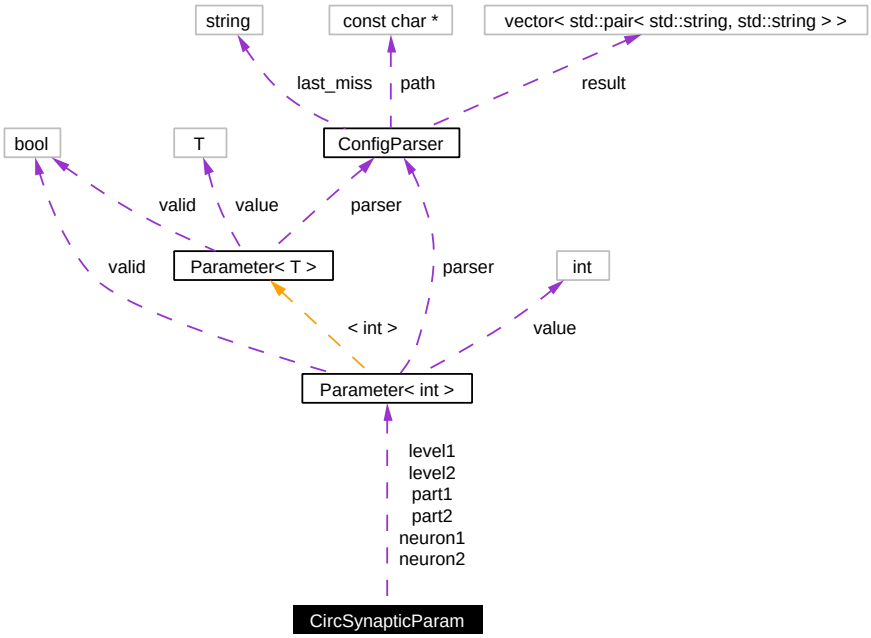


Fig. 36. *CircuitParams* class hierarchy



(a) Neuron parameters



(b) Synapse parameters

Fig. 37. *CircSomaParam* and *CircSynapseParam* classes in NeuGen

The *CircSomaParam* class needs the (x,y,z) coordinates of the cell's soma while the *CircSynapticParam* class needs the neuron ID, part, and level parameters of the two neurons involved in the synapse. A detailed description of these input parameters

is given in Section 4, which explains the specifics of the input parameters.

4. Definition of *Circuit.neu* configuration file

We defined a new configuration file called "circuits.neu" to specify the circuit parameters. An example "circuits.neu" file is shown in Table III.

Table III. Configuration file for a simple circuit in NeuGen

```
Circuit
    neuronlist
        neuron1
            id 0
            type starpyramidal
            soma0
                x 500
                y 300
                z 500
        neuron2
            id 1
            type smooth
            soma0
                x 650
                y 300
                z 700

    synapselist
        synapse1
            id 0
            type 0
            strength 5.0
            synaptic_param
                neuron1 1
                part1 2   # parts 0-> Soma, 1->Dendrite and 2-> Axon
                level1 0  # Level 0-> proximal, 1-> middle, and 2-> distal
                neuron2 0
                part2 1
                level2 0
        synapse2
            id 1
            type 1
            strength 7.0
```

```
synaptic_param
    neuron1 1
    part1 2
    level1 2
    neuron2 0
    part2 2
    level2 1
```

In Table III, the *neuronlist* entry lists all the neurons present in the circuit, their identification numbers, and their soma locations. We specify the soma locations based on heuristics or intuition. The neuron's dendritic and axonal arbors grow from that location. When two neurons need to be connected, we grow a synapse "vector" between them. Even though, it violates "proximity labeling", this is the only solution to connect two neurons which are placed spatially afar. The *synapselist* entry specifies the synapse vectors between the neurons from the *neuronlist*. Each synapse has an identification number, type (inhibitory or excitatory), strength, and synaptic parameters (*synaptic_param*).

The *synaptic_param* field specifies the source segment and the destination segment of the synapse. The source or destination segment is identified by three hierarchial parameters, namely: 1) *neuronID*, 2) *part* within a neuron, and 3) *level* within a part of the neuron. A neuron is identified by the identification number from the *neuronlist*. The *part* of a neuron is identified by a number. Part number "0" indicates that the part is a soma, the number "1" indicates a dendrite, and number "2" indicates an axon.

Similarly, the *level* within a part is identified by a number. Level number "0" represents the proximal (to the soma) segments of the part. Levels "1" and "2" represent the middle and distal parts respectively. In our implementation, we selected the first segment in a *level* as the representative segment of that *level*. Each *part* in the

neuron is divided into different *level*s each containing an equal number of segments. We currently implemented the following types of synapses in the code: axo-dendritic, axo-axonic, axo-somatic and dendro-dendritic. Other types of synapses, if found in the literature, can be easily added into the new class definition at a later point in time. Also, to make a connection between two distant neurons, code can be added to move the neurons closer or extend the axonic arbor to make a connection.

5. Definition of *CircuitParser* class

We added the *CircuitParser* class (which inherits from the *Configparser* class) to read and parse the *circuits.neu* configuration file described in Section 4 .

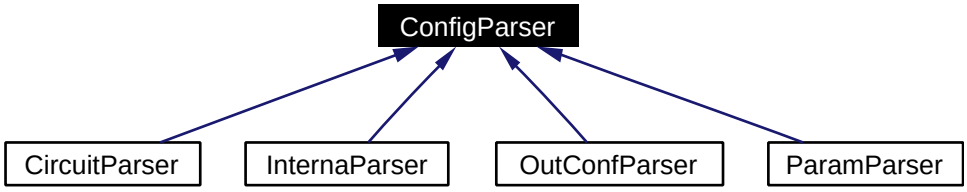


Fig. 38. Configuration parser classes in NeuGen

Each parser shown in Fig. 38 feeds into the *ConfigParser* class. The *InternaParser* class reads in from *Interna.neu*, which is a configuration file for expert users. *OutConfParser* class reads from the *OutputOptions.neu* file, which contains output options for the NeuGen software. The *ParamParser* class reads in from the *Param.neu* file, which contains the proximity labeling, neuron population, and neuron morphology description parameters.

6. Modification to OpenDX visualization output

We modified the *writetoDX* function in the *Net* class to handle visualization for circuits. Fig. 39 shows an example basic circuit containing two neurons and two

synapses between them. The figure is a visualization generated using OpenDX[35] from the configuration file described in Section 4. The neuron on the right is *neuron0* and the one on the left is *neuron1*. There is an axo-dendritic synapse between the axon of *neuron1* (*level0*) to the dendrite of *neuron0* (*level0*). The second synapse is of axo-axonic type between the axon of *neuron1* (*level2*) to the axon of *neuron0* (*level1*).

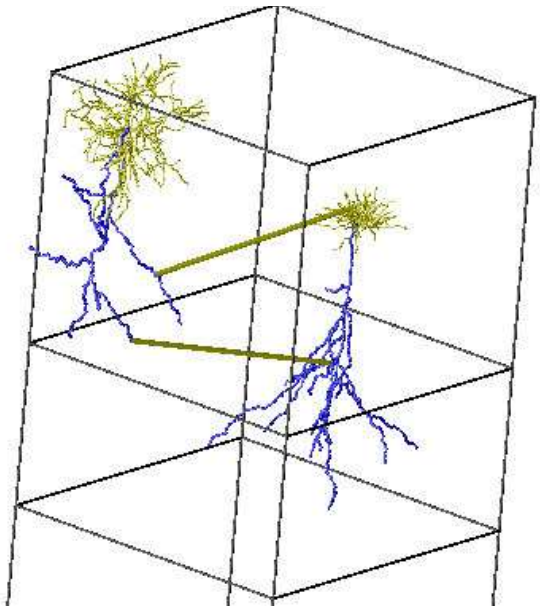


Fig. 39. A simple basic circuit in NeuGen which has one axo-axonic and one axo-dendritic synapse

7. Modification to NEURON simulation output

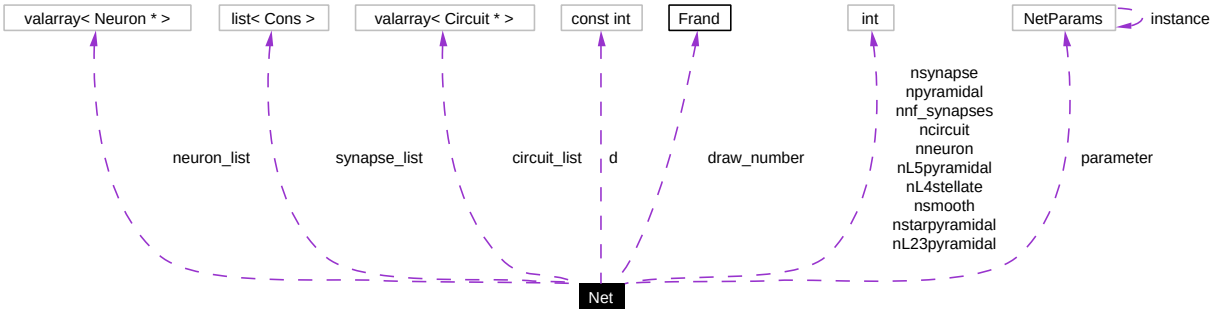
We modified the *writetoHocfile* function in the *Net* class to handle NEURON simulations for circuits. Fig. 40 shows the visualization of the simulation generated from the configuration file described in Section 4. NEURON simulation output (*.hoc* files) generated by NeuGen contains default electrophysiological parameters. These param-

eters can be adjusted through the NEURON GUI for running the actual simulations.

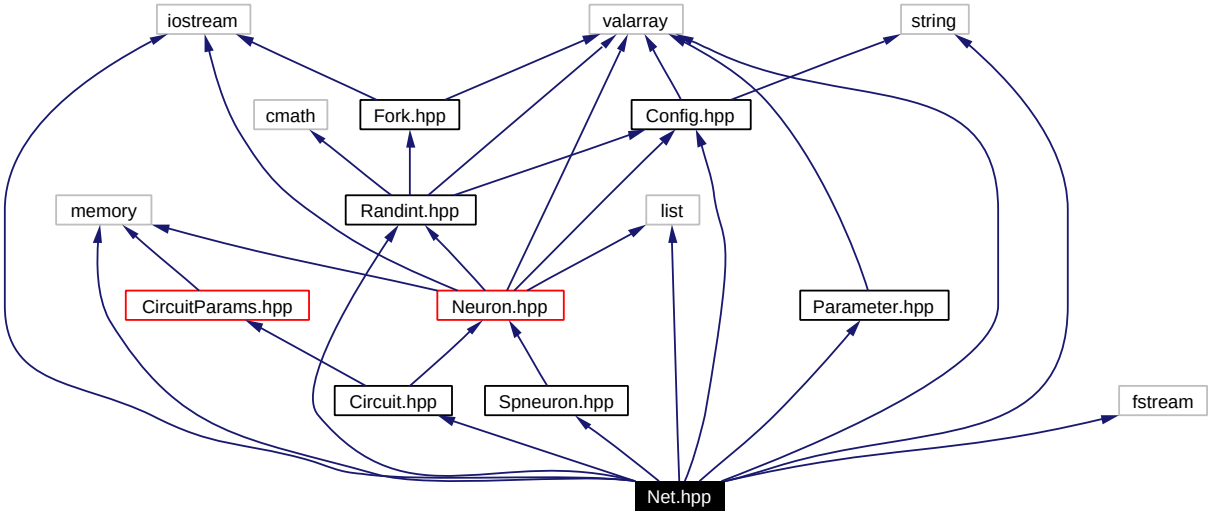
Fig. 40. A simple basic circuit generated by NeuGen, visualized in the NEURON simulator (the synapses are not explicitly shown)

8. Network of basic circuits

The architecture of the *Net* class, after including the circuits, looks like the description in Fig. 41.



(a) *Net* class parameters



(b) *Net* class

Fig. 41. The new *Net* class dependency chart in NeuGen

The broader problem now is, how does one grow the entire network from the basic circuits? The simple strategy of stochastically placing these circuit templates and connecting them is obviously flawed because many other factors need to be considered to build a realistic model. The lattice theory (Chapter V) and solid model (Chapter VI) concepts need to be used here to get a realistic growth model. This part is a possible future extension of this thesis. The framework we proposed in this thesis for building lattice models of the brain (see Fig. 1) , lays down the guidelines for the

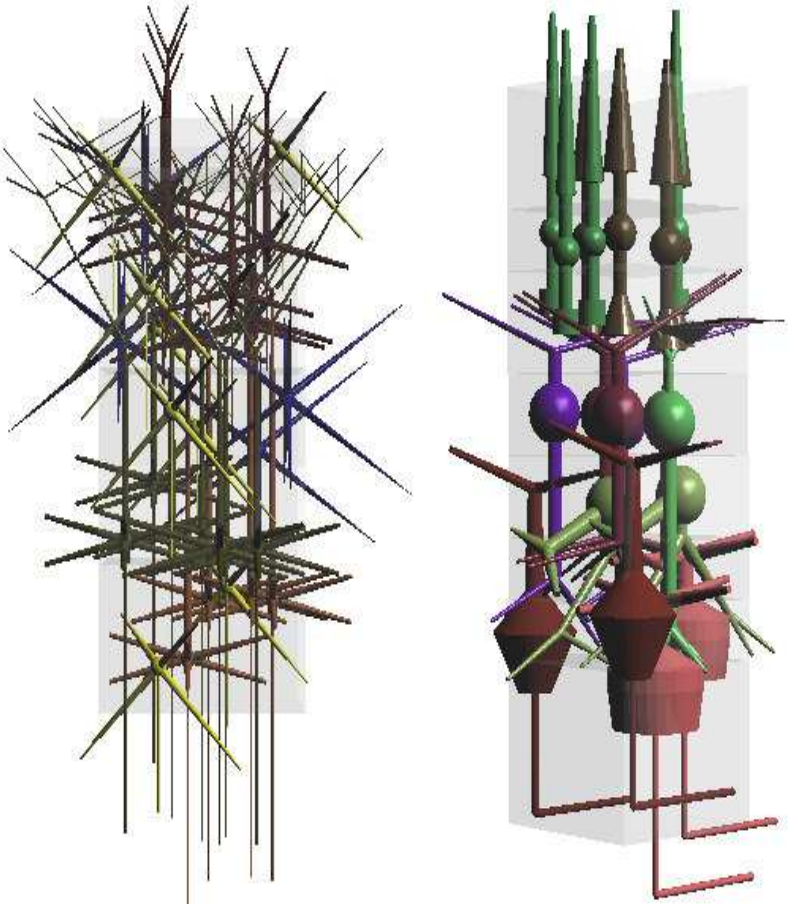
extensions.

C. Circuits in NeuroConstruct

NeuroConstruct[15] is a useful tool for building 3D network models. This section illustrates: 1) how one can build small networks of simplified neurons using NeuroConstruct and 2) how detailed morphologies described in other tools can be imported into NeuroConstruct.

1. Sample circuits

We built some circuit models to explore NeuroConstruct's ability to model neuronal circuits. Fig. 42 represents the morphological network model of two regions of the brain: the neocortex and the retina. For the neocortex (shown in Fig. 42(a)), the defining basic circuit was taken from Fig. 12.15 of [37] and for the retina (shown in Fig. 42(b)), the defining basic circuit was taken from Fig. 1.13a of [37]. The figures also illustrate NeuroConstruct's 3D Java visualization capabilities.



(a) Cortex (b) Retina

Fig. 42. Network representations of two brain regions: cortex and retina, implemented and visualized in NeuroConstruct

2. Models imported from NeuGen

The morphology information generated in NeuGen can be imported into NeuroConstruct via the NEURON output generated by NeuGen. The example in Fig. 43 shows one such cell imported into and visualized in NeuroConstruct. The morphology was first generated by NeuGen and then imported through the *NeuronMorpholgyReader* which reads NEURON's *.hoc* files into NeuroConstruct.

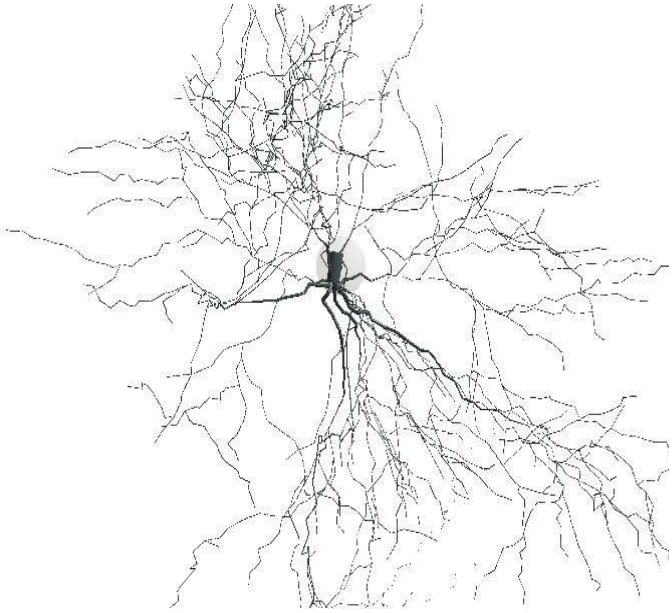


Fig. 43. Cell morphology visualization in NeuroConstruct of a cell generated in NeuGen

D. Summary

NeuGen, in its current form, can only build a network of individual neurons. We have extended the tool to handle basic circuits. Extending the tool further to support a network built by the "wiring up" of these basic circuits is still under development. NeuroConstruct is another useful tool for building small networks of simplified neurons. But the tool, in its current beta form, doesn't scale well for morphologically accurate neurons or for large networks. The framework that we discussed in this thesis for building biologically accurate brain networks can help in extending both NeuGen and NeuroConstruct to overcome their limitations.

CHAPTER IX

CONCLUSIONS AND FUTURE WORK

The major contribution of this thesis has been to place on a more rigorous basis the concept of *basic circuit*, as the fundamental replicating modules of mammalian brain microstructure. The search for basic circuits (basic circuit mining) has been transformed into the identification of the 1) cell types, 2) cell packing, 3) connection map, 4) lattice/group, and 5) solid model defining the replicated microstructure of a brain region. Methodologies for identifying each of the above five constituent parameter groups defining the brain network in a given mammalian brain region have been elucidated. This chapter concludes by briefly pulling together the suggestions for future work made throughout the thesis.

1. No current tool fully supports a brain network by the "stochastic wiring up" of basic circuits. For example, NeuGen in its current form, can only build a network of individual neurons. We have extended the tool to handle basic circuits. Extending the tool further is still under development.

2. We must accurately characterize the spatial distribution of the various types of neurons in order to build realistic models of the brain. Commonly used distributions (e.g., uniform, random) are limited in their ability to capture the spatial distribution in sufficient detail. Hence, we propose that Voronoi diagrams be used to quantitatively classify cell packing patterns for modeling the brain networks.

3. None of the existing brain modeling tools provide an accurate way of wiring up the network. The limitation of NeuGen [12] is that it did not scale well since it used an $O(n^2)$ algorithm (recent modifications have reportedly improved

the performance to $O(n \log(n)))$. NeuroConstruct [15] provides no means for automatic generation of putative synapses. Hence, algorithms like those by Lien et al. [23] and Kalisman et al. [19] need to be integrated into the existing tools to make these tools more useful for realistic modeling.

4. Regular lattice patterns are present all around us in the physical world (e.g., crystals) and biological world (e.g., plants, segmental architecture of vertebrates). So, it is only natural to expect such regularity in the brain. A *lattice/group* can determine how a repeating microstructure in the brain is locally assembled into a cortical area or brain region. If only selected regions in the brain exhibit lattice structure, this finding can go a long way in the simplification of that region's computational model.

5. Most regions of the brain have complicated and convoluted geometries imposed by their neurodevelopment. While lattice models can describe the local structure efficiently, a *Solid model of the brain region* (e.g., the manifold, the convoluted sheet underlying the cerebrum) is needed to reproduce the more global geometric structure in the brain region.

6. A theory for Level of Detail (LoD), much as exists in computer graphics, is badly needed for projection in brain networks. Such a theory exists in part for morphology projection of brain networks, but needs to be extended to the electrophysiological domain. Some of the existing LoD algorithms in graphics can be extended so they can be used for neuron morphology and physiology projection. Such algorithm development would constitute a major extension to this thesis.

Paraphrasing Cajal again [36], to grasp the architectural plan of the brain, we need to first generate a clear and accurate view of the structure of the relevant brain centers. We hope that the ideas discussed in this thesis contribute towards that grand vision.

REFERENCES

[1] G. Ascoli, "Progress and perspectives in computational neuroanatomy, invited review," *Anatom. Rec.*, vol. 257(6), pp. 195–207, 1999. [Online]. Available: http://krasnow.gmu.edu/L-Neuron/

[2] G. Ascoli and J. Krichmar, "L-Neuron: A modeling tool for the efficient generation and parsimonious description of dendritic morphology," *Neurocomputing*, vol. 32–33, pp. 1003–1011, 2000.

[3] T. Binzegger, R. Douglas, and K. Martin, "A quantitative map of the circuit of cat primary visual cortex," *J. Neurosci.*, vol. 24, pp. 8441–8453, 2004.

[4] J. Bower and D. Beeman, Eds., *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: Springer-Verlag, 1998. [Online]. Available: http://www.genesis-sim.org/GENESIS/

[5] BrainML. (2005, Jun. 12) Weill Medical College of Cornell University. Ithaca, NY. [Online]. Available: http://brainml.org/

[6] V. Braitenberg and A. Schuz, *Cortex: Statistics and Geometry of Neuronal Connectivity*, 2nd ed. Berlin: Springer-Verlag, 1998.

[7] R. Cannon. (2005, Aug. 17) Cvapp morphology conversion and editing software. [Online]. Available: http://www.compneuro.org/CDROM/docs/cvapp.html

[8] ——. (2005, May. 25) An online archive of reconstructed hippocampal neurons. University of Southampton. Southampton,UK. [Online]. Available: http://www.cns.soton.ac.uk

[9] CGAL. (2005, Jul. 20) Computational Geometry Algorithms Library. [Online]. Available: http://www.cgal.org

[10] J. Conway and N. Sloane, Eds., *Sphere Packings, Lattices and Groups*, 3rd ed. New York: Springer-Verlag, 1998.

[11] W. Denk and H. Horstmann, "Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure," *PLoS Biology*, vol. 2, p. 329, 2004.

[12] J. Eberhard. (2005, Aug. 17) Neugen - A generator for realistic neurons in 3d. online. University of Hiedelberg. Hiedelberg, Germany. [Online]. Available: http://neugen.uni-hd.de/

[13] J. Fallon, University of California, Irvine, CA, 2002, Personal communication.

[14] D. Gardner. (2005, Jun. 12) Neurodatabase. Weill Medical College of Cornell University. Ithaca, NY. [Online]. Available: http://www.neurodatabase.org

[15] P. Gleeson. (2005, Aug. 17) NeuroConstruct - A biophysical neural network modeling software. University College London. London, U.K. [Online]. Available: http://www.physiol.ucl.ac.uk/research/silver_a/neuroConstruct/

[16] J. Goodman and J. O'Rourke, Eds., *Handbook of discrete and computational geometry*. Boca Raton, FL: CRC Press, Inc., 1998.

[17] K. Harris. (2005, Aug. 10) Synapse web. Medical College. Georgia. [Online]. Available: http://synapses.mcg.edu/anatomy/chemical/synapse.stm

[18] M. Hines and N. Carnevale, "The neuron simulation environment," *Neural Comput.*, vol. 9, pp. 1179–1209, 1997. [Online]. Available: http://www.neuron.yale.edu/neuron/

[19] N. Kalisman, G. Silberberg, and H. Markram, "Deriving physical connectivity from neuronal morphology," *Biol Cybern*, vol. 88, pp. 210–218, 2003.

[20] W. Koh and B. McCormick, "Brain microstructure database system: An exoskeleton to 3d reconstruction and modeling," *Neurocomputing*, vol. 44–46, pp. 1099–1105, 2002.

[21] R. Kötter, Ed., *Neuroscience Databases: A Practical Guide.* Boston: Kluwer Academic Publishers, 2002.

[22] R. Kötter. (2005, Jul. 20) Cocomac - collations of connectivity data on the macaque brain. Brain Research Institute. Düsseldorf, Germany. [Online]. Available: http://www.cocomac.org/

[23] J.-M. Lien, M. Morales, and N. Amato, "Neuron PRM: A framework for constructing cortical networks," *Neurocomputing*, vol. 52–54, pp. 191–197, 2003.

[24] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, Eds., *Level of Detail for 3D Graphics.* San Francisco: Morgan-Kaufmann Publishers, July 2002. [Online]. Available: http://lodbook.com/

[25] D. Mayerich, Texas A&M University, College Station,TX, Aug. 2005, Personal communication.

[26] MBW. (2005, Aug. 17) Brain Networks Lab. Texas A&M University. College Station, TX, USA. [Online]. Available: http://research.cs.tamu.edu/bnl/

[27] B. McCormick, "System & method for imaging an object," U.S. Patent 09/948, 2003, for Knife-Edge Scanning.

[28] B. McCormick and K. Mulchandani, "L-system modeling of neurons," in *Proc. Visualization in Biomedical Computing*, vol. 2359, 1994, pp. 693–705.

[29] P. Meyer. (2005, Feb. 15) Lattice geometries. MediaWiki. [Online]. Available: http://www.hermetic.ch/compsci/lattgeom.htm

[30] ModelDB. (2005, Aug. 10) Shepherd lab. Yale University. [Online]. Available: http://senselab.med.yale.edu/senselab/ModelDB/

[31] MorphML. (2005, Jun. 20) An xml application for neuronal morphology data. [Online]. Available: http://www.morphml.org

[32] NeuronDB. (2005, Sep. 25) Brain database research. Yale University. [Online]. Available: http://senselab.med.yale.edu/senselab/NeuronDB/

[33] N.Goddard, M.Hucka, F.Howell, H.Cornelis, K.Skankar, and D.Beeman, "L-Neuron: A modeling tool for the efficient generation and parsimonious description of dendritic morphology," *Phil. Trans. Royal Society*, vol. series B, pp. 356–1412:1209–1228, 2001. [Online]. Available: http://www.neuroml.org

[34] N.T.Carnevale and M.L.Hines, "Cellbuilder," 2000. [Online]. Available: www.cs.stir.ac.uk/courses/31YF/Neuron/cbtut.pdf

[35] OpenDX. (2005, Sep. 25) Visualization data explorer user's guide. IBM. [Online]. Available: http://www.opendx.org/

[36] S. Ramón y Cajal, *Histology of the Nervous System.* New York: Oxford, 1995, N. Swanson and L. W. Swanson, Trans.

[37] G. Shepherd, Ed., *The Synaptic Organization of the Brain*, 5th ed. New York: Oxford University Press, 2003.

[38] Wikipedia. (2005, Aug. 10) Lattice and Group. MediaWiki. [Online]. Available: http://en.wikipedia.org/wiki/Latticegroup

[39] ——. (2005, Mar. 17) Projection. MediaWiki. [Online]. Available: http://en.wikipedia.org/wiki/Projection

VITA

Aravind Aluri was born on October 2, 1980 in Eluru, Andhra Pradesh, India. He received a dual degree in B.E.(hons) Computer Science and M.Sc.(hons) Economics from BITS, Pilani, India (1998-2003). He will receive a Master of Science degree in Computer Science from Texas A&M University in December 2005.

Aravind Aluri has acquired professional internship experiences with Intel, Bangalore, India (Jan.-Jun. 2003), and Real Networks, Seattle, WA (Summer 2004). He worked as a Graduate Teaching Assistant for four semesters (Fall'03-Spring'05) and was awarded the IEEE students' choice award for the "Best Computer Science TA" in May 2005. Starting September 2005, he will be working with Microsoft Corporation, Redmond, WA. He can be reached by e-mail at aravind.aluri@gmail.com. His permanent mailing address is: H.No. 23A-11/2, R.R.Peta, Eluru, Andhra Pradesh, India, 534002.

The typist for this thesis was Aravind Aluri.