A Model of the Lateral Geniculate Complex of the Turtle Visual System: Noise Suppression and Target Motion Detection

by

Ronald C. Anderson, B.A.

A Thesis

in

Mathematics

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for the Degree of

Master of Science

Approved

Dr. Bijoy K. Ghosh
Chair

Dr. Magdalena Toda

Dr. Clyde F. Martin

Peggy Gordon Miller
Dean of the Graduate School

December, 2011

ACKNOWLEDGMENTS

Dr. Bijoy K. Ghosh, director of the Center for Bio-Cybernetics and Intelligent Systems (CBCIS) at Texas Tech University, served as my thesis advisor and committee chair. To him I offer much thanks and gratitude. His strong interest in computational neuroscience, positive encouragement, and thoughtful guidance were crucial to the success of this undertaking. Special gratitude is also owed to Mervyn Ekanayake. He offered an uncountable number of suggestions regarding every phase of this work - showing an unwavering dedication to the topic of modeling vision neural systems.

As a long-time collaborator of the CBCIS, the late Dr. Phillip Ulinski of the University of Chicago provided much of the physiological data and biological grounding for the neural models described herein. My eternal thanks go to him for his devotion to the cause of neuroscience advancement and years of service to the scientific community.

Dr. Magdalena Toda and Dr. Clyde F. Martin served as members of my thesis committee, and I thank them for their many suggestions, comments, and services. Specifically, Dr. Toda provided many corrections and recommendations for the substance of this document; her efforts have added much to its exposition. Financial support for this work was provided by two years of funding through the Building Bridges GK-12 program at Texas Tech. To Dr. Dominick Casadonte, principal investigator for this grant, I extend the warmest gratitude. The GK-12 experience provided not only the funding, but the networking experiences required for my inherently interdisciplinary work.

Finally, I offer immeasurable gratitude to my parents, Ronald L. Anderson and Elizabeth M. Anderson. Their support, understanding, and encouragement have always been my foremost privilege and possession. I wholeheartedly dedicate this work to them.

This research was funded by NSF GK-12 Grant #0742402.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii
ABSTRACT v
LIST OF TABLES vi
LIST OF FIGURES vii
I. INTRODUCTION AND PROBLEM DESCRIPTION 1
1.1 Problem Description 2
1.2 Thesis Outline 3
II. BIOLOGICAL FOUNDATIONS AND PRIOR STUDIES 6
2.1 Foundations in Neuroscience 6
2.2 Mathematical Modeling of Neurons 8
2.3 Neural Simulations with GENESIS 14
2.4 Prior Modeling Efforts and Studies 16
2.4.1 Modeling the Retina of Freshwater Turtles 17
2.4.2 A Freshwater Turtle Visual Cortex Model 21
2.4.3 Investigating the Turtle Lateral Geniculate Complex 22
2.5 Perspectives 26
III. THE LATERAL GENICULATE COMPLEX MODEL 27
3.1 LGC Cell Plate Model and Parameters 29
3.2 LGC Neuropile Model and Parameters 31
3.3 Coding the LGC Cell Plate and Neuropile Cell Models 32
3.4 Building the 2D LGC Regional Model 35
3.4.1 Constructing the Cell Plate Layer 36
3.4.2 Constructing the Neuropile Layer 39
3.5 Deriving the Fused Visual System Model 40
3.5.1 Connecting the LGC Model to the Visual Cortex Model . . 40
3.5.2 Connecting the LGC Model to the Retina Model 44
3.5.3 Fused Visual System Model 45
IV. CALIBRATING THE FUSED VISUAL SYSTEM PARAMETERS . . 49
4.1 Geometric Patch Sizes and Radii of Influence 49
4.2 Synaptic Connection Strengths 52

4.3 On Noise and Synaptic Strength Calibration 53
V. SIMULATION RESULTS 56
5.1 Retina Noise Inhibition by LGC Sub-Model 56
5.2 Results of Angle Input Data Trials 60
5.3 Angle Separability Tests 74
VI. DISCUSSION AND CLOSING REMARKS 79
BIBLIOGRAPHY 80
APPENDIX 82

ABSTRACT

The interaction among neurons in the nervous system is a complicated biological and computational phenomenon. Efforts to understand the associations and behaviors of these cells alone and in networks have harnessed the tools from a myriad of academic disciplines. Herein, we explore an application of mathematics and computer systems to study the visual system of freshwater turtles. This thesis focuses on the development of a computerized mathematical model of the turtle lateral geniculate complex (LGC). The LGC is a region of the brain situated as an intermediary neuron layer between the retina and visual cortex and is an integral part of the visual system.

Using physiological data regarding the turtle lateral geniculate complex, a GEneral NEural SImulation System model LGC is constructed that successfully combines an inhibitory effect on noise from a model retina with excitatory relaying of retina input signals to a model visual cortex. This LGC model is then connected to extant models of the turtle visual cortex and retina to simulate input signal transfer from the retina into the cortex.

The combined visual system model is used to investigate retinal noise rejection as well as discernment of input signal trajectory with respect to the visual streak on the retina model. Such analyses are compared to prior works with the retina and visual cortex models without an LGC component. Results indicate inhibitory activity in the LGC cellular model assist in detection of input signal properties while simultaneously reducing the influence of white noise generated by the retina neurons.

LIST OF TABLES

3.1 Cell Plate Compartment Physical Dimensions 30
3.2 Cell Plate Area Computations 30
3.3 Cell Plate Membrane Parameters 31
3.4 Noised FVSM White Noise Variances 46
3.5 Properties of the FVSM and Important Notes 47
3.6 Cell Numbers by Type in the FVSM 47
4.1 FVSM Radii of Influence and Patch Size Values 52
4.2 Synaptic Connection Strengths for Noised FVSM 53

LIST OF FIGURES

1.1 Research project phases 3
2.1 Representation of a Generalized Neuron, credit: [5] 7
2.2 Chemical Synaptic Junction between Pre and Post-Synaptic Neurons, credit: [22] 9
2.3 Neuron Ionic Channels and Ion Pump Representation, [20] 10
2.4 Neuron Action Potential, credit: [5] 11
2.5 Circuit Diagram of HHM Neuron, credit: [2] 13
2.6 Compartmentalizing a Pyramidal Neuron, credit: [14] 16
2.7 Layers of Retinal Cells [2] 18
2.8 Distribution of Retinal Ganglions in Freshwater Turtle [5] 19
2.9 Retina Input Movie Representations. a) Discs incident on retina b) Angle measurements based on direction [12] 20
2.10 Retina Response to 0° Angle Input Movie 21
2.11 Distribution of Visual Cortex Model Neurons in Visual Cortex Model [4] 23
2.12 Visual Cortex Neuron Compartment Models [21] 24
2.13 Sample Cortex Wave Time Series 25
3.1 Three-tier hierarchy of the LGC Model 28
3.2 LGC Cell Plate Compartmental Model 29
3.3 LGC Neuropile Compartmental Model 32
3.4 Cell Plate Model Directory Structure 34
3.5 Neuropile Model Directory Structure 34
3.6 Block Diagram of the 2D LGC Model 35
3.7 Densities per 50×50 micron sector of the Turtle LGC 36
3.8 Distribution of Model LGC Cell Plate Neurons in the Model Space . 37
3.9 Densities per 50×50 micron sector of the Turtle LGC 39
3.10 Distribution of Neuropile Model Cells across the LGC Space 41
3.11 Forming Synaptic Connections in the LGC Model 42
3.12 LGC Model Illustrating Circular Patch 42
3.13 Mapping the LGC Model to the Linear LGC in the Visual Cortex . . 43
3.14 Mapping the LGC Model to the Retina Model 44

3.15 Fusion of LGC and Retina Models 45
3.16 Block Diagram of the Noised Fused Visual System Model 48
4.1 Two-node directed network example 50
5.1 Retina Response to 0° Input Movie and Gaussian Noise 57
5.2 Demonstration of noise inhibition in the FVSM via the LGC Sub-Model 59
5.3 Noised FVSM Visual Cortex output for input movie 0° 61
5.4 Noised FVSM Visual Cortex output for input movie 30° 62
5.5 Noised FVSM Visual Cortex output for input movie 60° 63
5.6 Noised FVSM Visual Cortex output for input movie 90° 64
5.7 Noised FVSM Visual Cortex output for input movie 120° 65
5.8 Noised FVSM Visual Cortex output for input movie 150° 66
5.9 Noised FVSM Visual Cortex output for input movie 180° 67
5.10 Noised FVSM Visual Cortex output for input movie 210° 68
5.11 Noised FVSM Visual Cortex output for input movie 240° 69
5.12 Noised FVSM Visual Cortex output for input movie 270° 70
5.13 Noised FVSM Visual Cortex output for input movie 300° 71
5.14 Noised FVSM Visual Cortex output for input movie 330° 72
5.15 Comparison between a) the Noised FVSM cortical response to 0° input and b) response of the Retina/VC in [12] to 0° input. 73
5.16 Comparison between a) the Noiseless FVSM cortical response to 0° input and b) response of the Retina/VC model in [12] to 0° input. Notice the cortex noise in d). 74
5.17 β components from PCA for the cardinal angle inputs. 76
5.18 a) β components of angles 0°, 30°, 60°, 90°. b) Beta components of angles 90°, 120°, 150°, 180° 77
5.19 a) β components of angles 180°, 210°, 240°, 270°. b) Beta components of angles 270°, 300^0, 330°, 0° 78

CHAPTER I
INTRODUCTION AND PROBLEM DESCRIPTION

As a field, neuroscience is a perfect model for the interdisciplinary research approach. Interactions among the constituent processing cells of bio-neural systems, neurons, are a complex phenomenon requiring the tools and insights of many subjects to clarify. In general, wherever complicated systems are found and a desire exists to decode them, mathematical models are brought to bear. The utility of mathematics lies in its ability to confront multifaceted systems by breaking them into their constituent parts and allowing investigation of these in turn.

The use of mathematical models to describe neural systems resides in the realm of computational neuroscience. But this branch of neuroscience includes much more than just mathematical descriptions of neurons on paper. It also takes tools from computer science and engineering to adequately represent the neuron as an equation (or many equations) that can be programmed on and solved by a computer [2], [14]. The result is a powerful tool for exploring not only the behavior of individual neurons, but also neurons in networks.

Development of such a neuronal network model underlies the present work. We seek possible avenues for future biological investigation into the nature of the freshwater turtle visual system using tools available in the computational neuroscience realm, including computer programming, mathematical analysis, and biological details. While our work is largely computational, the biological underpinnings of our efforts suggest findings may have merit in the biological networks comprising the turtle visual system, provided we always inform our models with known properties of the system under study.

Modeling neurons and their "social" network behavior has repercussions in a myriad of areas. A mathematical model that accurately captures behaviors of a brain region can be used, as in this thesis, to investigate possible functionality of the region in completing a processing task. In our case, the processing task is vision. In other instances, neuronal network models can be used to explore the topology of brain networks, how individual neuron parameters influence the global network behavior, and what certain changes to the network do to the effectiveness of the overall system.

Such models could revolutionize our understanding of certain neurological diseases and disorders, as well as suggest avenues of treatment.

1.1 Problem Description

Herein, the ultimate goal was to construct a model of the freshwater turtle visual system, complete with a retina, lateral geniculate complex, and visual cortex. Two of these sub-components, the retina and the visual cortex, were already in hand from prior studies conducted at the Center for Bio-Cybernetics and Intelligent Systems [5], [11], [12], [13]. The missing component was the lateral geniculate complex (LGC). Phase I of our study involved constructing and calibrating a mathematical model of the LGC for integration with the extant models [2]. The main issue in this phase was finding functional parameters describing two neuron types in the LGC, the cell plate and neuropile.

Phase II consisted of integrating the LGC model with the retina and visual cortex models and tuning the synaptic connection strengths, cellular influence radii, and interface methods among the models. Ultimately, the target was to create waves in the visual cortex model by stimulating the retina model with simulated inputs of a fixed velocity. Twelve sets of input data were tested, including the angles 0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, and 330 degrees. A similar approach is discussed in [5] and [9]. In each case, the retina relayed excitatory signals to the LGC model which in turn provided input to the visual cortex.

One key point in Phase II was to include Gaussian noise in the retina model. A noised model is more reminiscent of a true physical system; however, attempts to include noise in the retina with a direct excitatory connection to the visual cortex were troubled by the retina's hyperactivity to input noise [12]. The visual cortex, in essence, could not discriminate noise from relevant input. To address this issue, the newly constructed LGC model includes inhibition to counteract noise generated by retina cells.

In Phase III, the waves generated by the cortex in Phase II were analyzed via component analysis to determine if angle data were separable. That is, given a wave generated in the visual cortex, is it possible to determine which input angle was presented to the retina? The interest here is to see how the LGC model influences

the results. Of major note is that, unlike in prior studies, this study includes noise in the model retina, made possible through inhibition within the Lateral Geniculate Complex model. In Chapter II we look at a few of the prior studies in this area, particularly those in [4], [5], [12], and [13].

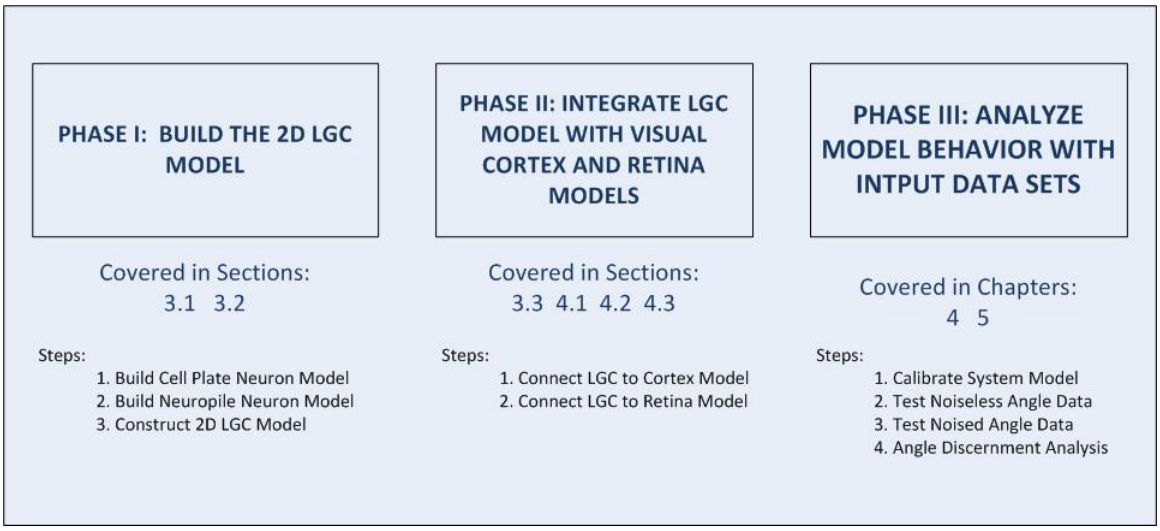


Figure 1.1: Research project phases

In all cases, our interests lie in shedding light on possible functions of the lateral geniculate complex in the greater visual neural system of turtles. Our results seem to suggest utility of the LGC as, among other things, a noise-inhibiting feature of the visual system [2]. To our knowledge, this is the first and only freshwater turtle visual system model constructed with a visual cortex, retina, and lateral geniculate complex. Therefore, this thesis contributes new analyses of neural network parameters and cellular compartmental models. These realizations should be useful in future studies involving not only the visual system of turtles, but also other brain networks in various organisms.

1.2 Thesis Outline

This exposition is divided into six chapters and one appendix. The six chapters follow the general format of defining the problems we wish to address, investigating the currently available resources and some past relevant studies, building the model,

and finally using the model to address the motivating questions.

In Chapter I we look at some general information about computational neuroscience and how this thesis fits into the framework of the field. We also look at some rationales for the utilization of mathematical models in the study of neural systems such as those found in the turtle visual system. Next, the research project is broken into three phases and each phase is described based on the study goals.

Chapter II presents some biological considerations regarding the turtle visual system that are important for understanding the extant visual cortex and retina models as well as the construction of the LGC model. Next, we look at the retina and visual cortex models that will serve as components in our FVSM. Along with the introduction to these models, we also consider some previous studies conducted with them. These results will be important in assessing the FVSM and its behavior. Concluding this chapter is a consideration of the need for an LGC component in the fused visual system model. This serves as a motivation for the three phase project approach first illustrated in Chapter I.

The model LGC is addressed in Chapter III. To illustrate the model formulation, we use a bottom-up approach. The first section will describe the cellular models used in the LGC region model: the cell plate model and the neuropile model. Section two describes how the 2D regional LGC model is constructed from physiological data and the two cell type models. Model dynamics and behavior are then discussed before the fused visual system model is described. It is this fusion of the visual system components that serves as the anchor for all our investigations. In essence, Chapter III describes Phase I of the study in its entirety as well as the results from Phase II after the FVSM is formulated. Chapter IV extends on Chapter III by providing a description of the major calibration steps for the FVSM.

Phase III of the project is addressed in Chapter V. Here we describe the analyses conducted with the FVSM, how the results compare with previous studies without the LGC (when such comparisons are possible and relevant), and look at some new functionality provided by the LGC in the area of noise suppression. We conduct the following studies with the FVSM: discernment of input angle via component analysis on the cortical waves generated by said input and investigation of how the FVSM responds to a noised retina input with specific angle measures. In all cases, unless

otherwise specified, the results come from the FVSM with all network and cellular parameters unchanged. This thesis closes with a short chapter observing the context of results and possible avenues of future work. In the appendix, the GENESIS code and MATLAB scripts referenced in the main chapters are collected for illustration purposes.

CHAPTER II
BIOLOGICAL FOUNDATIONS AND PRIOR STUDIES

2.1 Foundations in Neuroscience

Biological notions can be modeled using mathematics, but the first step in this process is understanding the concepts that must be emulated. For the described visual system model, the primary focus is on the constituent neurons of the lateral geniculate complex, retina, and visual cortex. Any venture into a neuroscience textbook will convince the reader of the broad variety of these building blocks in the central nervous system [15]. Over one hundred billion of these special-purpose cells make up the human brain alone. Neurons serve an important role in the bodies of complex organisms; they are cells that issue electro-chemical signals in the nervous system to control organs and process sensory input. Neurons accomplish these functions by interacting with each other in specific ways, sending communication signals among each other and conditionally relaying messages of their own based on received messages. In essence, while a neuron is an individual cell, its "social" behavior underlies the true functions of organismal nervous systems. Networks of neurons interact to solve complex tasks.

Figure 2.1 shows the structure of a generalized neuron. Here we see several components of the cell labeled. Since the neuron is a living cell, it has a nucleus and cell body, referred to as the soma. Among other activities, the soma carries out cellular metabolic processes. Most neurons exhibit dendritic extensions, shown in the figure as bushy appendages on the soma. These dendrites serve as receptors for electro-chemical signals from other neurons. On the right side of the neuron in Figure 2.1 are axon terminals. The axon and corresponding terminals are the message-sending structures of the neuron. Note that some neurons do not have a long axon as shown in this figure. For those with long axons, a myelin coating comprised of fatty cells is often present to insulate the axon fiber and speed message transmission. In some cases, these axons can be over a meter long [15].

It is important to emphasize that, while the generalizations we present here regarding neurons are applicable in our modeling venture, there are neuron types that do not exhibit the same behaviors. Our computer simulations all use neurons that

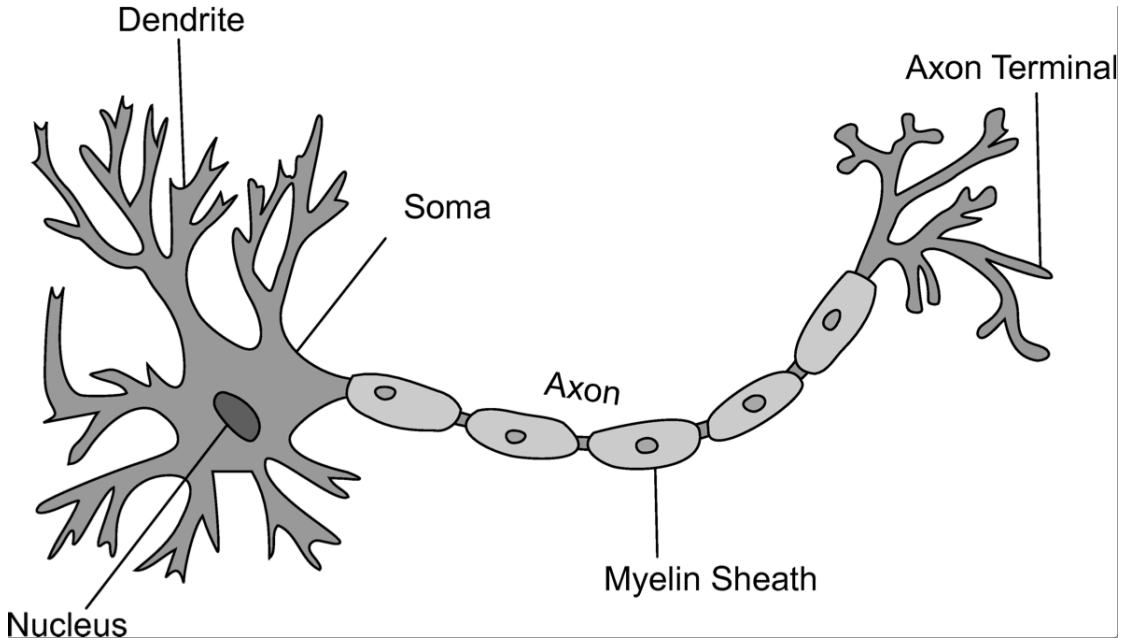


Figure 2.1: Representation of a Generalized Neuron, credit: [5]

can be characterized by understanding Figure 2.1 however, so we will not investigate these details further in this exposition.

Networks of neurons form through synaptic connections. Messages can be relayed by a neuron through its axon terminal(s) to the dendrites of nearby neurons (of the same or different physiological descriptions). This process of sending a message is often referred to as a neuron "firing". In this process, the neuron sending the message is termed the pre-synaptic neuron and the receiving cell is called the post-synaptic neuron. These messages are generated through a process of depolarization of the neuron cellular membranes, causing electrical signals to propagate along the neuron membrane toward the axon terminals. Upon reaching the axon terminal the signal releases chemicals known as neurotransmitters. These chemicals cross a small gap, called the synaptic junction, between an axon terminal of the pre-synaptic neuron and a set of chemical receptors on a dendrite of the post-synaptic cell. The synaptic junction is illustrated in Figure 2.2. When detected by the post-synaptic cell, these transmitters have an effect on its behavior.

While there are many types of neurotransmitters, their effects can be loosely

characterized with two categories: excitatory and inhibitory. During inhibition the neurotransmitter chemicals serve to hyper-polarize the receiving cell. This temporarily decreases the target cell's proneness to sending messages of its own. In the excitatory instance, the transmitters encourage the receiving neuron to depolarize. Usually, neurons collect signals from a large number of neighboring cells. In some instances, thousands of synaptic junctions exist per cell. Neurons can thus accumulate signals from many sources and weigh the contributions of each, conditionally firing based on these contributions [5] [14].

The interactions of neurons using excitatory and inhibitory signals prove very important in this thesis. Using these two concepts, it is possible to control the passing of messages within a neural network. These steps are, from the biological perspective, of paramount importance for neurons as computational entities. Decisions in neural networks are made via the number, type, and location of each synaptic connection in the network, as well as the relative importance, or "weight" ascribed to each synapse. These weights, referred to herein as "synaptic weights" or "synaptic strengths" are vital to network training. We will investigate these concepts later when discussing the Fused Visual System Model. For now, it suffices to remember that synaptic connections are not of one description. Some are more important than others for particular tasks and will, therefore, often have different synaptic weights in accordance with their functions.

The preceding discussion was a major simplification of an incredibly complex biological design. However, these simplifications are vital in producing mathematical models of neurons. By their very nature, mathematical models seek to remove complexities from problems that contribute little to the system behavior under study. We have in the previous paragraphs all the details needed to discuss basic neuron models and the GEneral NEural SImulation System (GENESIS).

2.2 Mathematical Modeling of Neurons

The mathematical model underlying neuron behavior is the Hodgkin-Huxley equation. This differential equation model can be used to simulate the depolarization of a neuron's membrane, where a cascade of ionic charges crosses the neuron's lipid bilayer through ionic channels. In Figure 2.3 we see the membrane surface of a

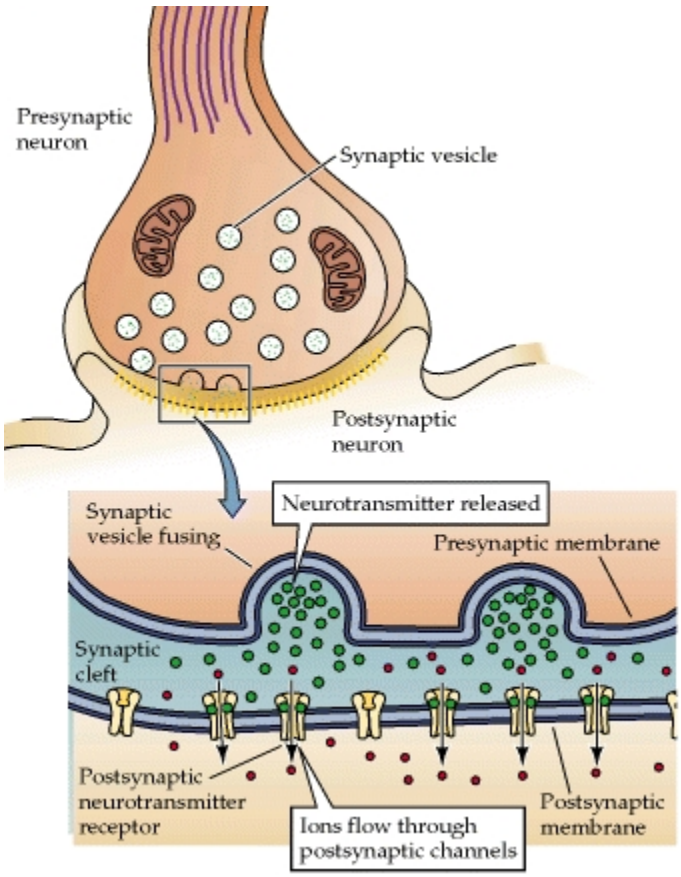


Figure 2.2: Chemical Synaptic Junction between Pre and Post-Synaptic Neurons, credit: [22]

neuron with the corresponding ionic channels labeled. In the resting state, neurons separate sodium and potassium ions across their cellular membranes, creating a potential difference across the between the cellular exterior and interior. Typically, this differential is about $-65mV$, but can vary from neuron type to neuron type. It is this separation of charged ions across the neuron membrane that creates the potential for message passing among cells. When a neuron receives sufficient excitation to fire, the potential difference across the membrane is used to generate an action potential, sending an electro-chemical signal across the length of the neuron. Figure 2.4 shows a graph of membrane potential versus time for a given neuron during action potential. As can be seen in the figure, the potential across the membrane looks like a spike

during neuron firing, giving rise to the phrase "spiking neuron". As will be seen in the modeling steps later in this paper, neurons can spike many times, causing what is called a spike train. The frequency of such spikes is indicative of how excited the neuron is. In general, if a neuron is stimulated sufficiently to fire, if that stimulus remains constant, the cell will continue spiking at a particular frequency. As the intensity of stimulation rises, the frequency of spiking usually goes up as well until it reaches the theoretical limit of spiking rate for the neuron, defined by the cell's refractory period.

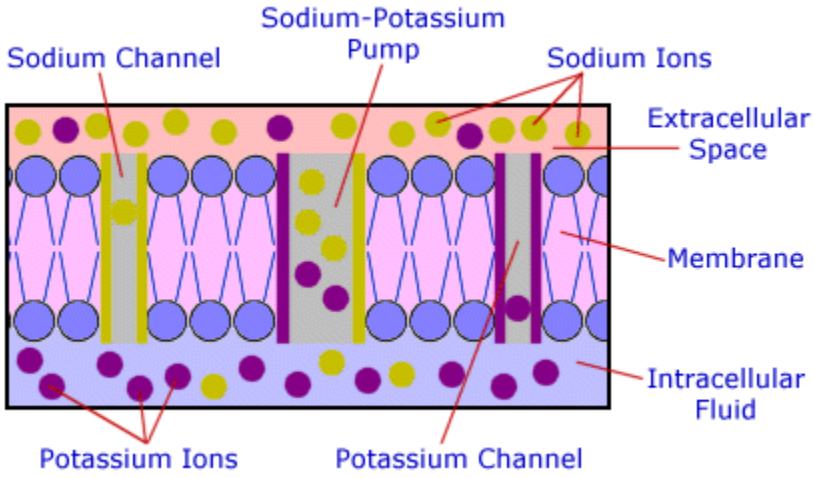


Figure 2.3: Neuron Ionic Channels and Ion Pump Representation, [20]

In order to understand the chemical origins of the Hodgkin-Huxley neuron model, hereafter referred to as HHM, it is constructive to use figure 2.4 and discuss what is happening at each stage represented on the plot. The traditional HHM assumes two ions of interest, Sodium (Na^{+1}) and Potassium (K^{+1}). As shown in the formulae, both these ions have a positive one charge. From basic chemistry and physics, it makes sense that the ions exhibit repulsive electrical forces on one another, seeing all are of positive charge. Envisioning the ions and a neuron submerged in fluid, the neuron's membrane forms a semi-permeable enclosure, defining the cellular exterior and interior.

Each neuron has ionic channels through its membrane that are selectively perme-

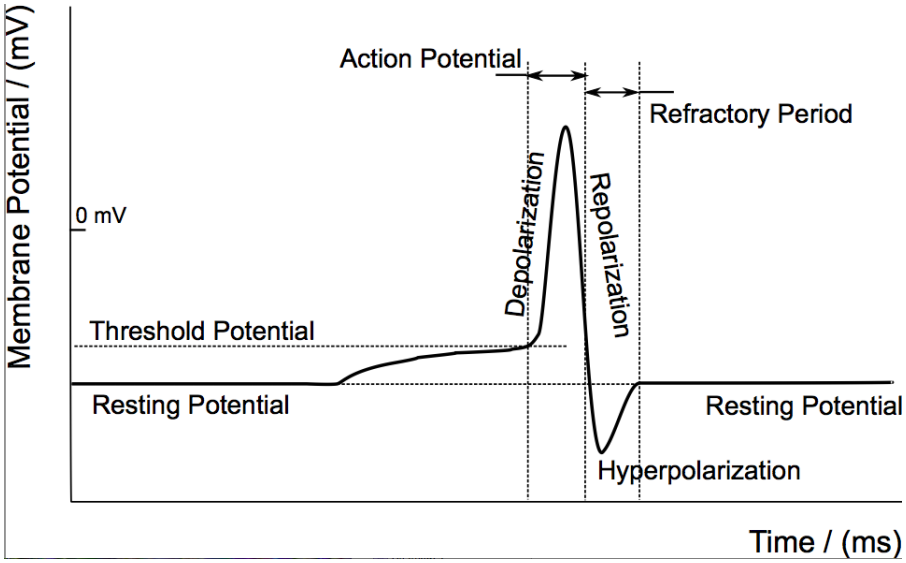


Figure 2.4: Neuron Action Potential, credit: [5]

able by certain ions [15]. Thus, sodium ions (and only sodium ions) can cross the Na ionic channels into and out of the neuron whenever the channels are open. Likewise, potassium ions (and only potassium ions) can cross the K channels. These channels can be opened or closed by the cell, allowing control of the movement of ions across the membrane. The neuron also has at its disposal ionic pumps which can be used to force ions of particular types to cross the cellular membrane. Using these pumps in tandem with the ionic channels, a neuron can change the density (concentration) of ions of specific types inside of the cellular space. Therein lies the neuron's ability to form a potential difference across its membrane. The neuron actively pumps sodium and potassium ions across its membrane, causing the outside of the cell to become ionically more positive with respect to the cellular interior. This pumping continues until the neuron reaches a defined resting potential P_{rest}. In this case, the ionic channels are kept closed to ensure the ions being transferred do not re-cross the membrane.

At the resting potential P_{rest}, the neuron maintains a differential of about $P_{rest} = -65$ millivolts with respect to the cellular exterior. Due to influences of neurotransmitters, the cell can be excited or inhibited. When inhibition occurs, the cell becomes temporarily hyperpolarized, which changes the membrane potential P_m such

that $P_m < P_{rest}$. During excitation, the cellular membrane potential P_m rises so that $P_{rest} < P_m$. Should P_m rise such that it reaches the threshold potential P_{thresh}, the neuron undergoes depolarization. During depolarization, the sodium channels on the cell membrane open, allowing a flood of sodium ions to enter the less-positively charged cellular interior. As the sodium ions enter, the cellular membrane potential rises. Each channel in the neuron's membrane is voltage dependent, meaning the channels open and close at defined potential levels across the membrane. When the depolarization brings the cellular membrane potential to about 30 millivolt above zero potential, the sodium channels close, stopping further sodium ions from crossing into the cell via the ionic channels.

Now, repolarization occurs. The potassium channels open, allowing K^{+1} ions to exit the neuron. They do so rapidly due to the high density of positive sodium ions and potassium ions in the cell. As these potassium ions escape through the K channels into the external cellular space, the neuron's potential drops rapidly from its apex, falling back into negative potential ranges. This stimulates the potassium ionic channels to shut, blocking potassium ions from exiting the cell through the channels. Now, all sodium and potassium ionic gates are shut. The neuron has, at this point, a cellular potential slightly below resting potential, termed hyperpolarization. It now activates its ionic pumps to force potassium ions into and sodium ions out of the cell until the resting potential is again reached. This rise back to resting potential is referred to as the neuron's refractory period. During this "resetting" period, the cell cannot fire. Since this period is well-defined for each neuron, it places a limit on the frequency of "spikes" the neuron can generate. Once the refractory period is over, the cell is again at resting potential and susceptible to excitation.

This simplified action potential description allows the representation of neuron spikes as an electric circuit. Figure 2.5 shows such a circuit diagram. Since the cellular membrane serves to separate charges between the cellular interior and exterior, it behaves much like a capacitor in an electric circuit. This is C_m in the diagram. The cellular membrane has a resistance to ionic flow, denoted by R_m. E_K and E_{Na} are the Nerst Potentials generated by separation of the ions in question across the cellular membrane. I_{in} denotes an inject current, usually from an external source such as another neuron. This current is the electrical representation of synaptic signals being

passed from one neuron to another. Each of the variable resistors g_K and g_{Na} represent the cumulative behavior of ionic channels on the neuron for the given ion. When the K channel is closed, g_K ideally has resistance of ∞ to current flow, creating an open circuit branch, disconnecting E_K from the circuit. A similar description applies to g_{Na} and g_x. Here, g_x is a generalized channel, where x is replaced with the particular ion the channel is compatible with, for instance calcium. E_x is the Nerst Potential for that particular ion. For our purposes, we assume $g_x = \infty$; we will only consider K and Na ions in our models.

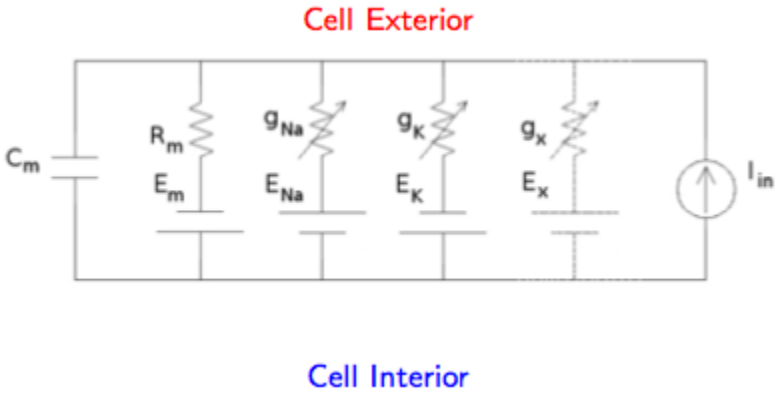


Figure 2.5: Circuit Diagram of HHM Neuron, credit: [2]

At resting potential, both g_K and g_{Na} are of high resistance (low conductance). This results in no net current flow through the cell due to these ions. Thus, the capacitor is charged by current flow from E_m through g_m until it reaches equilibrium, at which point the capacitor behaves as an open circuit to current flow. This leaves a potential difference across the capacitor (which in this case indicates a potential difference across the membrane.) The inject current I_{in} can be used to further charge the capacitor, or discharge it, depending on flow direction. During charging due to an inject current, the potential difference rises. When it reaches a set threshold, the variable resistor g_{Na} decreases resistance to current flow, opening the ionic channels and allowing current flow from E_{Na} to contribute to the capacitor's charge, causing it to drop. This is the depolarization phase. When the capacitor's charge reaches a set level, g_{Na} returns to infinite resistance while g_K drops from infinite resistance to low resistance, allowing potassium ions to contribute to the capacitor's charge. Since

E_K is of opposite polarity to E_{Na}, this causes the capacitor's charge to begin rising, defining the repolarization phase. Once the repolarization phase reaches its climax, g_K return to infinite resistance, shutting down the flow of potassium ions. Thus, the cellular membrane capacitor begins to return to the resting state defined by R_m and E_m.

We can translate this diagram and verbal description into mathematical representation using differential equations and the law of conservation of energy. Via current analysis and the law of energy conservation:

$I_{in}(t) = I_C(t) + I_m(t) + \sum I_x(t) = I_C(t) + I_{Na^+}(t) + I_{K^+}(t) + I_m(t)$
Then, $-I_C(t) = I_{Na^+}(t) + I_{K^+}(t) + I_m(t) - I_{in}(t)$
Recall: $I(t) = C\frac{dV(t)}{dt}$ Thus,

$$-C\frac{dV(t)}{dt} = I_{Na^+}(t) + I_{K^+}(t) + I_m(t) - I_{in}(t)$$

This equation underlies the Hodgkin-Huxley Neuron model [14]. The parameters that control the ionic channels are not discussed further here, however detailed discussions can be found in [5] and [14]. With the properly chosen parameters for ionic channels and cellular membrane properties, the Hodgkin-Huxley equation above will have an action potential like that seen in Figure 2.4.

2.3 Neural Simulations with GENESIS

The Hodgkin-Huxley model underlies the GEneral NEural SImulation System (GENESIS) and several other software tools used to simulate the behavior of neurons. In this study, we rely only on GENESIS for our neural simulations since our previous works used this scripting language. Basically, GENESIS and similar neural simulation software programs provide a method to program neurons and networks of neurons into a computer. This program is usually a text script (see the examples in the appendix) that provides the Hodgkin-Huxley parameters governing the model neuron behaviors. The script also describes how many neurons are in the model, what their physical sizes and locations are, and how they synaptically connect to one another. Based on these details in the script files, the simulation software couples the differential

equation models of synaptically connected neurons in the network and incrementally solves the equations. This allows the simulator to trace action potentials from neuron to neuron and determine how they influence each other.

Equations common in neuronal modeling are usually quite complex and must be solved by software programs using numerical analysis techniques. This is particularly true with large models involving many neurons or highly detailed models of individual neurons. GENESIS is equipped with such routines, reducing the burden on the computational neuroscience researcher as far as mathematical coding is concerned. Of course, since discrete computations are involved when using a computer, care must be taken to ensure major numerical errors do not crop up in simulations. Many a researcher has been lead astray with "interesting" simulation results, only to later learn those results were representative of computational hiccups in the simulation [14].

Despite the challenges in developing numerical models using software, the benefits are great. A well-designed simulation allows access to neuronal activity (even at the sub-cellular level) that can't be readily observed in a biological subject. The simulations also allow the modification of otherwise inaccessible parameters governing the network. A major question now arises. How do we actually model a neuron in software? The answer ultimately has to do with finding the correct numerical parameters for the HHM equations. This defines the neuron's action potential and thus, its polarization properties. Arriving at these values is often a trial and error process, attempting to match what is known biologically about the subject neuron(s). Usually, we look at morphological differences among the cells and classify them according to these properties, using the same model for each grouping. More detailed studies are of course possible, but individualizing thousands of neurons in a network is usually not feasible. As a result, such approaches are more common with detailed models of individual neurons, like those described in chapters one through three in [14].

While there are several vantage points for neuron modeling, the usual approach is through a process called "compartmentalization". In this procedure, information about the biological structure of the subject neuron type is investigated and reduced to a series of spheres and cylinders. Each of these represents one compartment of the model neuron. The behavior of each compartment is described mathematically

using the HHM and its requisite parameters. With this information in hand, the compartment is coupled with other compartments comprising the neuron so as to preserve the important structural aspects of the cell. Figure 2.6, taken from "The Book of GENESIS," illustrates one possible compartmentalization of a Pyramidal neuron, which is one of many neuron classes found in the nervous system. One will note that the compartment model looks quite unlike the real neuron. This is due to simplification of the "bushy" nature of the dendritic extensions and the axon branches. There are mathematical descriptions and rules that are usually applied during this compartmentalization process to arrive at a simplified model. We will see the compartmentalization process in action in Chapter III.

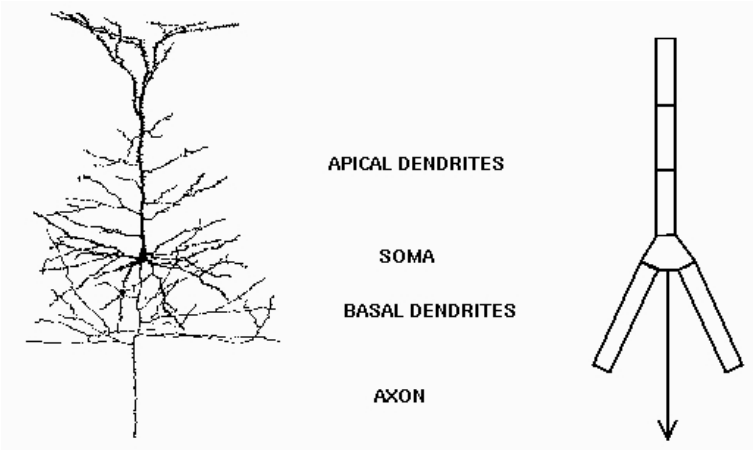


Figure 2.6: Compartmentalizing a Pyramidal Neuron, credit: [14]

2.4 Prior Modeling Efforts and Studies

Like all vertebrates, freshwater turtles have a central nervous system comprised of neurons. It thus makes sense to look at the turtle brain and its associated structures as entities that can be described in mathematical terms. Over the last decade, the Center for Bio-Cybernetics and Intelligent Systems (CBCIS) has worked in the area of computational neuroscience to better understand the nature of the turtle visual system. This system has proven to be a complex network to model, but has yielded interesting and surprising results regarding the ability of neural systems to encode visual information.

In this section, we look at several studies that have been conducted in recent years to model activities of the turtle visual system. Since these studies are building blocks for this thesis, but do not define novel work for the resulting Fused Visual System Model, we only look at the aspects of these efforts that are vital to understanding the present work.

2.4.1 Modeling the Retina of Freshwater Turtles

While argument can be made that the retina is not part of the brain, it is a vital part of the visual system of turtles and other organisms. Biologically, the retina is a collection of cells at the back of the eye. These cells include various versions of neurons, called ganglions. Cells in the retina that are sensitive to electromagnetic radiation of defined wavelengths (for instance, rods and cones in the human retina) send signals to the ganglion neurons. There, the signals are processed and passed by the ganglions over the optic nerve into the brain. Depending on the animal, these ganglions can vary in their morphology and detection abilities. The availability of various sensory cells in an organism's retina determines characteristics of its sight abilities [5] [15].

Figure 2.7 illustrates the cellular structure at the back of an eye. Notice the layers of cells in the retina. Light (the green arrow) is incident on the retina. Photosensitive cells called rods and cones detect this light and send neural signals through the Horizontal, Bipolar, Amacnine layers to the Retinal Ganglions. The signals arriving at the ganglions excite them, and these excitatory neural action potentials are sent down the optic nerve to vision processing centers in the brain.

While the image above describes the human retina, the principles for the turtle visual system are largely the same. Also, since the retinal ganglion cells are neurons, they are governed by the HHM and can thus be modeled using the methods described earlier. Dr. Mervyn Parakrame Ekanayake of the CBCIS Laboratory developed a model turtle retina, complete with five classes of ganglion cells based on the detection abilities of the turtle eye. For purposes of this retina model, only the ganglions are simulated [5]. The model is comprised of "A_OFF", "A_ON", "B1", "B2", and "B3" ganglions. The physiological meaning of each of these classes can be found in [5]. The B cells comprise the majority of the retina model and are divided into three classes

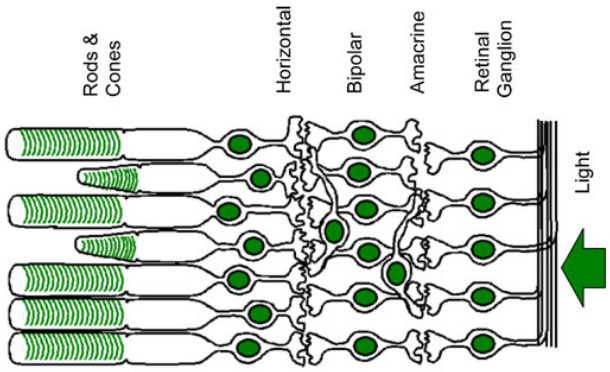


Figure 2.7: Layers of Retinal Cells [2]

based directional preference for retinal input. The "A_OFF" and "A_ON" cells are direction insensitive.

As we will see later, all of our models are based off of biological data regarding cellular densities in their respective regions. The retina model is no exception. In fact, cells of various types are not evenly distributed across the retina surface. In turtles, cells are concentrated in the "visual streak". This equates to the turtle having its most-sensitive vision in this elongated region on the retina. Color coding represents densities of ganglion cells in this region of the retina, as seen in Figure 2.8. Blues represent few ganglions per square unit, and the densities increase up to highest values, represented as dark red on the image. The white circle is called the "retinal patch" and indicates the region where we use when modeling the turtle retina. Due to the large number of neurons in the turtle retina (about 370,000) and limited computational resources for handling such a complex model, we choose a small patch region of the retina about 50 microns in radius.

To build the retina model, the cell types defined and coded in GENESIS by Dr. Ekanayake are randomly spread across the 50 micron radius disc so as to meet the density requirements for those ganglion types in the chosen patch of the retina. We choose a patch such that the area of highest cellular density is included in the disc. Once the cellular placement is complete, the retina model is exposed to injected current values into its HHM compartments, triggering neural signals from the cells that can be relayed to models of brain regions responsible for vision processing.

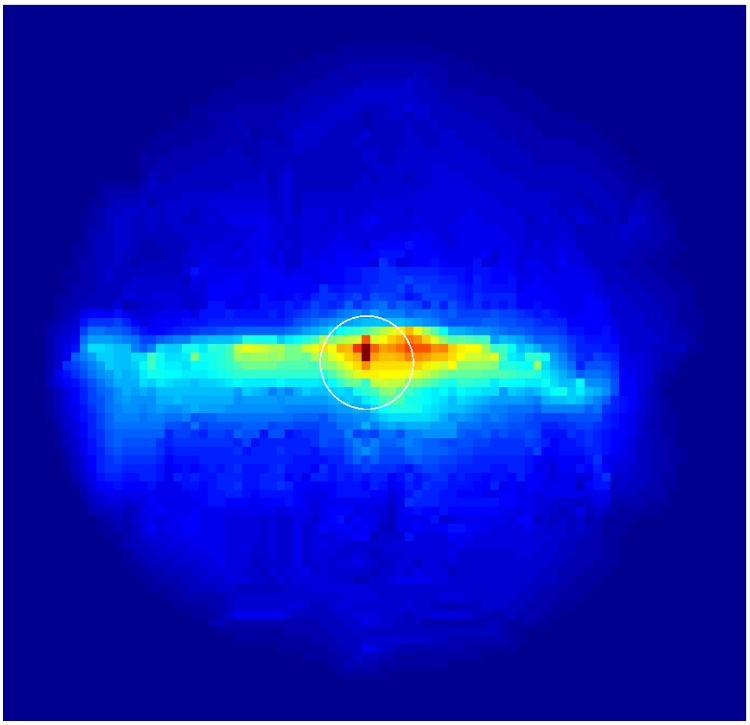
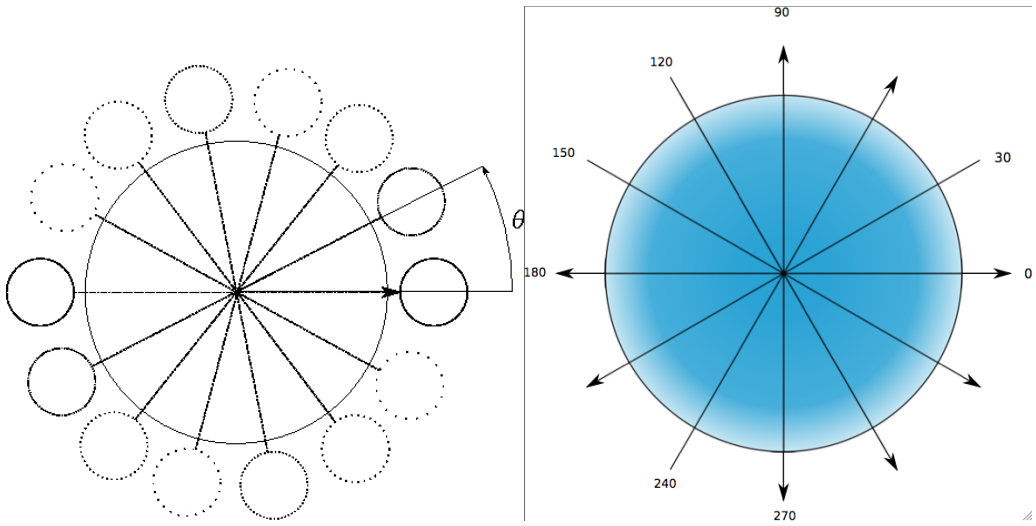


Figure 2.8: Distribution of Retinal Ganglions in Freshwater Turtle [5]

While input to this model could be any numerical value, we generally use "movies" simulating a moving white disc. This disc progresses across the retina in a defined direction and at a specified speed. The direction can be defined using an angle θ with the horizontal and a defined direction. See Figure 2.9a for a graphical representation. The large disc represents the retinal patch from Figure 2.8 and the smaller discs represent instances of the light disc stimulating the retina during the movie. Note that each movie has only a single disc. The movement is relatively fast, requiring on the order of $700ms$ to traverse the retina. One can envision the movie as a disc that moves over the retina, stimulating all ganglion cells under the disc at the given instant. These stimulations are what trigger the neural signals.

Figure 2.10 shows the response of retinal cells to sample movie input of $0°$. The colors represent relative neuron activity levels due to stimulation. Notice that, as time advances, a disc of activity can be seen moving across the retina model from left to right along the horizontal axis. This input is characteristic of all movie inputs to the

model retina in [5], [10], [12], and this thesis. As we will see later however, Gaussian random noise of a specific variance and zero mean can be added to the retina input signals to induce "noisy" conditions. This topic will be explored in Chapters IV and V.



(a) (b)

Figure 2.9: Retina Input Movie Representations. a) Discs incident on retina b) Angle measurements based on direction [12]

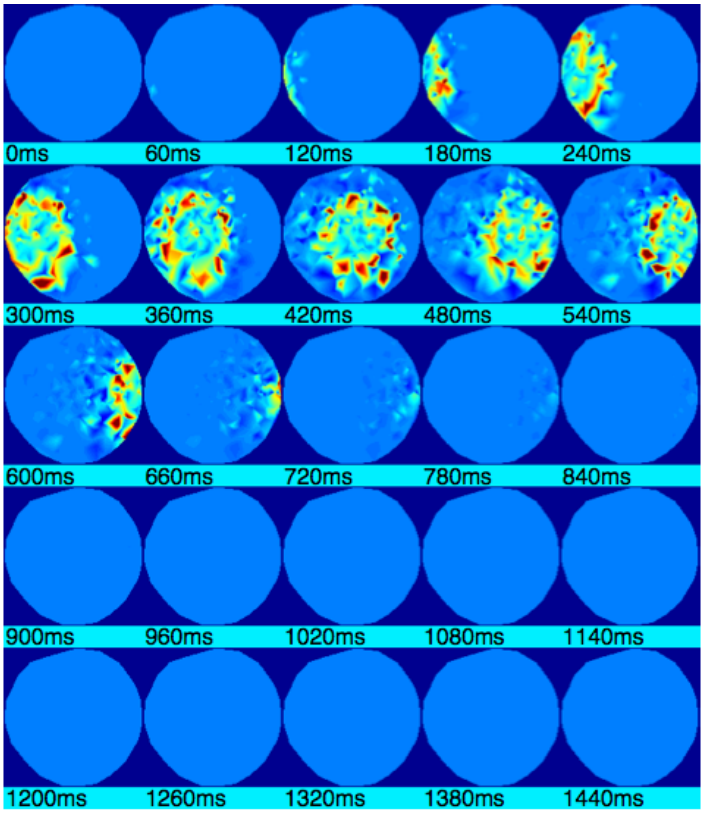


Figure 2.10: Retina Response to 0° Angle Input Movie

2.4.2 A Freshwater Turtle Visual Cortex Model

In [4] and [13], and [21], a freshwater turtle visual cortex model was constructed and studied. This model was specifically configured to yield cortical waves matching those found in the turtle visual cortex. This is an example of designing a neural model using known biological details, including cellular morphology and densities, and then calibrating it based on biological observations of network behavior. This model has been the focus of many theses and publications, including [4], [8], [9], [12], and [13].

The visual cortex model we use is comprised of six cellular types. Five of these cells reside in the cortex, while the sixth type is representative of a prior attempt to model the lateral geniculate complex. This is discussed in the next section. The five cortex cell types are the Horizontal, Lateral, Medial, Subpial, and Stellate neurons. These cells are distributed throughout the cortex model according to turtle visual

cortex properties. In Figure 2.11, we see the five cell types displayed in the visual cortex model space. In this diagram, the directional axes are labeled. L=Lateral, M=Medial, R=Rostral, C=Caudal. Figure 2.12 shows the compartmental models of each of the five visual cortex model cells. Details regarding their derivation and behaviors can be found in the aforementioned literature. Of note, however, is that these cells exhibit specific inhibitory and/or excitatory influences on one another to generate realistic cortical waves based on input.

Generation of cortical waves in the visual cortex is vital to the work in this thesis. While we will see many examples of cortical waves in the following chapters, it is appropriate to show a sample output wave now to illustrate this cortical behavior and place in context for upcoming discussions. This example is shown in Figure 2.13. The color scheme in this figure represents the spiking activity of neurons in the cortical model. Blue shading indicates low excitation of neurons in that region, while red shading indicates high spiking activity in that region. Yellow shading represents medium excitation levels of nearby neurons. In the figure, a "wave" of activity is seen propagating through the visual cortex model as time advances.

Like the retina model, the cortex cells are exposed to input signals. While the retina experiences input from movie files, the visual cortex model receives spike trains from the lateral geniculate complex model. These spike trains, if sufficiently energetic, will trigger a cortical wave from the neurons in the cortex. It is from this wave that we extract information about the stimulus responsible for the wave. Such information could include its speed and direction. In essence, the cortex waves appear to encode information about the stimulus that generated them.

In Chapter III we develop the FVSM, which includes the visual cortex model discussed above. There, we will see more information about the model's structure and encoding abilities.

2.4.3 Investigating the Turtle Lateral Geniculate Complex

During construction of the visual cortex model in the last sub-section, a 201 cell model of the LGC was used to feed input into the cortex. The location of each of the 201 model LGC cells was not defined biologically. Instead, each cell was one element in an array. The LGC cell axons in the model project into the visual cortex along

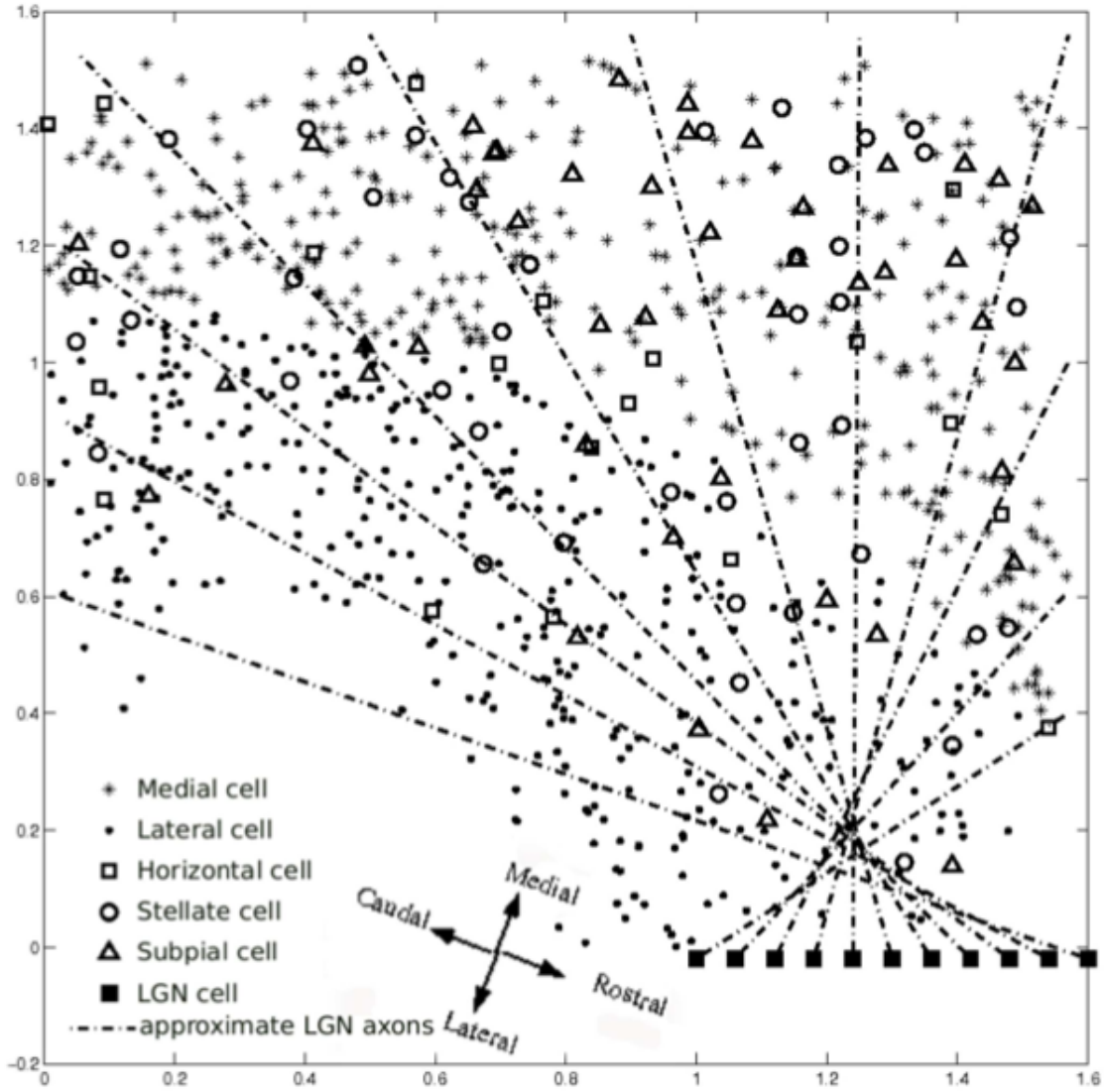


Figure 2.11: Distribution of Visual Cortex Model Neurons in Visual Cortex Model [4]

the lines shown in Figure 2.11. These projections, termed the axonal projections of the LGC [6], feed signals from the simple one-compartment cells of the LGC linear model into the visual cortex.

Indeed, when the visual cortex was constructed, the LGC was not a modeling target. This was largely due to lack of physiological information about the LGC

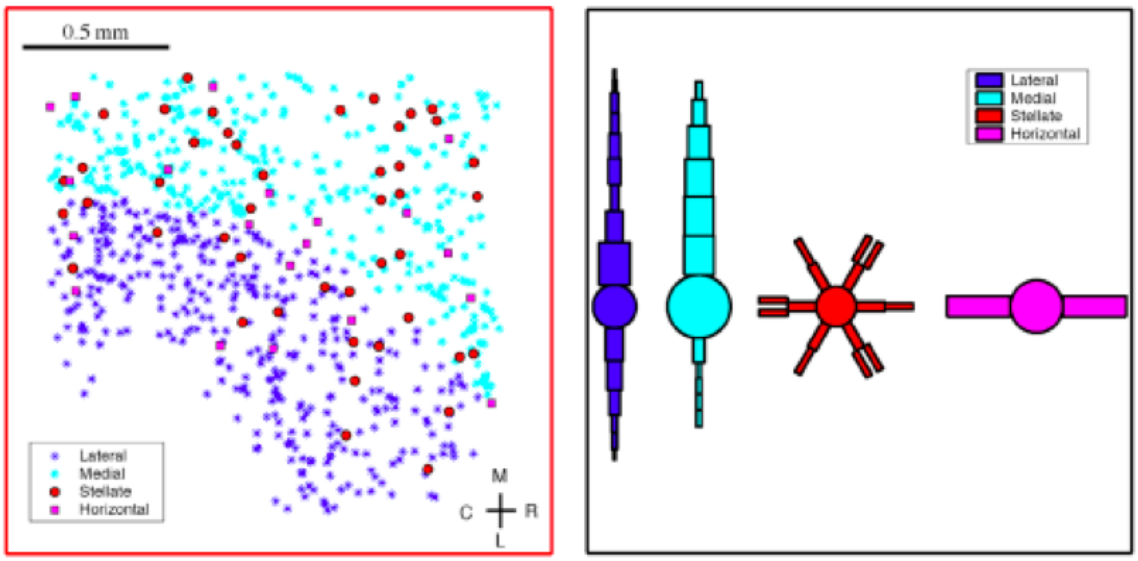


Figure 2.12: Visual Cortex Neuron Compartment Models [21]

cells and their distributions. However, since the LGC serves as the input route for the cortex, a basic LGC was needed for model interface purposes. This linear LGC model is still considered, even as of this writing, as a major part of the cortex model. In fact, it still serves as the input medium for our modified LGC model.

In [11], Jenner Joseph discusses projections of images onto a 2D LGC model. Note that this exposition uses the term Lateral Geniculate Complex (LGC) while the cited study uses the term Lateral Geniculate Nucleus (LGN). They are referring to the same structure in the turtle brain. The work involved the use of images, or "scenes" mapped onto a realistic distribution of model neurons comparable with the densities of cells in the turtle lateral geniculate complex. These realistic densities were also used to construct the current LGC model as part of the present project.

The key differences between Jenner's study and the one described herein have mostly to do with model structure and input type. At the time the study cited in [11] was conducted, a model turtle retina as described in subsection 2.4.1 was not available. This excluded the possibility of building the fused visual system model presented in this thesis. Also, the inputs for Dr. Jenner's LGC model were real photographic images. The work thus focused on how such images could be encoded

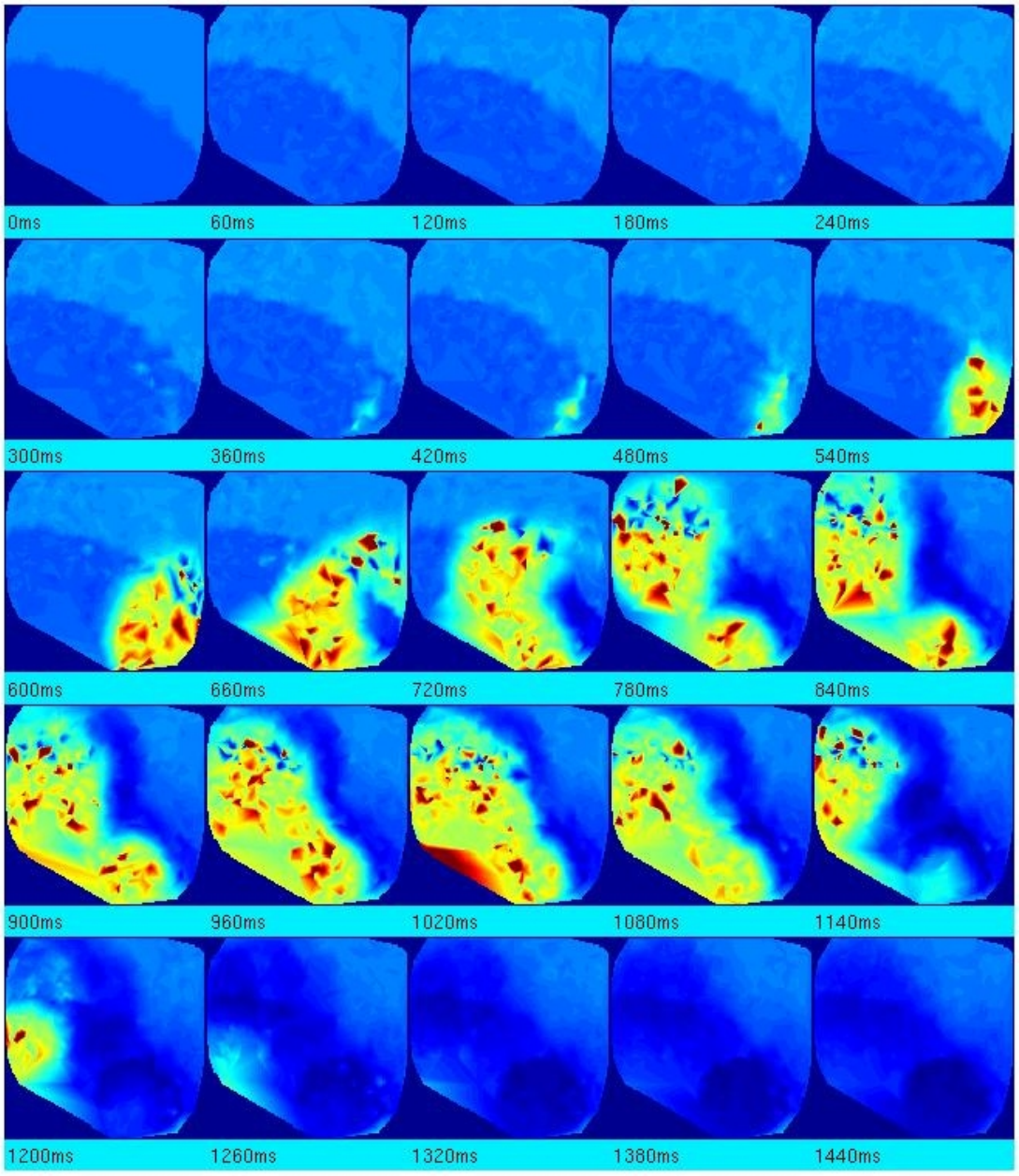


Figure 2.13: Sample Cortex Wave Time Series

using the LGC and served as input to the visual cortex model in Section 2.4.2. In the present study, we seek an LGC model with two cell classes that focuses on inhibiting noise from the attached retina model while sending important retinal signals to the

visual cortex.

2.5 Perspectives

With the observations from this chapter, an argument has been made for mathematical modeling of the neurons of the freshwater turtle visual system. The presence of both a visual cortex and retina model comprised of such neurons encourages, as in [12], attempts to design large-scale models of the visual system, including the retina and visual cortex. However, a deeper desire exists to extend the combined Retina/VC model investigated in [12] to include a model Lateral Geniculate Complex and explore its possible functions in vision. Therein lies the primary objectives of this research and exposition.

CHAPTER III
THE LATERAL GENICULATE COMPLEX MODEL

In this chapter we look at the development of the lateral geniculate complex (LGC) model via a cellular to network, bottom-up approach. First, GENESIS models for each cell type in the LGC are produced. These models then serve as templates to produce a 2D LGC regional model incorporating over 1000 such cells. Physiological data from the literature [1] [3] [6] [7] and previous models [4] [5] [11] [12] [13] are used to guide cell and network topological parameter configuration. The end goal is a model that incorporates the two primary neuron classes found in the freshwater turtle LGC that remains true to the approximate cellular distributions of that region.

Our main premise was that the neuropile cells found in the LGC may serve as inhibitors for noise in the retina. As a result, these model cells were given inhibitory synaptic connections with the more numerous cell plate LGC neurons. The finished 2D LGC model serves as an intermediary structure between existing models of the turtle retina and visual cortex, described in Chapter II. The combined model is referred to as the Fused Visual System Model (FVSM) and is described at the end of this chapter. The models discussed below are implemented using the General Neural Simulation System (GENESIS). A brief description of GENESIS is provided in Chapter II. All code referenced herein is included verbatim in the appendix. The LGC model is combined with the extant retina and visual cortex models in a single GENESIS script called *FVSM.g*.

Before beginning the construction process a point needs to be made regarding what is meant by "model" in the following sections. There are many different models that this thesis addresses. These are best described in a three-tier hierarchy. At the top of the hierarchy is the Fused Visual System Model, referred to as the system level. This model is made up of three sub-models, the visual cortex model, retina model, and the new LGC model. These three models exist at the regional level (so named since both the LGC and visual cortex are brain regions). Each of these regional models is divided into sub-models addressing their individual cellular constituents. These are the cellular level models. While all three models have multiple cellular components, we only address the two cellular models in the LGC in this document;

the cellular models making up the other two regional models are not original work to this project.

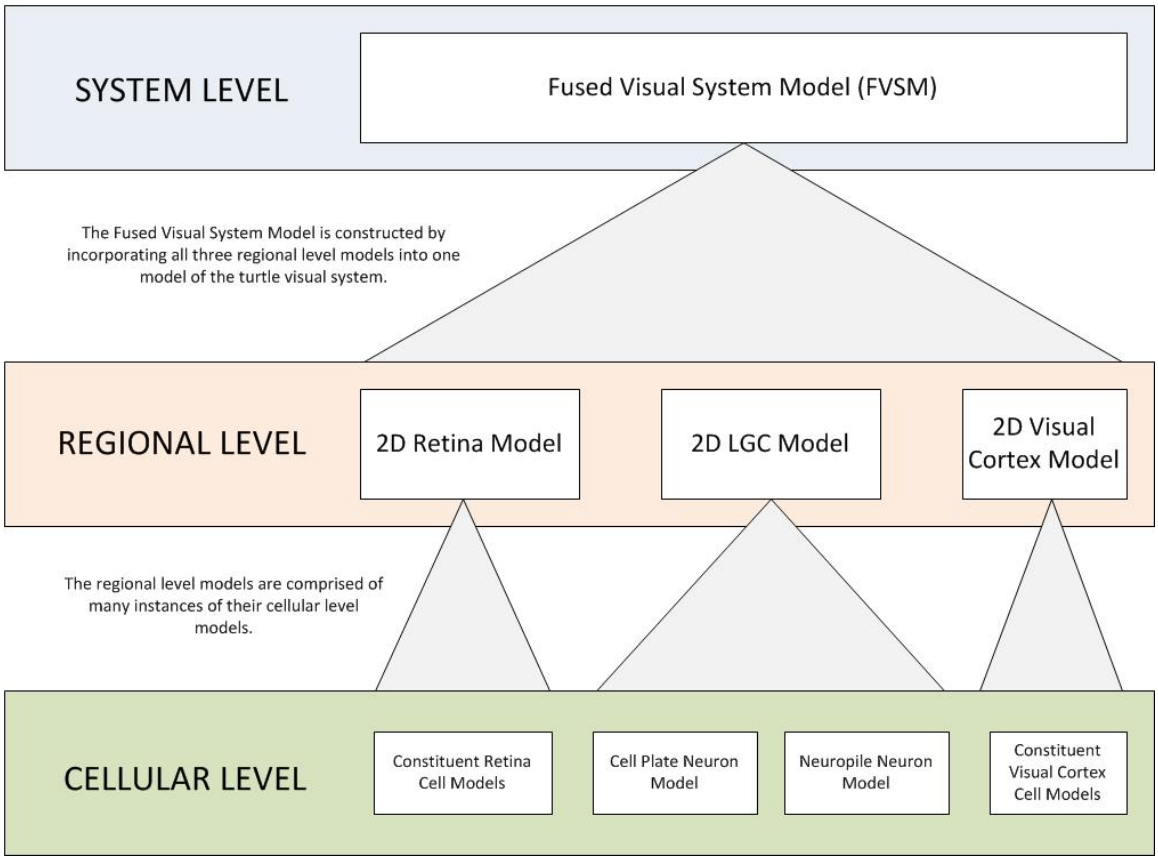


Figure 3.1: Three-tier hierarchy of the LGC Model

The freshwater turtle lateral geniculate complex is comprised of two major neuron classes: the cell plates and neuropiles. Both of these classes are incorporated into the model LGC and are constructed using compartmentalization. In this procedure, the actual neurons structure is simplified into a series of boxes (or compartments) that retain some physical details from the original cell being modeled. Through the compartmentalization and coding process a set of GENESIS functions are obtained that, when called by a GENESIS script, produce an instance of the requested LGC neuron. This neuron model instance is complete with all membrane and dimension parameters required for inclusion in the multi-cell 2D LGC model discussed in Section

3.5. Also defined are the Hodgkin-Huxley mathematical parameters governing the modeled aspects of each member neuron.

3.1 LGC Cell Plate Model and Parameters

Figure 3.2 illustrates the compartmental model of the LGC cell plate neuron class. This model structure was suggested by Dr. Philip S. Ulinski of the University of Chicago in correspondence [3]. In Figure 3.2, the circular compartment represents the soma of the cell plate neuron. Each rectangular compartment represents dendritic extensions from the cell. The line extending from the soma compartment represents the cell plate's axon and is modeled in GENESIS as a synaptic connection with delay. It has no compartment structure of its own.

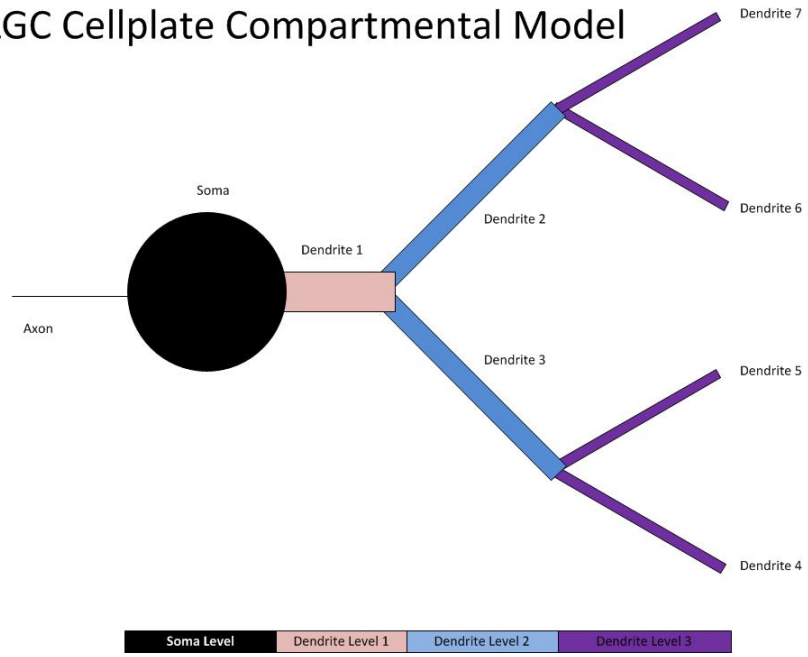


Figure 3.2: LGC Cell Plate Compartmental Model

Each compartment is three-dimensional and has physical dimensions suggested by Dr. Ulinski to be in keeping with the relative size of cell plate neurons found in the turtle LGC. The soma is modeled by a spherical compartment while all dendrites are constructed using cylindrical compartments. The dimensions of each compartment

shown in Figure 3.2 are given in Table 3.1 below.

Of keen interest in constructing these models is the surface area of each compartment. The parameters in the Hodgkin-Huxley model equations depend, to an extent, on the surface area of the cell since it is through this membrane that ions move. These ions underlie the neurons spiking behavior, as described in Chapter II. Table 3.2 lists the area computations used for the model geniculate cell plate cells for each of the compartments. Here, *dend_dia* is the compartment diameter and *dend_len* is its length. The approach used for these computations is identical to those found in the visual cortex model.

Table 3.1: Cell Plate Compartment Physical Dimensions

Compartment	Compartment Shape	Dimensions (microns)
Soma	Spherical Compartment	Radius: 15
Dendrite 1	Cylindrical Compartment	Length: 20
		Width: 5
Dendrites 2 & 3	Cylindrical Compartment	Length: 100
		Width: 2
Dendrite 4 to 7	Cylindrical Compartment	Length: 50
		Width: 2

Table 3.2: Cell Plate Area Computations

Compartment	Area Computation
Soma	$\pi(\frac{soma\_dia^2 - dend1\_dia^2}{4})$
Dendrite 1	$\pi(dend1\_dia \times dend1\_leng)$
Dendrite 2	$\pi(dend2\_dia \times dend2\_len)$
Dendrite 3	$\pi(dend3\_dia \times dend3\_len)$
Dendrite 4	$\pi(dend4\_dia \times dend4\_len + \frac{dend4\_dia^2}{4})$
Dendrite 5	$\pi(dend5\_dia \times dend5\_len + \frac{dend5\_dia^2}{4})$
Dendrite 6	$\pi(dend6\_dia \times dend6\_len + \frac{dend6\_dia^2}{4})$
Dendrite 7	$\pi(dend7\_dia \times dend7\_len + \frac{dend7\_dia^2}{4})$

Considering Figure 3.2, Table 3.1, and Table 3.2, we now have all the basic structural parameters for the cell plate neuron model. Remaining to be specified are the electrical properties of the neurons membrane. These parameters are vital to

proper generation of spike signals from the neuron model and are determined largely by trial and error. The approach involved copying the parameters originally used in the Linear LGC model [4] and adjusting them until the desired action potential plot was obtained. Table 3.3 outlines the membrane parameters chosen for the cell plate model. Units of measurement in Table 3.3 are: Ω (Ohms), S (Siemens), V (Volts), F (Farads), and m (meters).

Table 3.3: Cell Plate Membrane Parameters

Parameter	Numerical Value	Description
RM	$2.45\ \Omega \times m^2$	Membrane resistance
CM	$0.02429\ \frac{F}{m^2}$	Membrane capacitance
RA	$1.389\ \Omega \times m$	Axial resistance
EREST	$-57 \times 10^{-3} V$	Resting membrane potential
Eleak	$-57 \times 10^{-3} V$	Membrane leakage potential
Gleak	$0.027 \frac{S}{m^2}$	Max leakage potential
ENa	$0.045 V$	Sodium reverse potential
EK	$-0.9 V$	Potassium reverse potential
GNa	$6000 \frac{S}{m^2}$	Sodium max conductance
GK	$300 \frac{S}{m^2}$	Potassium max conductance

3.2 LGC Neuropile Model and Parameters

No detailed physiological data were available for the neuropile cells found in the turtle LGC. As a result, the LGC model includes a generalized single-compartment soma neuron for this cell class. For this model, the soma diameter is assumed the same as the cell plate class discussed in the previous section, 15 microns. The soma is spherical. Figure 3.3 illustrates the structure of the neuropile model neuron.

The single compartment soma model of neuropile cells is undeniably simple. However, given the reality of scant physiological information and a desire to investigate global network behavior, such a basic model was deemed appropriate for our purposes. In [8] and [12] , a similar approach was taken when modeling the lateral geniculate complex as a linear collection of 201 cells. Each cell was of the single compartment soma variety. This model proved quite effective in numerous studies [3] [4] [8] [9] and seemed to encourage such generalization in the face of limited data.

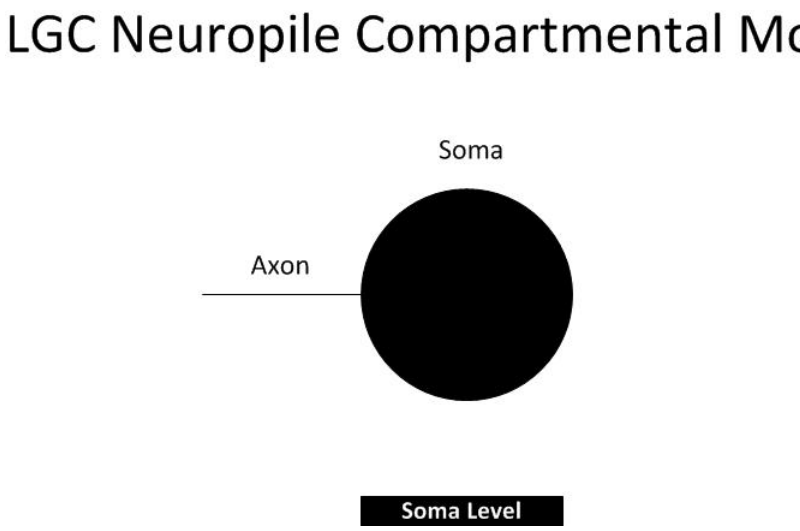


Figure 3.3: LGC Neuropile Compartmental Model

3.3 Coding the LGC Cell Plate and Neuropile Cell Models

The parameters specified in the previous two sections effectively define the mathematical properties of the LGC neuron model classes. These parameters must now be integrated with the General Neural Simulation System (GENESIS) to produce a mathematical model of each cell class whenever needed. This is accomplished by writing a series of functions that, when called by a GENESIS script, create the compartments specified in the discrete model diagrams, link them, and apply the parameters to each compartment of the model. To accomplish the tasks in the next sections of this chapter, the functions should produce an instance of either the LGC cell plate or LGC neuropile model, depending on which is requested.

In the appendix, these functions, along with the parameters specified in Sections 3.1 and 3.2, are found in the file *lgc_neuron.g*. The first part of this file contains the parameter declarations for membrane resistance, capacitance, dimensions, etc. Following the declarations, a series of functions are listed that build the cellular model. These functions call one another in a specific order. As specified with an "include" command in the code, *lgc_neuron.g* relies on the presence of another GENESIS file, *phil_channel.g*. This file includes parameters and functions for defining the ionic channels used in the cell's soma to generate spikes. As this code was developed and

used with previous models and is unchanged in the LGC model instance, no further mention of it is given here. There are also other GENESIS files used by the model codes that are not included in the appendix. The reason for excluding these codes is that they remain unchanged from previous publications.

The parent function responsible for initiating the model construction process is called *make_lgc_neuron*. It requires that two parameters be passed by the caller: a location for the cell instance and the "soma_only" flag. The former is a directory in the GENESIS environment that will house this particular neuron model instance. The latter is a numerical parameter which, when set to zero, tells the function to build dendrite compartments. If set to another numerical value, the function will produce a cell model with only a soma. In essence, a flag of '0' defines the cell plate model. Other numerical values define the neuropile model.

As we will see later, it is necessary to link many cells together via synaptic connections. While *make_lgc_neuron* links cellular compartments together as part of the cellular level model construction process, it does not synaptically join neurons. That is done at the regional and system levels, not the cellular level. However, in order to make synaptic connections among neurons, it is necessary to understand where each model's compartments are located and how to refer to them. When *make_lgc_neuron* runs, it produces subdirectories in the "location" argument passed to it and populates those directories with parameter values and objects (such as synaptic channels) necessary for cell function.

Now we consider by example what happens when the *make_lgc_neuron* command is executed. Consider the following command:

```
make_lgc_neuron  {{cell_location}/{cell_name}} {0}
```

The directory structure produced by this call is illustrated in Figure 3.4.

Now, consider the following snippet and note the difference between it and the previous code. This call produces a neuropile cell since the flag is not set to zero.

```
make_lgc_neuron   {{cell_location}/{cell_name}}   {1}
```

The directory structure produced by this call is like that in Figure 3.4, but does not contain the dendrite subdirectories since the neuropile model has no dendrites. It

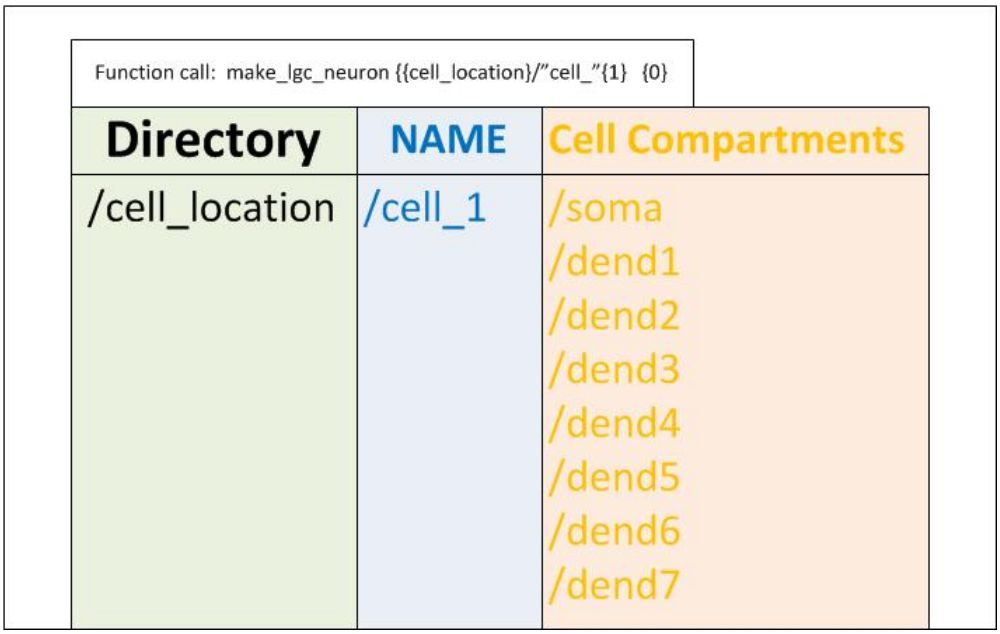


Figure 3.4: Cell Plate Model Directory Structure

does, however, contain the "soma" directory. Figure 3.5 shows the directory structure resultant from this call.

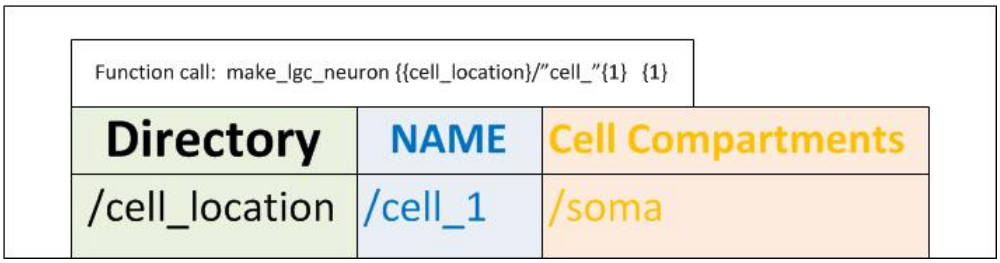


Figure 3.5: Neuropile Model Directory Structure

Hence, whenever we wish to refer to a compartment in a created neuron model, we navigate to the directory containing the cell and then change directory into the name of the desired compartment. In GENESIS, compartments, cells, and most objects are treated as directories. With the described functions and definitions in *lgc_neuron.g*, we can create model cell plate and neuropile LGC neurons. We now turn to the regional level to build a large collection of such cells reminiscent of the

turtle lateral geniculate complex.

3.4 Building the 2D LGC Regional Model

The two neuron types of interest are not distributed evenly throughout the turtle LGC region. Philip S. Ulinski et. al. have determined via dissection studies the approximate distribution and densities of both neuropile and cell plate neurons. The total cell count for both classes is estimated between 10,000 and 20,000 cells, although the exact number of cells will vary from turtle specimen to turtle specimen and species to species [1] [3].

Our LGC model is to provide a cell plate neuron layer for accumulating and relaying signals from a retina model to a visual cortex model. A neuropile layer is also to be included that serves an inhibitory effect on the cells in the cell plate layer. The general block diagram of the LGC model is provided in Figure 3.6. Here, the red arrow represents synaptic connections from the neuropile cell classes to members of the cell plate classes. These inhibitory signals will be described later.

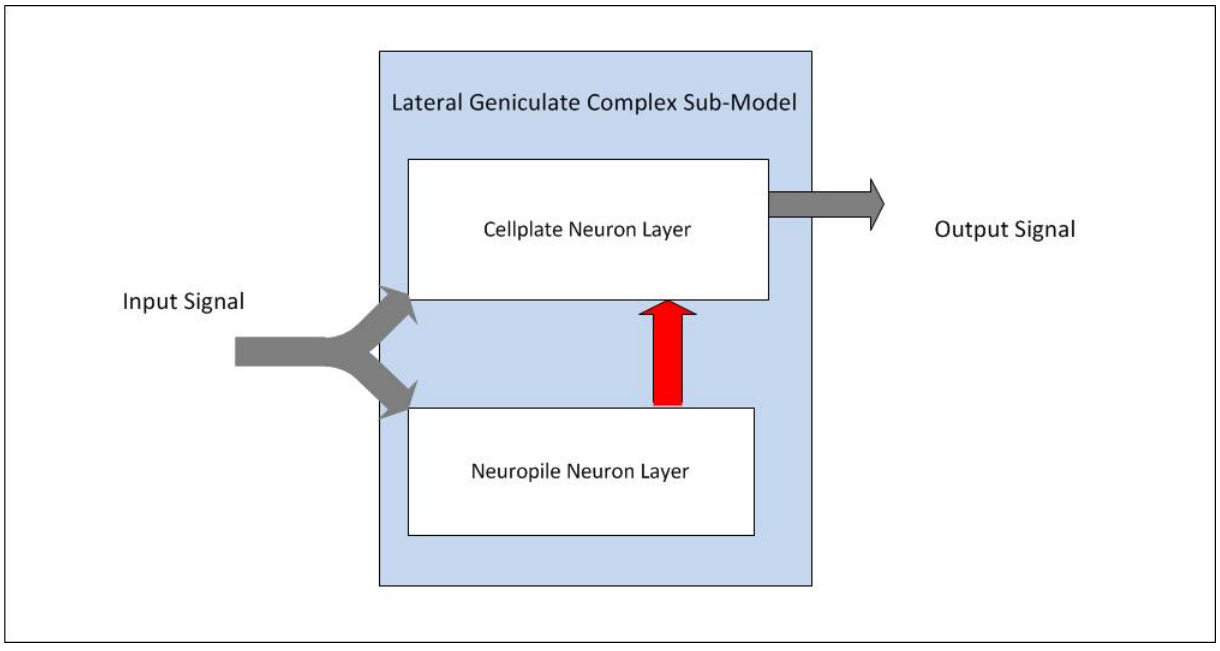


Figure 3.6: Block Diagram of the 2D LGC Model

In correspondence, Dr. Philip Ulinski provided density matrices for the cell plate

and neuropile classes, and these are shown in Figures 3.7 and 3.9. These densities can be used to generate planar coordinates for LGC neurons with approximately the same density per 50 micron by 50 micron sector as the actual LGC. However, due to computational restrictions, we scale the number of cells down to a more reasonable range. The process is documented in the following subsections.

3.4.1 Constructing the Cell Plate Layer

According to [1] [3] and the density table in Figure 3.7, the cell plate neurons of the turtle lateral geniculate complex comprise about 92 percent of its the total neuron population. These cells are distributed throughout the LGC roughly as described in the cell plate densities in Figure 3.7. The cell plate neuron classes, as we will see later in this chapter, form synaptic connections with neurons in the retina and visual cortex models. They do not interact with one another. Effectively, they serve to accumulate and relay activity in the retina to the visual cortex for further processing. As a result, the model cell plate layer is just a collection of unconnected neurons. Figure 3.8 shows a sample distribution of the LGC cell plate neurons in the model space. The number of cells in this figure is higher than the number chosen for our simulations.

0	0	0	0	0	0	0	0	0	0	20	18	11	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	6	61	74	16	0	0	0	0	0	0
0	0	0	8	37	37	75	22	9	35	75	93	39	6	8	0	0	0	0
0	0	0	53	100	107	102	68	52	65	72	84	51	37	13	0	0	0	0
0	4	25	97	165	137	130	103	106	84	73	66	42	54	20	0	0	0	0
0	12	81	142	185	131	138	105	107	97	85	58	46	54	37	8	0	0	0
0	24	68	151	190	138	153	110	132	81	83	56	47	44	34	20	1	0	0
0	61	82	124	154	152	162	118	176	104	74	59	40	39	36	25	1	0	0
0	98	113	81	88	154	164	123	175	125	90	57	69	67	46	34	1	0	0
0	128	134	52	19	130	150	118	161	125	92	58	60	75	73	53	2	0	0
0	121	113	23	0	33	83	83	143	112	74	53	53	65	86	51	6	0	0
0	101	93	4	0	0	46	57	84	100	61	47	29	65	82	58	13	1	0
0	42	24	0	0	0	8	30	44	41	8	11	8	47	67	54	27	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	30	60	50	35	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	17	45	54	39	5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	5	39	52	52	11	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	20	38	11	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	10	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4	1

Figure 3.7: Densities per 50 × 50 micron sector of the Turtle LGC

Each model cell, when created in GENESIS, has field parameters including spa-

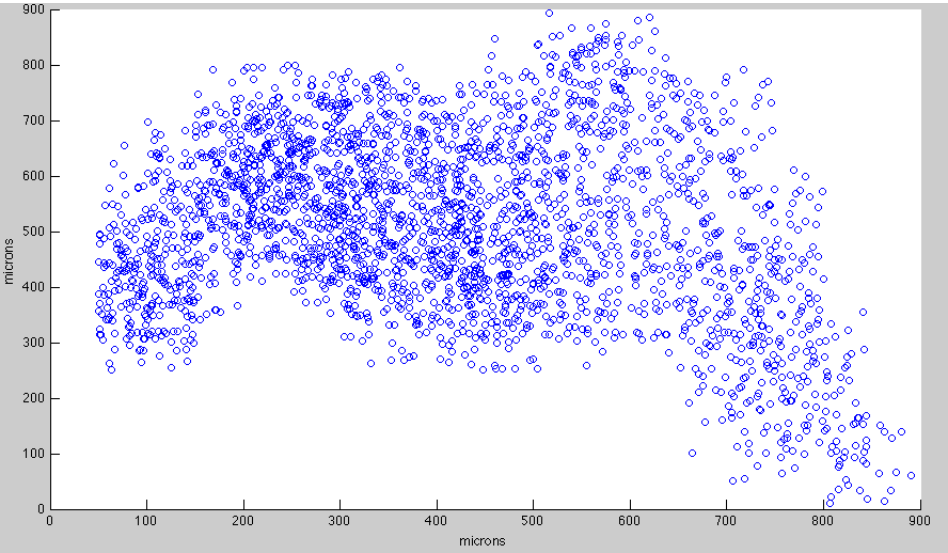


Figure 3.8: Distribution of Model LGC Cell Plate Neurons in the Model Space

tial coordinates X, Y, and Z to place it within a three-dimensional region. It is our goal to set these model parameters with values derived from the density data. Since our model is two-dimensional, we set the Z coordinate of all cells to zero, which is the GENESIS default for coordinate values when they are not specified.

The first step in the process is to obtain cellular coordinates from the distribution data. The number needed is equivalent to the number of cell plate cells in the model, which must also be defined. Once the coordinate list for the cells is obtained, it must be imported using GENESIS so as to assign each cell plate neuron a location on the 2D model surface.

The LGC used in data collection consisted of 1169 cell plate model neurons. This number was obtained by taking the distribution data and dividing each entry by 10 and rounding up to the nearest integer, effectively scaling the model size. Shown in the appendix, the MATLAB code entitled *LGC_distro_creator.m* is responsible for producing random cell coordinate values in each block of the scaled matrix of densities. The code steps through each block of the scaled density matrix and randomly computes X and Y pairs in the given block of 50 microns by 50 microns such that the number of pairs generated equates to the number of cells in the scaled sector. These coordinates are then stored in a data file that will be used as input to the GENESIS

script that builds the 2D LGC Model.

A comment is in order regarding the choice of scaling factor. This number was selected, not in a deliberate attempt to match some biological parameters, but rather to provide enough cells to preserve the distributional variation in the LGC while at the same time keeping the computational complexity of the model down. The model is designed to be scalable, so the number of cells can be easily changed and new coordinate values computed later for targeted studies addressing these parameter choices.

The *lgc_coords.dat* file generated by the MATLAB code consists of three columns and, for the present model, 1169 rows. Each row represents one cell plate neuron in the LGC region. Each column consists of the cells X, Y, and Z location values. In all instances, the Z value is set to zero. Inclusion of the Z column despite its lack of useful information allows the reuse of the coordinate reader script first developed with the visual cortex model. Since the LGC, retina, and visual cortex models are destined for fusion into a single system, sharing of code among models makes the integration and configuration simpler.

Importing the cellular coordinates into the GENESIS system is straightforward. First, the model script reads in the number of LGC cell plate neurons desired (call this number N) and, inside a loop, repeatedly calls the cellular model constructor function *make_lgc_neuron* providing the location of the neuron as "network_lgc/cell_{n}", where {n} is replaced with the loop index running from 1 to N. The "soma_only" flag is set to zero since the desired neuron is of the cell plate variety.

After the cells have been produced, the coordinate values can be applied to the desired compartments. Every compartment can have coordinate locations defined. For our models however, we only set specific coordinate values for the soma compartment. The other compartment coordinates are left at their default values, the origin of the coordinate system. This is not a problem since the compartments are correctly linked by *make_lgc_neuron*. Thus, regardless of the compartments location, it associates with the correct soma. To set the soma coordinate values, a loop successively calls the *coord_reader.g* script to read the coordinates data file line by line. The first line of the data file is assigned as the coordinate values for cell one, line two for cell two, and so on. Through this process, the Cell Plate Neuron layer is produced in the

LGC model.

3.4.2 Constructing the Neuropile Layer

About eight percent of the total neuron population of the turtle LGC is of the neuropile class. Their approximate densities are found in Figure 3.9. The process for creating this layer is almost identical to that discussed in 3.4.1. The differences reside in the density matrix chosen and the model neuron type. Also, all calls to the *make_lgc_neuron* function must contain a non-zero "soma_only" flag in this process to ensure the neuropile model is produced.

0	0	0	0	0	0	0	0	0	0	1	2	4	0	0	0	0	0	0
0	0	0	0	0	0	3	0	0	4	2	3	4	2	2	0	0	0	0
0	0	0	4	3	4	8	4	2	5	4	3	6	11	8	0	0	0	0
0	0	0	7	9	5	3	12	4	10	8	7	2	14	11	0	0	0	0
0	1	3	1	13	10	8	9	4	8	8	5	1	7	8	3	0	0	0
0	2	8	5	18	20	4	8	11	6	8	7	2	6	7	1	0	0	0
0	3	4	4	15	23	10	5	7	4	9	9	4	8	8	10	0	0	0
0	7	14	5	23	26	12	8	11	10	14	8	3	1	5	11	0	0	0
0	9	8	7	21	18	16	7	6	8	9	7	3	9	4	7	0	0	0
0	16	13	9	11	13	12	12	20	6	4	8	7	4	8	11	0	0	0
0	21	12	8	0	2	9	9	8	8	5	7	4	2	3	8	1	0	0
0	11	16	4	0	0	0	9	4	4	6	7	5	7	3	5	10	0	0
0	3	6	0	0	0	0	0	1	2	0	0	0	2	5	6	7	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	7	3	1	5	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	1	7	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	9	8	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4	3	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.9: Densities per 50×50 micron sector of the Turtle LGC

After finding coordinates for each neuropile and assigning those values to the cell population (See Figure: 3.10) as was done in the previous subsection, we proceed with one additional step that is of keen importance if noise is to be used in the forthcoming Fused Visual System Model: providing inhibitory connections from the neuropile layer to the cell plate cells. In the present model, these synaptic connections are determined based on geometric distance and neuron type. Cell plates in our model, as previously mentioned, do not interact with each other synaptically. Their synaptic connections extend to the visual cortex model. Hence, the only synaptic connections from neuron to neuron within the LGC model are from neuropile cells to cell plate neurons.

To establish the synaptic connections in the LGC we step through each neuropile cell in the model, extract its planar coordinates, and compare those with the coordinates of every cell plate instance. If the distance between the currently chosen neuropile and cell plate is below a defined connection radius, a synaptic connection is created between the cells. If the distance is greater than that radius, the cells remain disconnected. A more detailed discussion of this parameter is provided when model calibration is covered later. Figure 3.11 shows pictorially the radius of influence and formation of synaptic connections based on geometric distance in the LGC. The kind of synaptic connection is important here. The connections from neuropile to cell plate neurons are inhibitory, meaning activity in the neuropile cells should reduce the sensitivity of cell plate neurons to input. The second type of connection, the excitatory, is discussed when the FVSM is built in the next section. The purpose of inhibitory and excitatory synapses and how they interact in our model should become apparent at that time.

3.5 Deriving the Fused Visual System Model

This section marks the last level of the three-tier hierarchy illustrated in Figure 3.1. Here, merging of the three regional models is described and the resulting system is calibrated. We will look at two specific instances of the Fused Visual System Model (FVSM) here. The first is the simplest model, which lacks Gaussian noise in the retina. For the purposes of this model, the inhibitory connections within the LGC are non-existent. The second model is the full noise model, which has noise in the model retina to better-approximate a true biological system. We look at both versions since calibration of the noiseless model parameters helps guide configuration of the noised parameters.

3.5.1 Connecting the LGC Model to the Visual Cortex Model

The cell plate neurons of the LGC are synaptically connected to the visual cortex model via the linear LGC model discussed in Chapter II. In this sub-section, a method of linking the LGC to the cortex is illustrated. There are many approaches that could be used to accomplish this end. Here, a linear-division geometric process is applied.

Figure 3.13 shows the mapping from LGC to Linear LGC used in the current

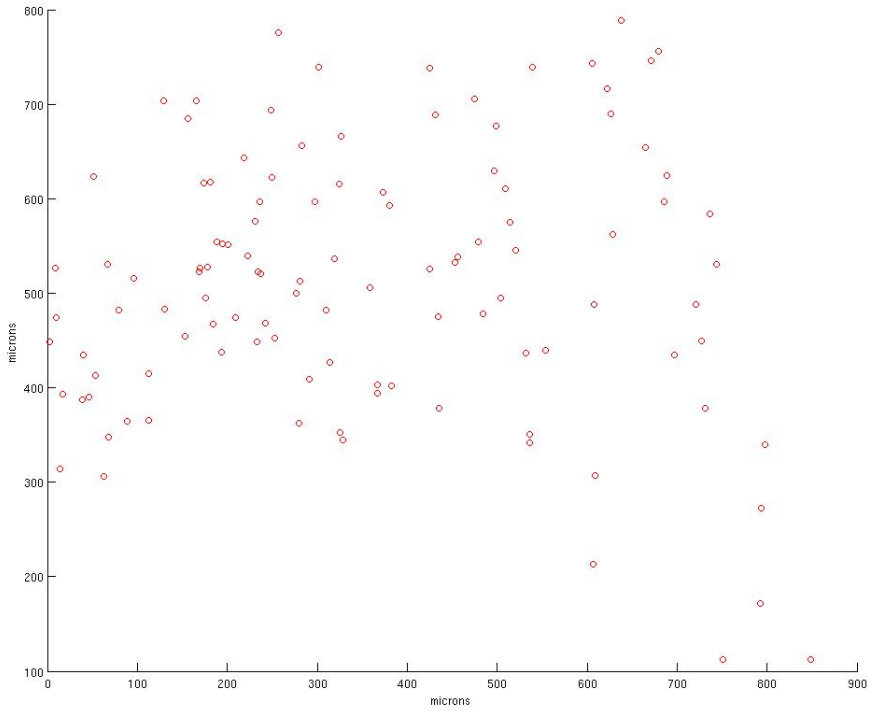


Figure 3.10: Distribution of Neuropile Model Cells across the LGC Space

model. As was pointed out in Chapter II, the original visual cortex model had a simple LGC model represented by a linear array of 201 soma-only compartmental models. These cells connect to the visual cortex model using a multitude of varicosities [6] [7]. The new LGC model is connected to the cortex via this linear model.

The model setup maps a 200 micron radius circle within the LGC model, shown in Figure 3.12 to the 201 linear LGC cells, as seen in Figure 3.13. We refer to this disc as the 2D LGC Patch. It is used in both this sub-section and the following, and never changes. The result is a mapping through the LGC using only about 650 of the total LGC cells.

The mapping process involves centering the 200 micron radius circle about the point $(450, 500)$ in the LGC, with the origin taken as the lower left-hand corner of the LGC in Figure 3.12. This disc is divided into 201 vertical strips and a vector is then

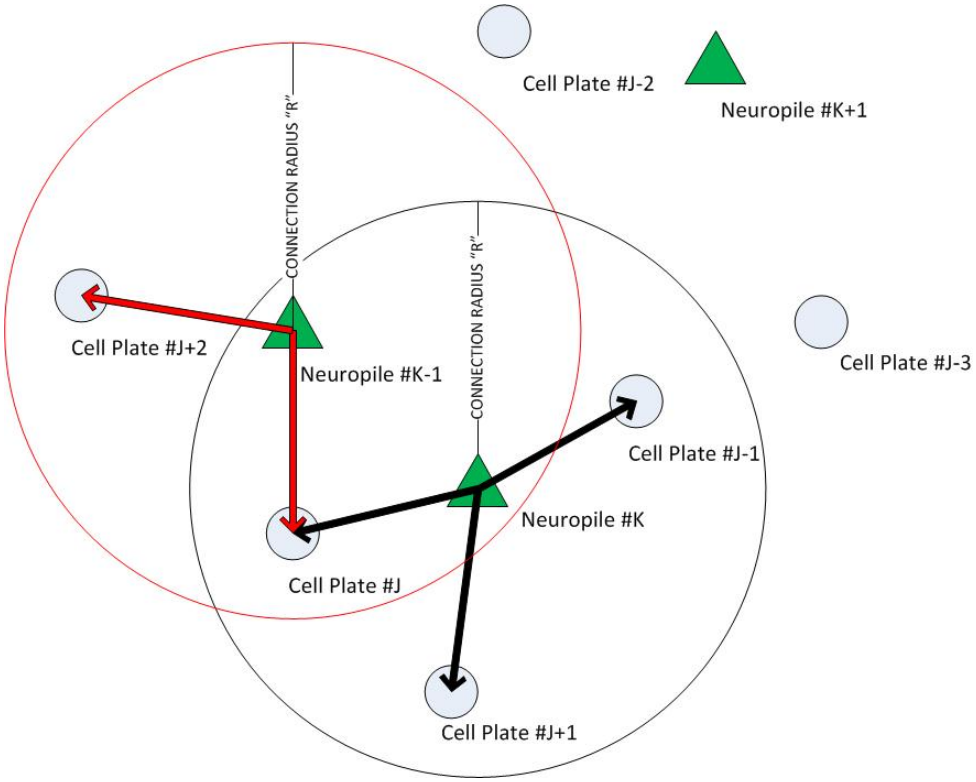


Figure 3.11: Forming Synaptic Connections in the LGC Model

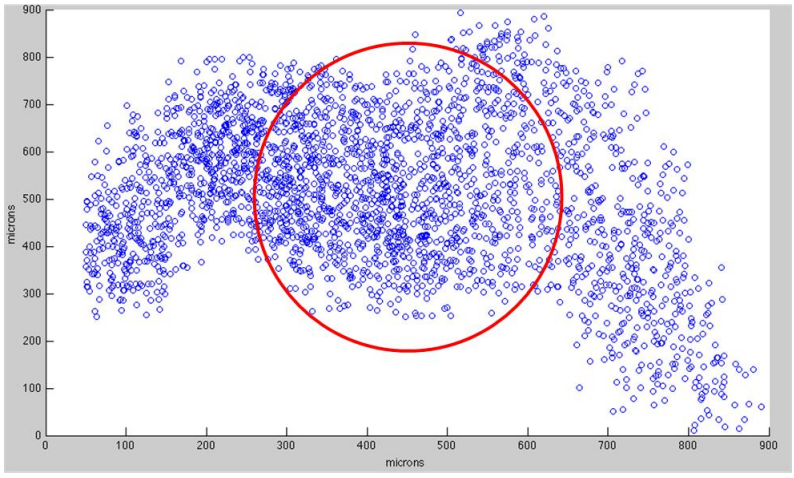


Figure 3.12: LGC Model Illustrating Circular Patch

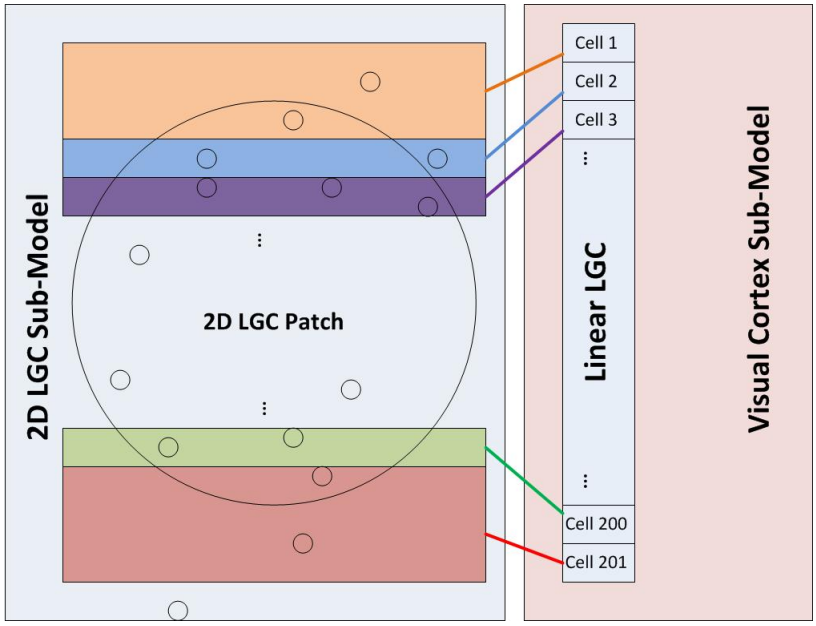


Figure 3.13: Mapping the LGC Model to the Linear LGC in the Visual Cortex

produced specifying the upper ceiling of each strip. The middle 199 strips are of the same width. The first and last strips are wider so as to assure LGC cells within the connection radius for the retina (see the next subsection) but outside the 200 micron radius, are successfully connected to the linear model.

The connections from each cell plate neuron to the appropriate linear LGC cell are of the excitatory kind. Excitatory signals from the 2D LGC model are generated by input from the retina. These signals will, when strong enough, trigger a wave of activity in the visual cortex. It is this wave that gives us information about the nature of the retinal input. To ensure effective communication between the LGC and the visual cortex, the synaptic connection strength value used by the excitatory connections must be calibrated. If the synaptic strengths are too weak, no wave will be generated in the cortex when valid input is provided. If the connections are too strong, then this will trigger a wave in the cortex too early and corrupt analysis of the results. This calibration process is discussed in Chapter IV.

3.5.2 Connecting the LGC Model to the Retina Model

In this section, the general procedure for connecting the retina to the 2D LGC model is described. Figure 3.14 demonstrates the association between the retina model and the LGC. This section deals with the blue arrows connecting the retina to the LGC model. As with the previous subsection, there are a number of different connection topologies that could be utilized. Here, a geometrical approach is applied whereby the constituent cells of the retina are mapped to the LGC neurons via circular regions in both the LGC and retina.

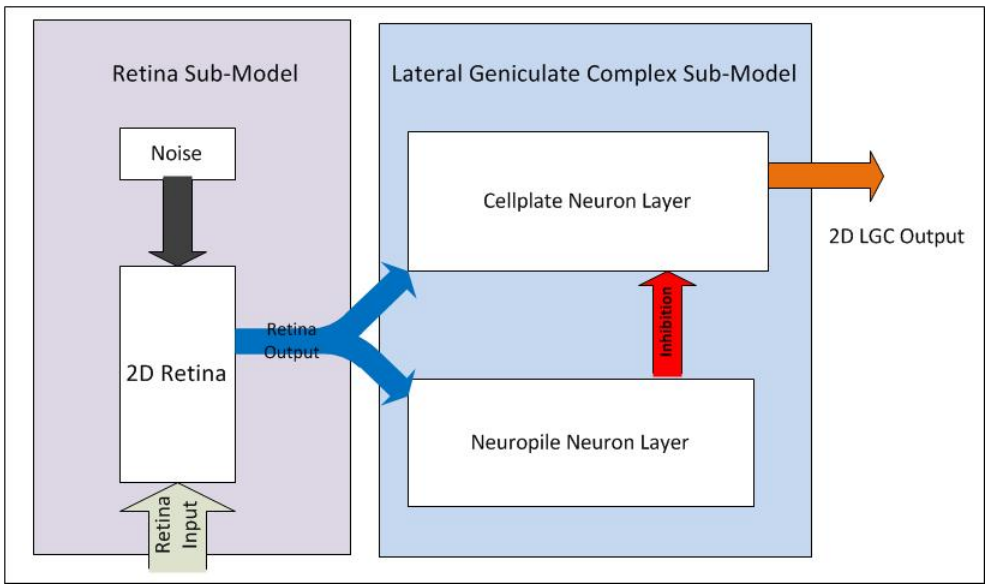


Figure 3.14: Mapping the LGC Model to the Retina Model

The blue arrows in Figure 3.14 represent synaptic connections from the retina model into the lateral geniculate complex model. The arrow splits, indicating that synaptic connections are sent from the retina to both the neuropile and cell plate cell layers. Recall that the neuropile cells form inhibitory synaptic connections within the LGC (the red arrow in the LGC Sub-Model). Production of inhibitory signals within the LGC relies on the neuropile cells receiving signals from the retina, and the procedure for accomplishing this is discussed first.

In Figure 3.15, the mapping procedure from the retina to the LGC neuropile neurons is shown. This process involves mapping a circular region in the retina (red

circle) to neuropile cells in the LGC model. Retinal cells within the mapping radius r are stepped through, their coordinates extracted, and distance from each neuropile in the LGC is computed. If that distance is below a particular threshold, an excitatory synaptic connection is formed from the retinal cell to the neuropile. Otherwise, no connection is made. The choice of threshold distance is discussed later when we calibrate the FVSM.

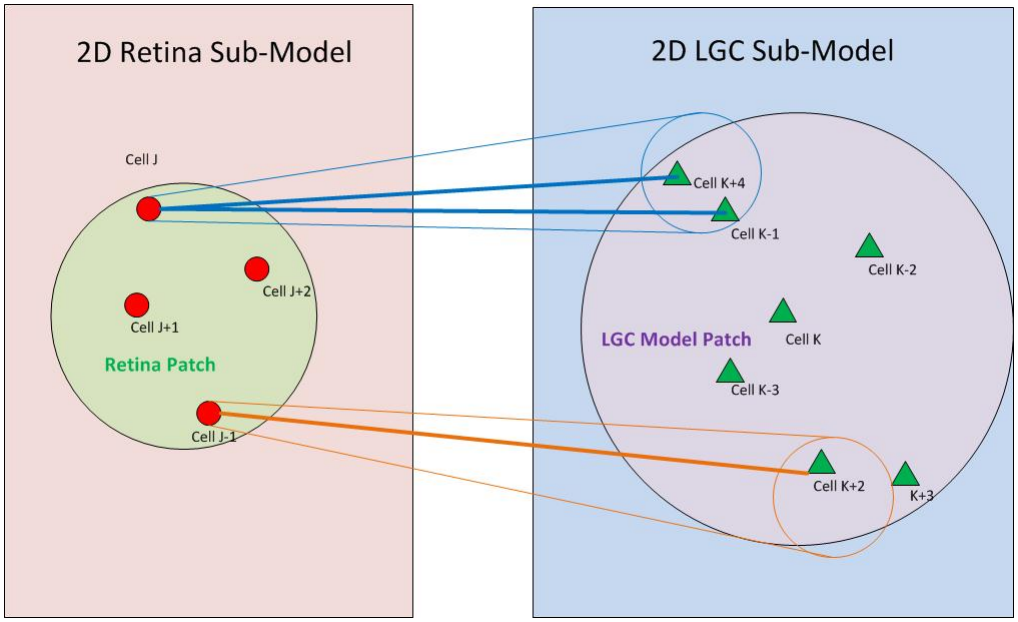


Figure 3.15: Fusion of LGC and Retina Models

This completes the connections from the retina cell models to the LGC neuropiles. Now, the cell plate cells must receive excitatory connections from the retina cells as well. The process for accomplishing this is identical to that described above, but with the neuropile cells replaced with cell plate neurons and the radius of influence r having a different value.

3.5.3 Fused Visual System Model

In this chapter, we have progressed from the cellular level models to the regional network models. With the completion of Section 3.5.2, we have successfully formulated the Fused Visual System Model (FVSM). This system level model was our main

goal throughout this chapter, and Figure 3.16 shows its block diagram. From the material covered thus far, each sub-model and its connections will be familiar to the reader.

For the noised version of the FVSM, two Gaussian white noise generators are needed. They are visible in Figure 3.16. The defining parameters for these generators are their variances and means. Table 3.4 shows these chosen variances and means. For all simulations described herein, unless noted otherwise, these values are used in the noise generators. Figure 3.16 represents the "Noised" version of the FVSM. Removal of the Gaussian noise generator from the retina model and removal of the inhibitory synaptic connections (red arrow) in the LGC Sub-Model yield the simpler "Noiseless" version of the model. In said noiseless system, the neuropile cells have no function in the model. The noiseless system retains a noise generator in the visual cortex, however. This could, if desired, also be removed.

This system-level model has a number of parameters that must be configured. All synaptic connections have a strength (also called a weight) and each cell also has a "radius of influence" when forming synaptic connections. These radii were encountered in this chapter already. Table 3.5 notes these parameters, where they reside in the FVSM, and key information about them. We will look at the values for these parameters and their configuration in the beginning of Chapter IV.

In Table 3.6 the number of each model cell in the FVSM is provided. These numbers are for all three sub-models and are valid for all simulations presented in the results section of this paper. For the LGC model, the values in parenthesis are the number of cells in the model that are connected to the retina and/or visual cortex during simulation. Due to the circular geometric mappings chosen, some LGC cells are outside of the mapping, so are not used during simulations.

Table 3.4: Noised FVSM White Noise Variances

Generator Location	Variance	Mean
Retina	3×10^{-11}	0
Visual Cortex	4×10^{-10}	0

Table 3.5: Properties of the FVSM and Important Notes

Location	**Association**	**Notes and Description**
Retina	Retina Patch Size	Radius of retinal patch in Figure 3.15
Retina - LGC Interface	Retina to Neuropiles	Synaptic Connection Strength
Retina - LGC Interface	Retina to Cell Plates	Synaptic Connection Strength
Retina - LGC Interface	Retina to Cell Plates	Cellular Radius of Influence
Retina - LGC Interface	Retina to Neuropiles	Cellular Radius of Influence
LGC Model	LGC Patch Size	Patch radius in Figures 3.12, 3.13, 3.15
LGC Model	Neuropiles to Cell Plates	Synaptic Connection Strength
LGC Model	Neuropiles to Cell Plates	Cellular Radius of Influence
LGC - Cortex Interface	LGC to Linear LGC	Synaptic Connection Strength

Table 3.6: Cell Numbers by Type in the FVSM

Sub-Model	Neuron Class	Number of Cells
Retina	A_ON	54
Retina	A_OFF	55
Retina	B1	134
Retina	B2	136
Retina	B3	141
2D LGC	Cell Plate	1166 (501)
2D LGC	Neuropile	112 (36)
Visual Cortex	Lateral	368
Visual Cortex	Medial	311
Visual Cortex	Horizontal	20
Visual Cortex	Stellate	45

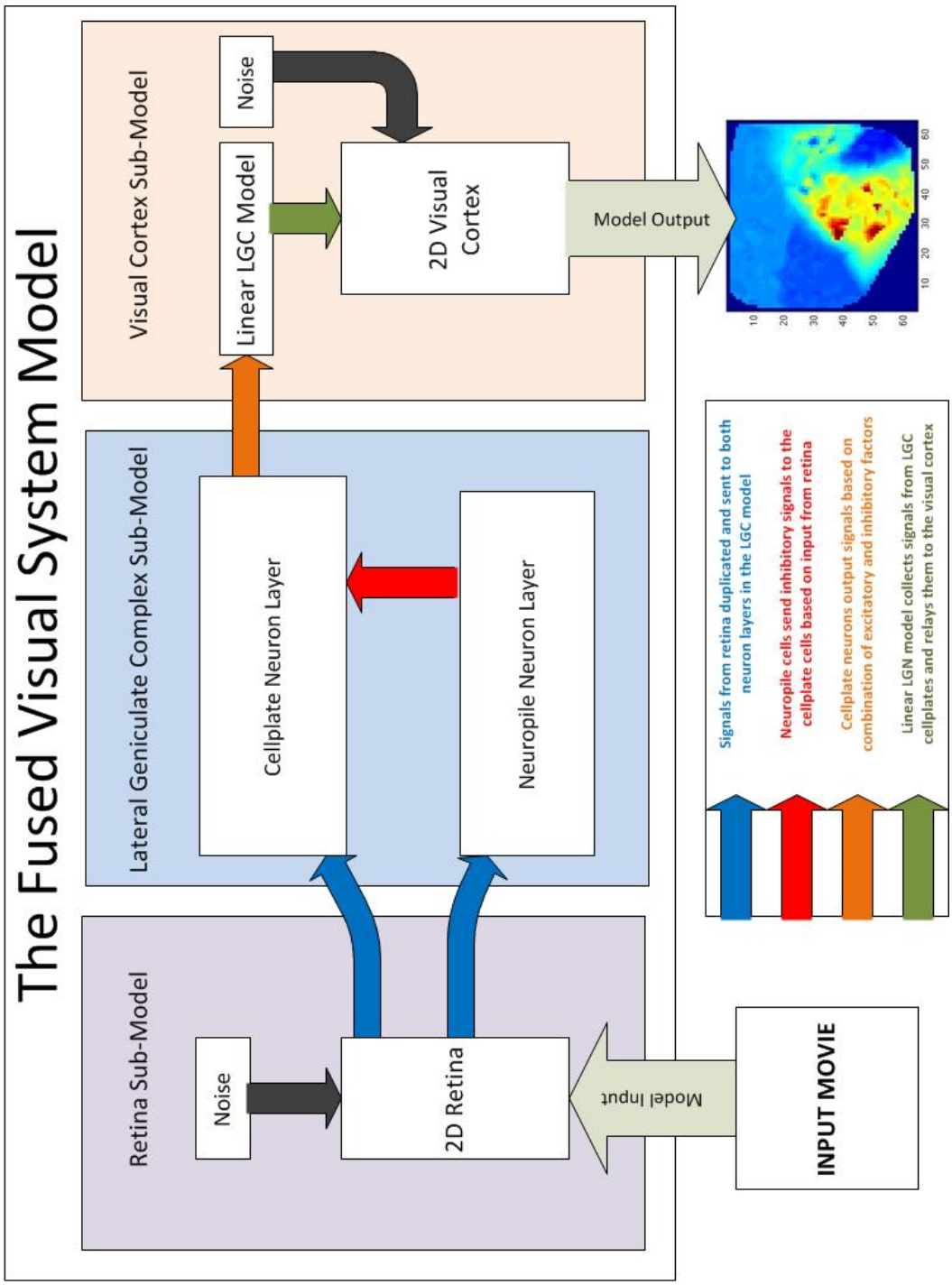


Figure 3.16: Block Diagram of the Noised Fused Visual System Model

CHAPTER IV
CALIBRATING THE FUSED VISUAL SYSTEM PARAMETERS

Chapter III involved model construction but left several major parameters unspecified. These parameters are considered again in this chapter, where we calibrate their values based on the nature of our research objectives. In general, the parameters specified herein are from the regional and system levels, not the cellular level.

4.1 Geometric Patch Sizes and Radii of Influence

Back in Section 3.4.2 we introduced the "radius of influence" as it pertains to the neurons in the LGC Model. We also made mention of the Retinal and LGC patches. The latter patches are defined by their respective radii and a center point. In the case of the retina model, the retinal patch taken is the same as in [5] and [12]. Hence, we have a retina patch of radius circa 50 microns. For the LGC patch, a scaling factor of 4 was chosen and applied to the retina patch radius. This yields an LGC patch radius of 200 microns. As mentioned in the last chapter, this circle is centered at $(450, 500)$ with respect to the lower left corner of the LGC density patch [2] [3].

As far as the patch radii are concerned, the choices for size were made in a somewhat arbitrary manner. The major goal in selecting these parameters was to keep the number of neurons (and thus the number of time-consuming computations involving synaptic connections) to a minimum while still providing sufficient complexity in the model to observe new behaviors from the FVSM.

In a biological sense, neurons tend to form local networks dedicated to particular processing tasks. This idea should be familiar by now, as both the LGC and the visual cortex can be viewed in a similar light. From the notions of graph theory, each neuron can be viewed as a node in a larger network (graph). Each directed edge of the graph then represents a synaptic connection from the pre-synaptic neuron to the post-synaptic neuron. According to the Neuron Doctrine, the direction of synaptic communication is largely fixed, so in general communication signals across the synapses travel in the pre to post neuron direction exclusively [3] [14] [15]. Our mathematical models behave in exactly the same way.

How do we decide if a given neuron (one node in our graph) is to form a synaptic

connection with a particular neuron in the same network? One of the simplest ways is to decide based on geometric distances. The larger the distance between two neurons, the less likely a synaptic connection will exist between them. As a result, a simple distance computation as done in Chapter III can be used as a test for influence. The radius defined for this test is called the neuron's radius of influence [3]. It is important to note that the "graph" we are using to represent the neural network is comprised of more than one neuron type. For instance, in the LGC model, we have the neuropile class and cell plate class. Hence, representing the LGC model in network form results in two kinds of nodes in the network. This translates to having several different node classes. Figure 4.1 illustrates an example digraph with two node types, denoted as red and blue. The connections are directed according to the arrowheads on the graph edges.

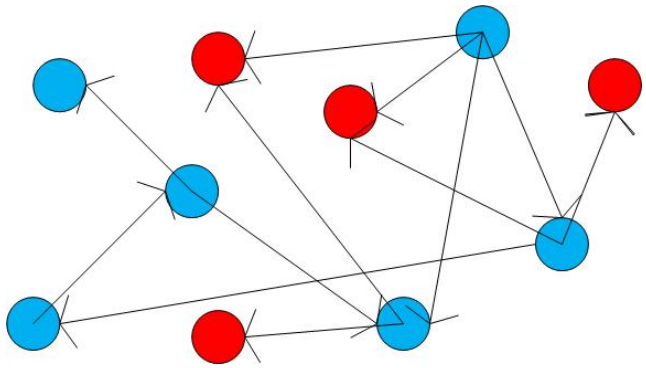


Figure 4.1: Two-node directed network example

There are rules regarding interactions among neuron types. An example can be found, again, in the LGC model. During construction of the LGC, we stated that neuropile neurons can form synaptic connections with cell plate neurons. However, the cell plate neurons do not form synaptic connections with neuropile cells or other cell plates in our model. Also, neuropile neurons never connect with one another synaptically in this model. As a result, we see that making a decision about which neurons connect synaptically with each other is more complex than simply finding all neurons in a given radius of the chosen pre-synaptic cell. We must ensure that we only assign connections that are legal according to our rules. As an example, the connection rule used between the node types in Figure 4.1 is that blue nodes can

direct connections to both red nodes and blue nodes. However, red nodes make no connections with other nodes.

In choosing the radii of influence for the model LGC, it was decided to use a fixed radius for each class exhibiting pre-synaptic connections. Therefore, each class of neuron has a fixed radius of influence irrespective of its location on the 2D model surface or other conditions during simulations. This is, of course, not the only approach. In fact, it has been suggested [3] that the radius of influence of cells in the freshwater turtle LGC may vary in an inverse manner with the cellular densities. Future simulations like those described herein should investigate this suggestion. For now, we define the radius of influence of the neuropile class with respect to the cell plate neurons as a constant positive real number R_{lgc}.

In the LGC model, the radius of influence is simple since all of the neurons reside on the same plane. However, the situation can be more complex, as is the case when connecting the retina model to the LGC model. Noting that the LGC patch is 200 microns in radius and the retinal patch is only $\frac{1}{4}$ this value, it is clear that the LGC and retina patches must be mapped to one another. Since both patches are circular, the process is still rather straightforward. We scale the retinal patch by a factor of 4 and overlay it on the LGC patch, sans rotation, as was shown in Figure 3.15. Once this mapping is defined, the applicable radius of influence can be applied to find and establish all relevant synaptic connections from the retina neurons to their LGC counterparts.

Our model description of the LGC indicates that both the neuropile cells and the cell plate neurons receive post-synaptic connections from the retina. As a result, each neuron class in the retina has a defined radius of influence (R_{np}) with respect to the neuropile cell type and another radius of influence with respect to the cell plate neurons (R_{cp}). For simplicity, our model uses the same R_{cp} and R_{np} values for each retina neuron class. In other words, R_{cp} is the same value for each of the "A_on", "A_off", "B1", "B2", and "B3" neurons in the retina [5]. The main reason for this simplified approach is that no biological information was available to contradict the assumption.

The table below lists the finalized radii of influence for the LGC model and its interface with the retina. These are the parameter choices used in the simulations

described later in Chapter V.

Table 4.1: FVSM Radii of Influence and Patch Size Values

Parameter	Chosen value (microns)
LGC Patch Size	200
Retina Patch Size	50
R_{cp} $\forall$ retina neuron types	15
R_{np} $\forall$ retina neuron types	130
R_{lgc}	140

4.2 Synaptic Connection Strengths

The synaptic connections discussed above not only have a direction, but also a corresponding strength. In certain instances, neuron models define a strength range $S = [0, K]$ of positive real values. Here, K would be the maximum synaptic influence possible for the pre-synaptic neuron on the post-synaptic neuron, while a strength of 0 would indicate no synaptic connection. Another way of describing the meaning of synaptic strength is to look at this value as how much "weight" the post-synaptic neuron ascribes to signals from a particular pre-synaptic neuron [14]. Often the range $[0, K]$ is the output of some function relating certain properties of the neurons and model configuration. For instance, a linear function can be defined that ascribes a synaptic strength of K for post-synaptic neurons next to the pre-synaptic cell (here, the distance between the cells is very small) and decreases steadily as the distance increases, reaching a strength of 0 at the radius of influence boundary. Outside of the radius, the synaptic strength value is taken as 0. Such a linear mapping is shown below. In this mapping, d is the distance from the pre-synaptic cell, R is the radius of influence, and S is the synaptic connection strength.

$$S(d) = \begin{cases} K - d\frac{K}{R} & ; R \geq d \\ 0 & ; \text{otherwise} \end{cases}$$

The mapping used in the LGC model under investigation is defined as a step function where the synaptic strength within the radius of influence is taken as K and the synaptic strength immediately drops to 0 outside of the radius. This is illustrated by the following mapping.

$$S(d) = \begin{cases} K & ; R \geq d \\ 0 & ; \text{otherwise} \end{cases}$$

Of course, more advanced mappings are possible and, in some experiments will be better choices. The radial mappings discussed thus far work well connecting the retina to the LGC and associating neuropile neurons with cell plate cells within the geniculate. However, as discussed earlier, a different mapping scheme was chosen for connecting the LGC model to the visual cortex. Since the visual cortex model, described in [4] and [13], already has a simpler "linear LGC" model of 201 single-compartment neurons, the LGC patch was sub-divided into 201 slices of specific widths. All cell plate LGC neurons in slice $\omega \in [1, 2, ..., 201]$ were then mapped to cell ω in the linear LGC.

Thus, we are not using the notion of radius of influence here. Instead, all LGC cell plate neurons in a defined strip ω are pre-synaptic to only a single cell of index ω in the linear LGC model. There is, of course, an associated synaptic weight S_{lv} for all of these connections. We define S_{lv} formally in the next section. Table 4.2 summarizes our model synaptic connection strengths. The method used to establish these values will be described in the next section.

Table 4.2: Synaptic Connection Strengths for Noised FVSM

Parameter	Synaptic Strength
S_{cp} $\forall$ retina neuron types	9
S_{np} $\forall$ retina neuron types	.60
S_{lgc}	.7
S_{lv}	1

4.3 On Noise and Synaptic Strength Calibration

Determining the synaptic connection strengths in the previous section involves a binary search with parameter adjustments at the conclusion of each test simulation. Essentially, we "guess" what reasonable parameter choices would be and then test the result. Adjustments are then made to the parameter value(s) based on the simulation results and the process is repeated. The goal is for the FVSM to output a cortical

wave like the one shown in Figure 2.13 back in Chapter II. If the FVSM is of the noiseless variety (no noise in the retina) then the process is relatively easy. In this case, S_{np} is set to zero since inhibition is needed only if noise is present. This leaves S_{cp} and S_{lv} to be determined. From experience with the combined retina/VC model investigated by Neshadha Perera in [12], it was known that $S_{lv} = 1$ was an effective choice for synaptic weight between the LGC and the visual cortex. Hence, the only parameter we have to search for is S_{cp}.

To find an effective value for S_{cp} in the noiseless FVSM, one simply guesses a weight, say w, and runs a test simulation with input angle data. If a cortical wave is generated as seen in Figure 2.13, then that weight value is appropriate. If the choice causes too strong a cortical response, w must be reduced and tested again. If a wave is not generated, then w must be increased. After several iterations of this process, we will find a weight w_k such that $w_k = S_{cp}$ generates the desired cortex response. Ideally this weight is tested on multiple input values to ensure it works admirably on the whole data set.

The complexities begin when we look at the noised version of the FVSM. While we still make the assumption that $S_{lv} = 1$, we now have to find effective weights for both S_{cp} and S_{np} since the neuropile layer needs excitation from the retina in order to detect noise. The general notion, however, is that there should be a balance between inhibitory signals from the neuropile cells and excitatory signals from the retina to the cell plate layer. This balance exists when the inhibitory signals from the neuropile layer are, for the most part, suppressing the excitatory signals from the retina to the cell plate layer when the only input to the retina model is noise.

In the absence of input to the retina, all we have in the system is retinal noise. Hence, the approach used to find effective values for S_{cp} and S_{np} is to first use the noiseless model to determine an effective weight for S_{cp} with a number of inputs. Once this value is found, it is used in the noised FVSM and a binary search is executed to find S_{np} such that, when the only retinal input is noise, the LGC model "filters" the noise, preventing it from reaching the visual cortex and triggering a wave. The difficulty is that there are an infinite number of weights for S_{np} that will suppress the noise. Choosing too strong a weight will prevent the noised FVSM from responding to any retinal input. Hence, one needs to find the value of S_{np} that is sufficient to

cancel noise, but not so high as to cancel all retina signals destined for the visual cortex model.

CHAPTER V
SIMULATION RESULTS

The FVSM, developed in detail throughout Chapter III and calibrated in Chapter IV, was constructed to investigate how the introduction of the LGC component in the turtle visual system model influences the behavior of angle encoding and decoding by the visual cortex. In this final chapter, some results from these simulations and tests are presented and discussed.

In Section 5.1, we investigate the ability of the LGC model in the FVSM to suppress retinal noise. This is the major contribution of this model to the results obtained in [5] and [12]. Until the advent of the two layered LGC model of this thesis, realistic noise in the retina confounded attempts to detect retinal input from the movies described in Chapters II and IV.

In the last two sections, we look at experimental results from model simulations as described in [12]. There, the goal is to expose the Noised FVSM to all 12 angle inputs and record the resulting cortical waves. Independent component analysis is then conducted on the simulation results to investigate angle separability for several chosen angles.

5.1 Retina Noise Inhibition by LGC Sub-Model

Back in section 4.1 we provided the radii of influence for the retina-LGC interface. Those values are given in Table 4.1. From this table, we see that $R_{cp} < R_{np}$. This property of our model is very important, as it allows the LGC to execute its primary function in the FVSM: noise suppression in retinal responses. Our inputs of interest during simulations are localized discs that move across the retina. Their relatively small size in comparison to the retina patch indicates their detection is best done by a localized system. Also, the input signals from the localized discs are significantly stronger than the background noise, so noise in the region of the disc input should produce negligible effects. For visual demonstration, see Figure 2.10, which shows a time series image of cellular responses in the retina to the 0° input plus Gaussian noise of 4×10^{-8} variance. One can clearly see the input movie disc as it traverses the retina. The speckles are caused by the random noise from the retina Gaussian noise

generator. This particular simulation with the retina was conducted over 2 seconds. The simulations that we will see in the next sections were conducted with simulation times of 1.5 seconds, implying the input movie disc moves faster in our simulations than seen in Figure 5.1.

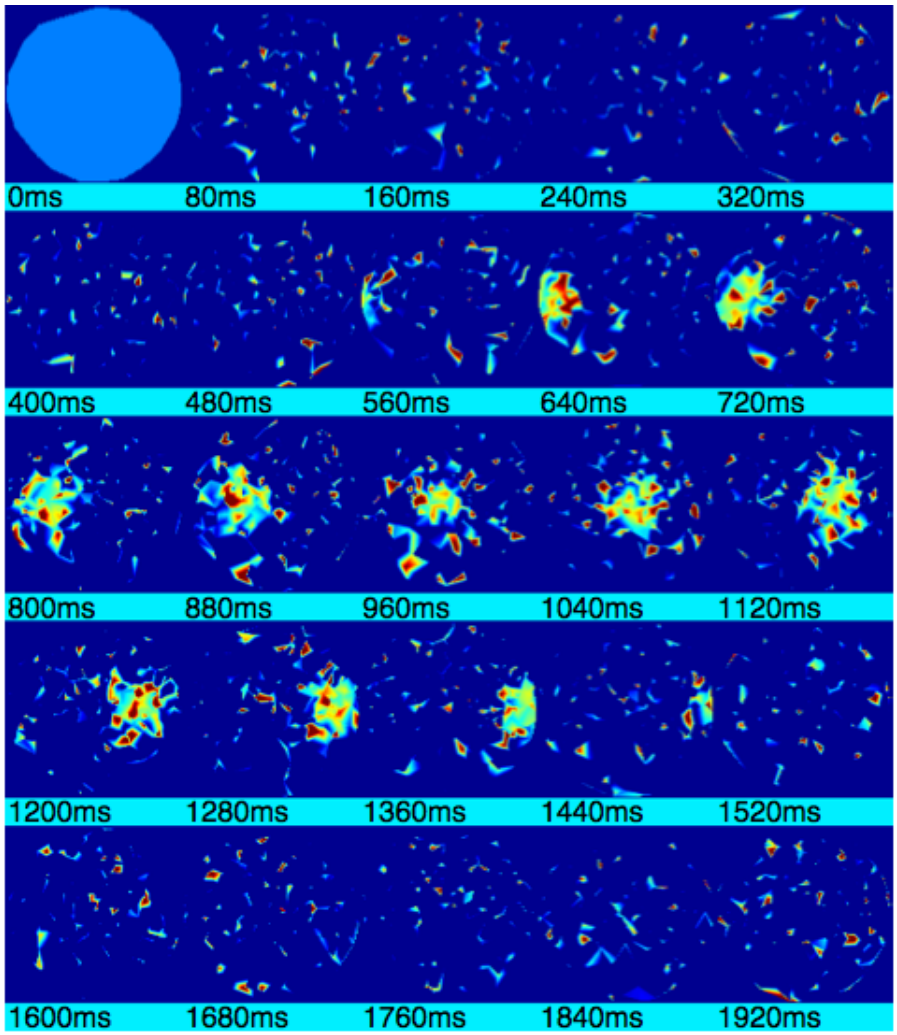


Figure 5.1: Retina Response to 0° Input Movie and Gaussian Noise

The cell plate neurons of the LGC model serve the desired localized detection purpose, as signified by R_{cp} being smaller than R_{np}. Effectively, the cell plate neurons, the most numerous cells in our LGC model, are detecting activity in small circular sectors of the retina. Of course, these cells detect localized noise as well. By

themselves, the cell plate neurons can't tell the difference between localized input and localized noise. Input from both sources is strong enough to elicit spiking behavior. However, by including inhibition signals from neuropile cells configured to detect activity across large areas of the retina, it is possible to counteract the effects of local noise on the cell plate neurons.

The noise used in our models is Gaussian with a particular variance and zero mean. Since the noise is by definition random, we can get a general estimate of its prevalence in the retina by sampling a large number of signals from the retina. Each neuropile cell does just that. The larger radius of influence R_{np} allows each neuropile cell to accumulate signals over a large area of the retina. This union of signals will be dominated by the noise, provided input to the retina is localized to small patches of cells, which as noted before is a valid assumption in our case. The resulting accumulated signals excite the neuropile cells, which then inhibit the cell plate neurons to reduce their sensitivity to detected local noise. This inhibition effectively blocks the retinal noise signals from exciting cortical waves in the visual cortex.

Figure 5.2 illustrates graphically the output of various configurations of the FVSM to demonstrate this noise-canceling process. In a) the output of the noiseless FVSM is shown for an angle input of 150°. Notice the cortical wave generated by the input movie. This is an expected result. In b) the noised FVSM was run without any movie input, but the inhibitory effects of the LGC neuropile cells were turned off by setting the synaptic connection strength between the retina and neuropiles to zero. One can clearly see a cortical wave in b). Since no input movie was used, the wave in b) is clearly generated by the Gaussian noise present in the retina of the Noised FVSM. Since cortical waves should not be generated by noise in the retina, this is a poor response from the model. In c), the inhibitory signals from the neuropiles to the cell plates are restored in the FVSM and the model is subjected to the same noise variance as in b). For this situation, no movie input was used. Notice that the noise-induced cortical wave found in b) is not present in c), indicating the inhibitory activity of the neuropile cells is suppressing the noise in the retina. Finally, in d) the model configuration used in c) is exposed to the 150° input movie used in a). As desired, we find in d) a cortical wave much like the one exhibited in a). Sub-figure d)

shows that the complete Noised FVSM is capable of suppressing noise in the retina while allowing movie input to excite the model visual cortex.

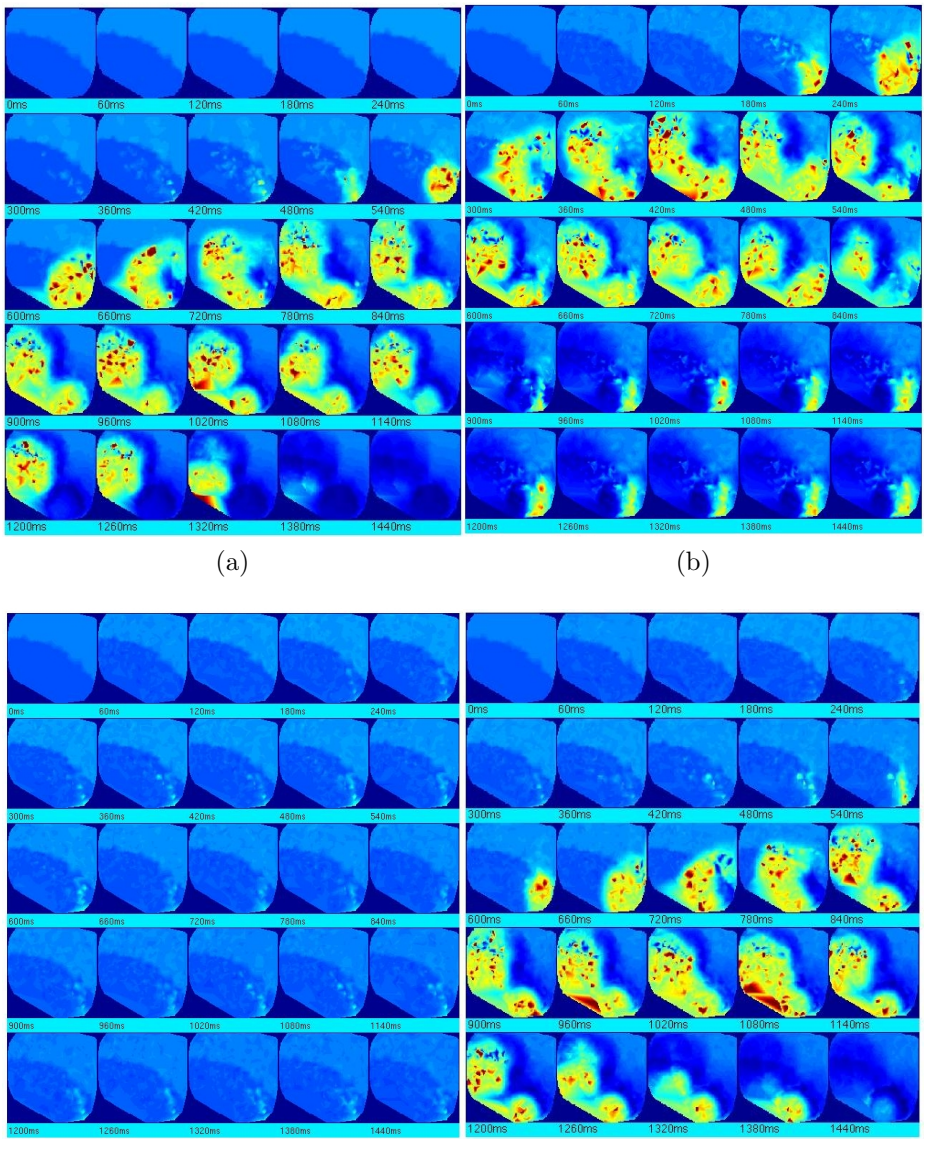


Figure 5.2: Demonstration of noise inhibition in the FVSM via the LGC Sub-Model

5.2 Results of Angle Input Data Trials

This section lists the visual cortex output from the Noised FVSM. Recall that the noised version of the Fused Visual System Model includes Gaussian noise in both the retina and visual cortex. As a result, the outputs here were obtained with full inhibitory effects in the LGC sub-model. In all cases, it can be seen from the figures that the LGC is effectively suppressing noise while providing adequate excitatory signals from the retina to trigger cortical waves in the visual cortex.

In Figures 5.3 through 5.14, the cortical waves resulting from the given input movies are displayed. Notice the differences in wave shapes, particularly in the first 700ms of simulation time among the twelve test angles. These differences are due to a combination of retinal/cortical noise and angle movie differences. From these differences, we investigate angle separability in the final section of this thesis.

It is constructive to note that the images shown below are one instance of each angle simulation. Since noise is present in the system, the visual cortex will respond in slightly varied ways from simulation to simulation, even if the input movie and parameters are kept the same. As a result, it is difficult to make decisions about angle separability based solely on the visual appearance of the waveforms. Many trial runs of the same angle are needed, along with principal component analysis, to make any determinations about the separability or non-separability of angles.

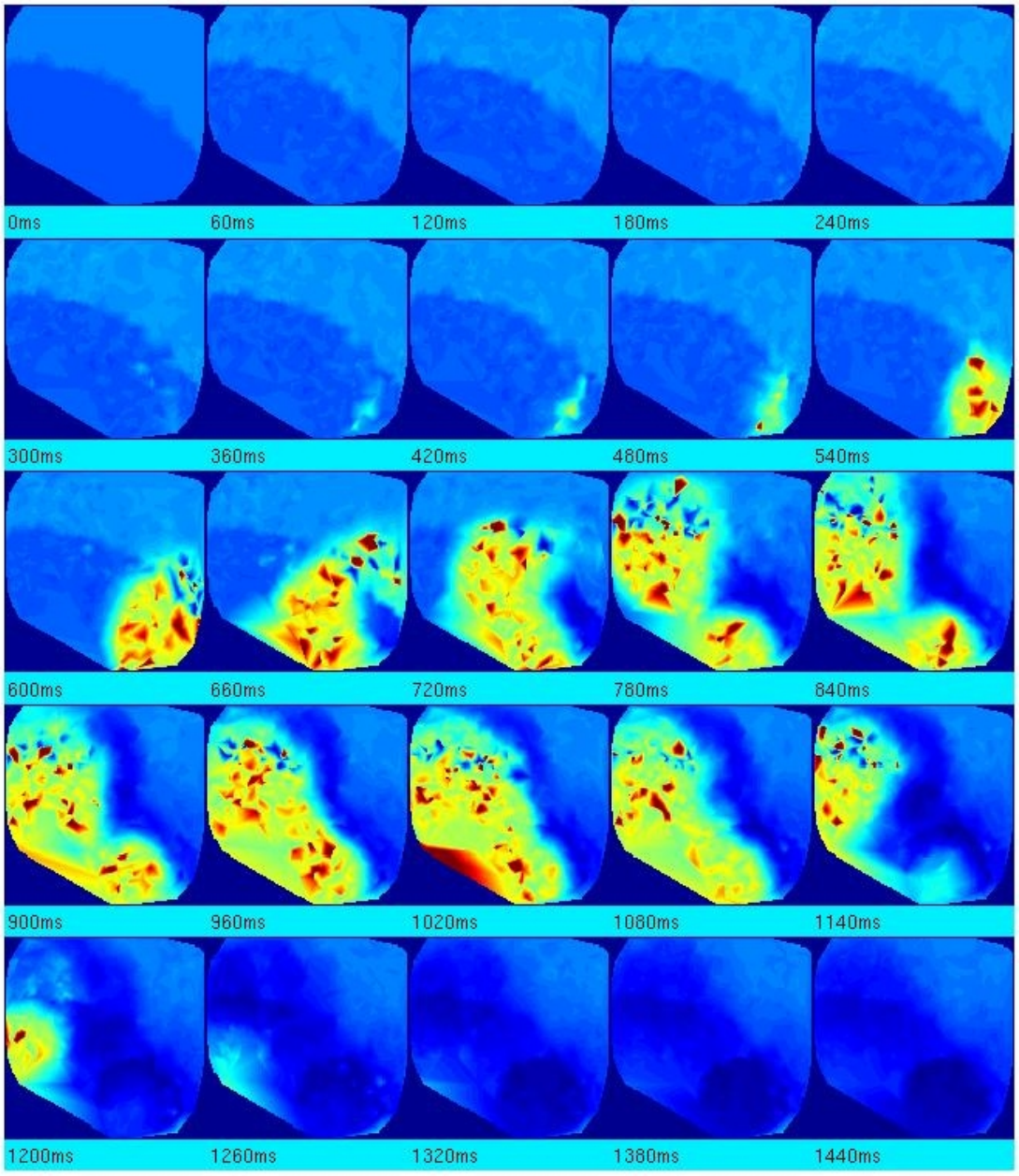


Figure 5.3: Noised FVSM Visual Cortex output for input movie 0°

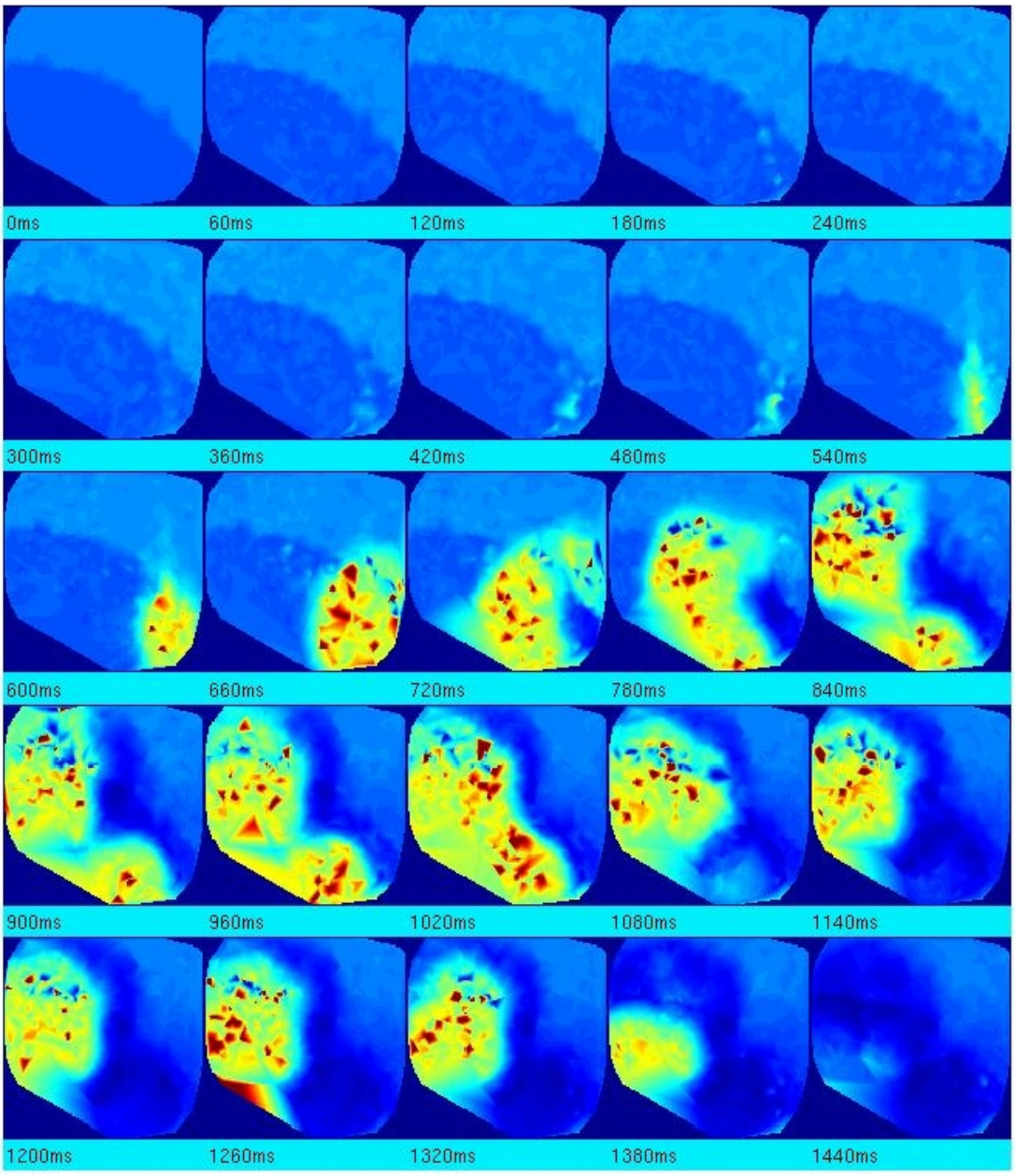


Figure 5.4: Noised FVSM Visual Cortex output for input movie 30°

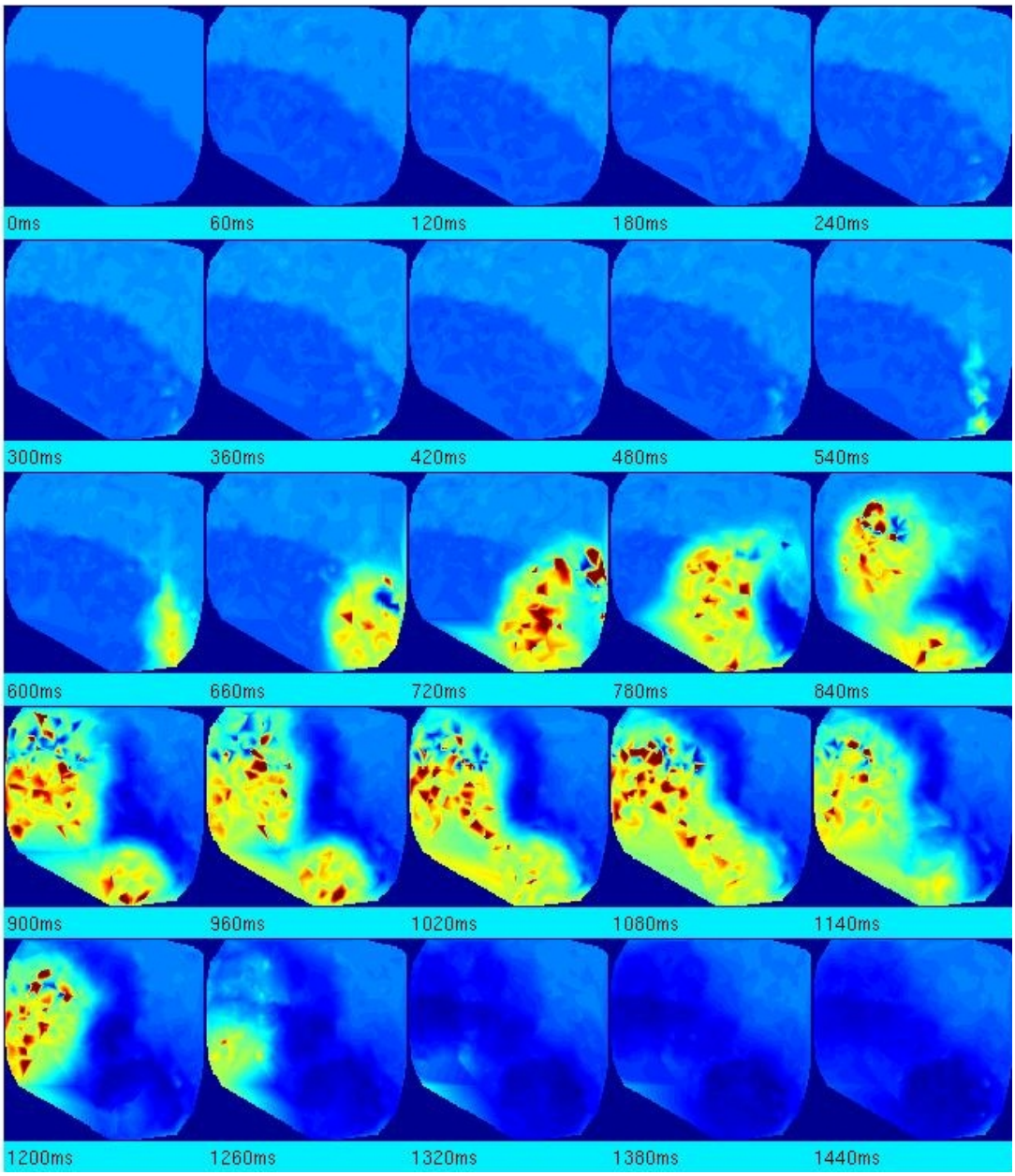


Figure 5.5: Noised FVSM Visual Cortex output for input movie 60°

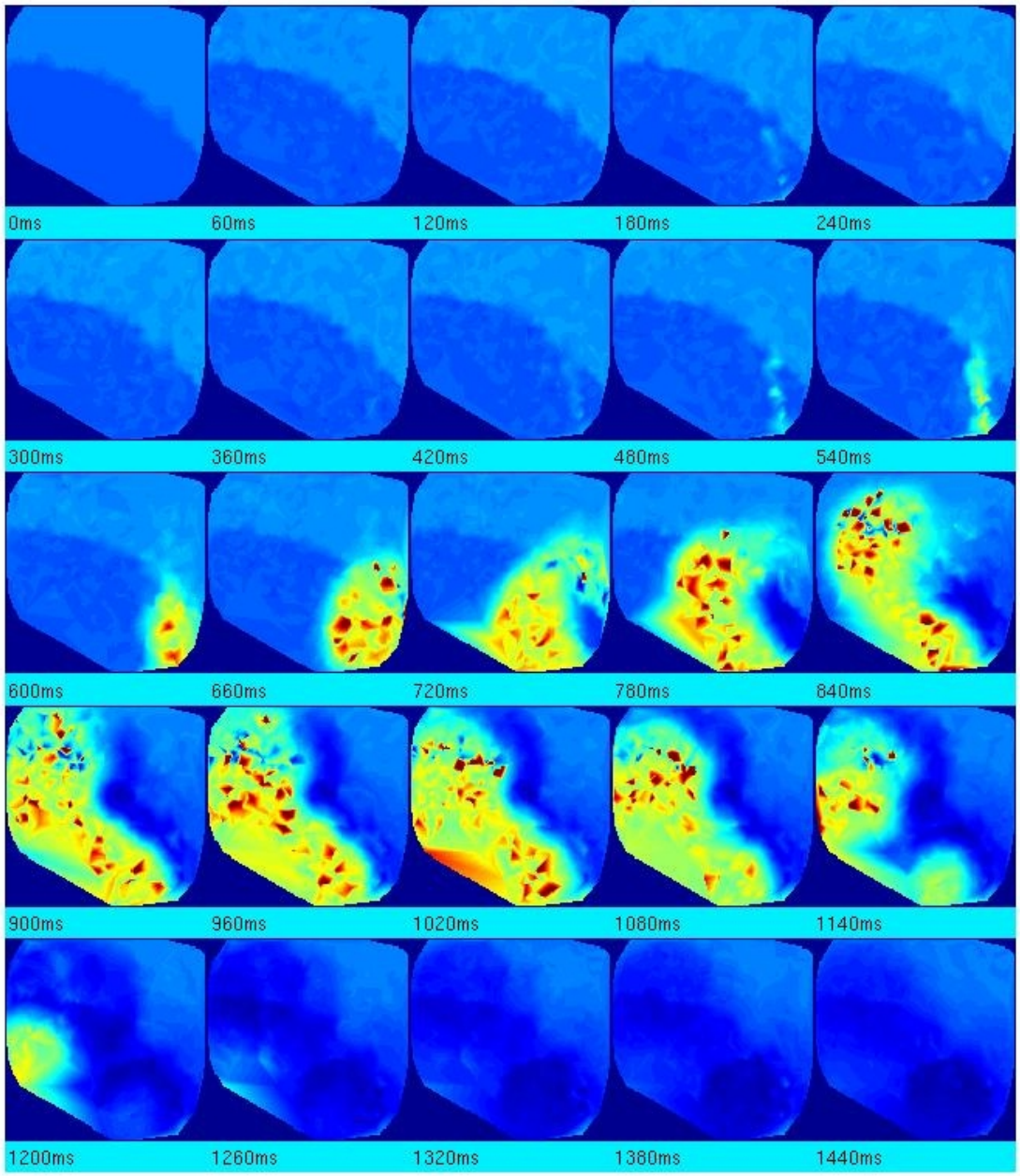


Figure 5.6: Noised FVSM Visual Cortex output for input movie 90°

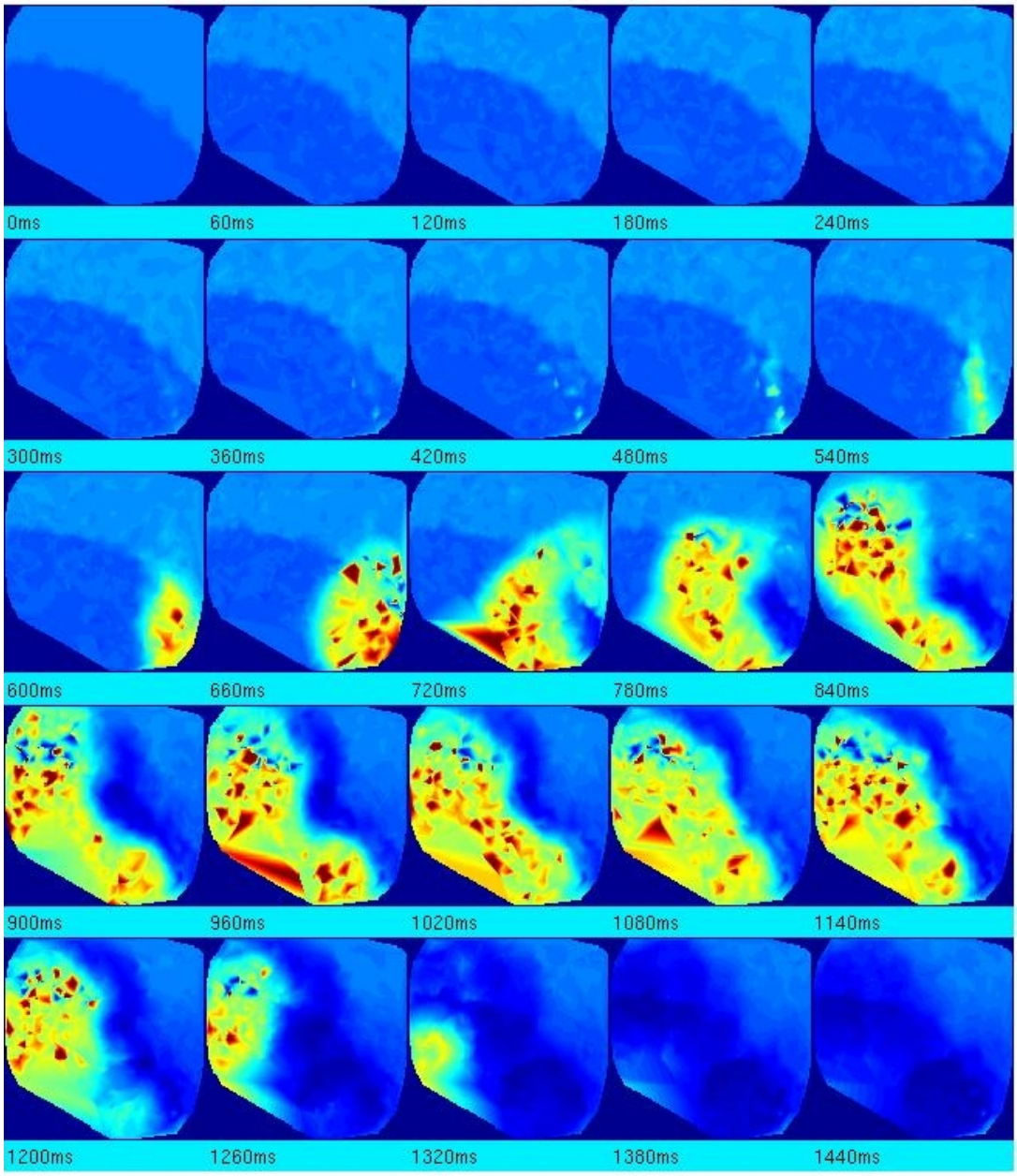


Figure 5.7: Noised FVSM Visual Cortex output for input movie 120°

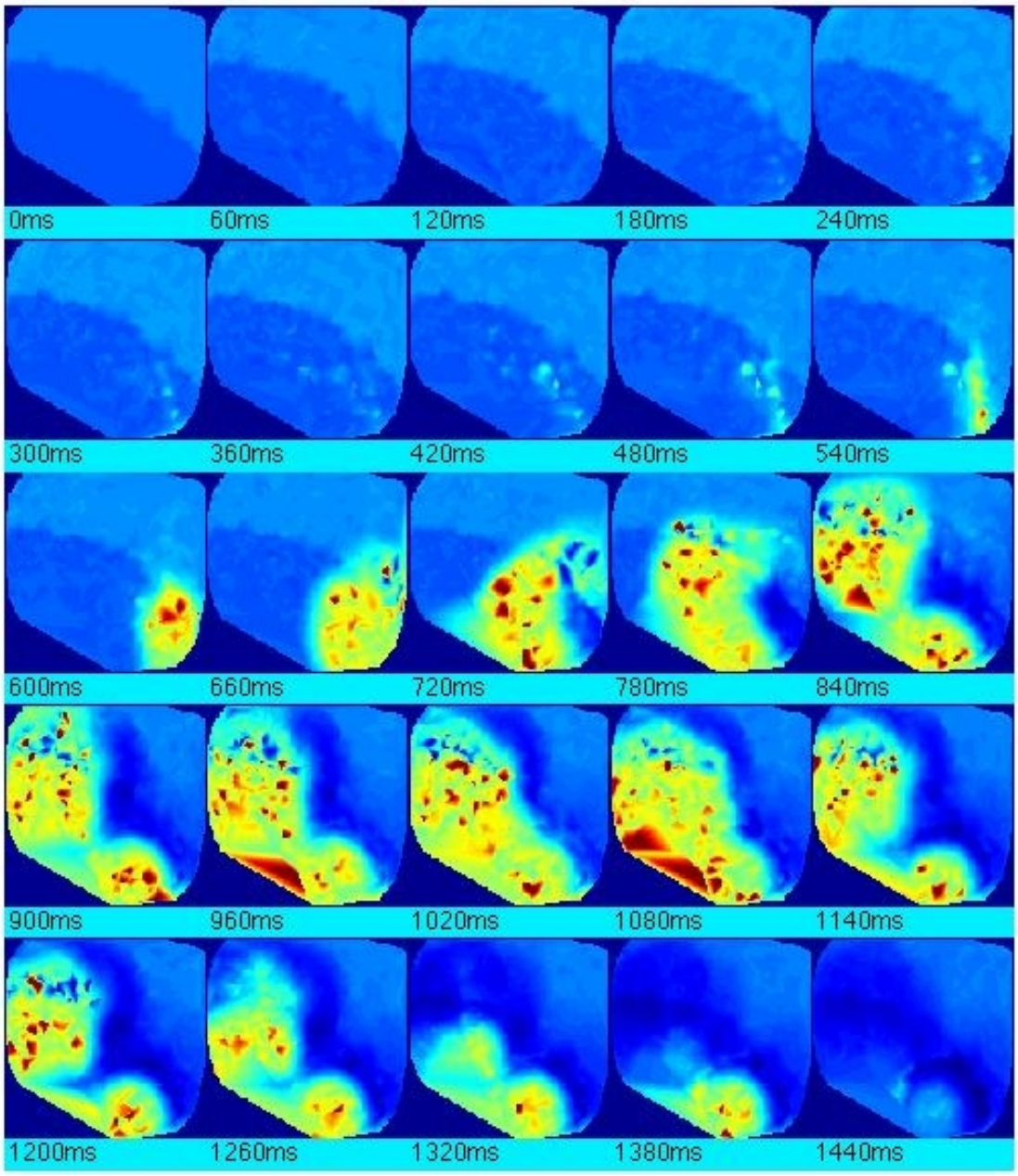


Figure 5.8: Noised FVSM Visual Cortex output for input movie 150°

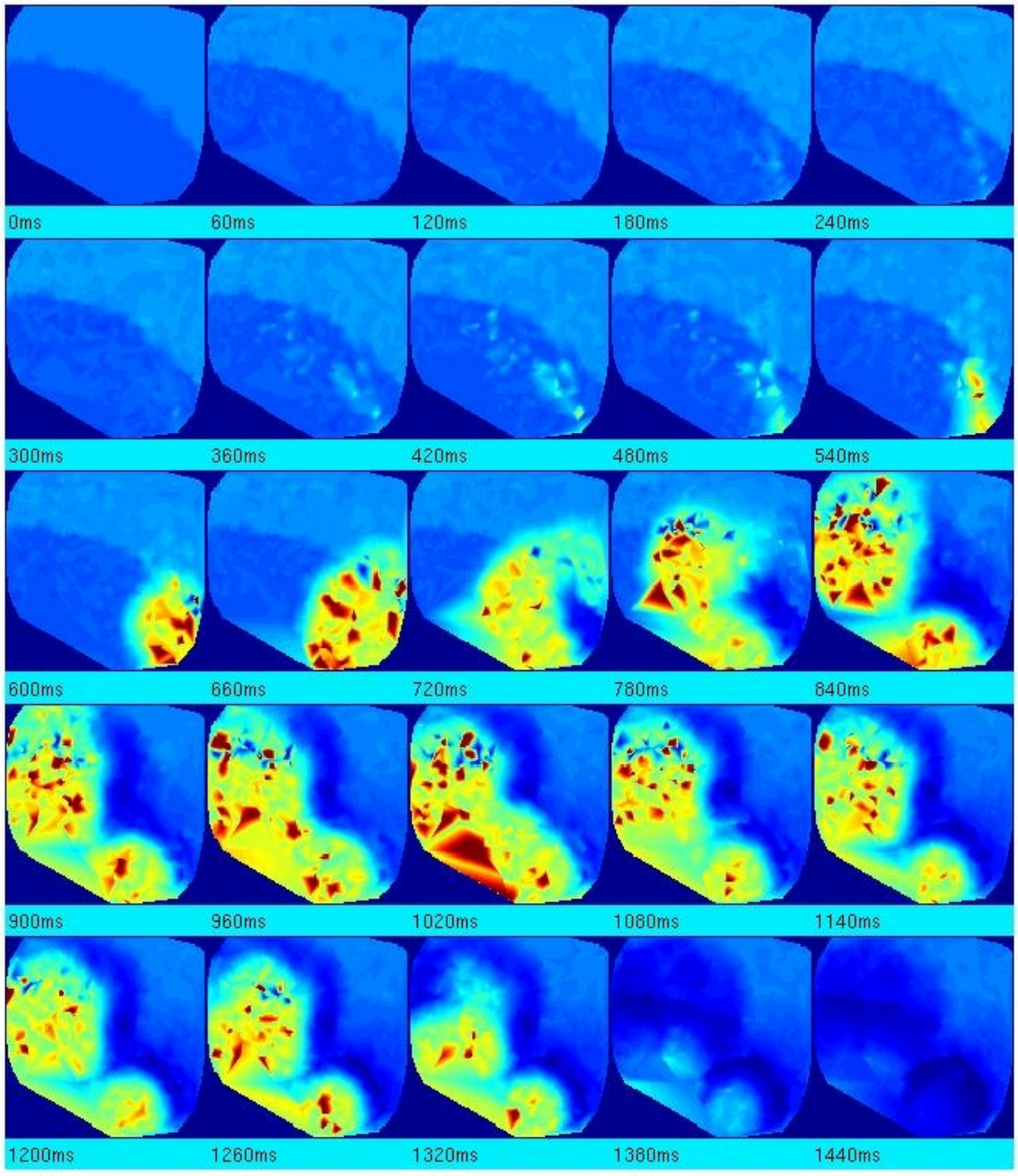


Figure 5.9: Noised FVSM Visual Cortex output for input movie 180°

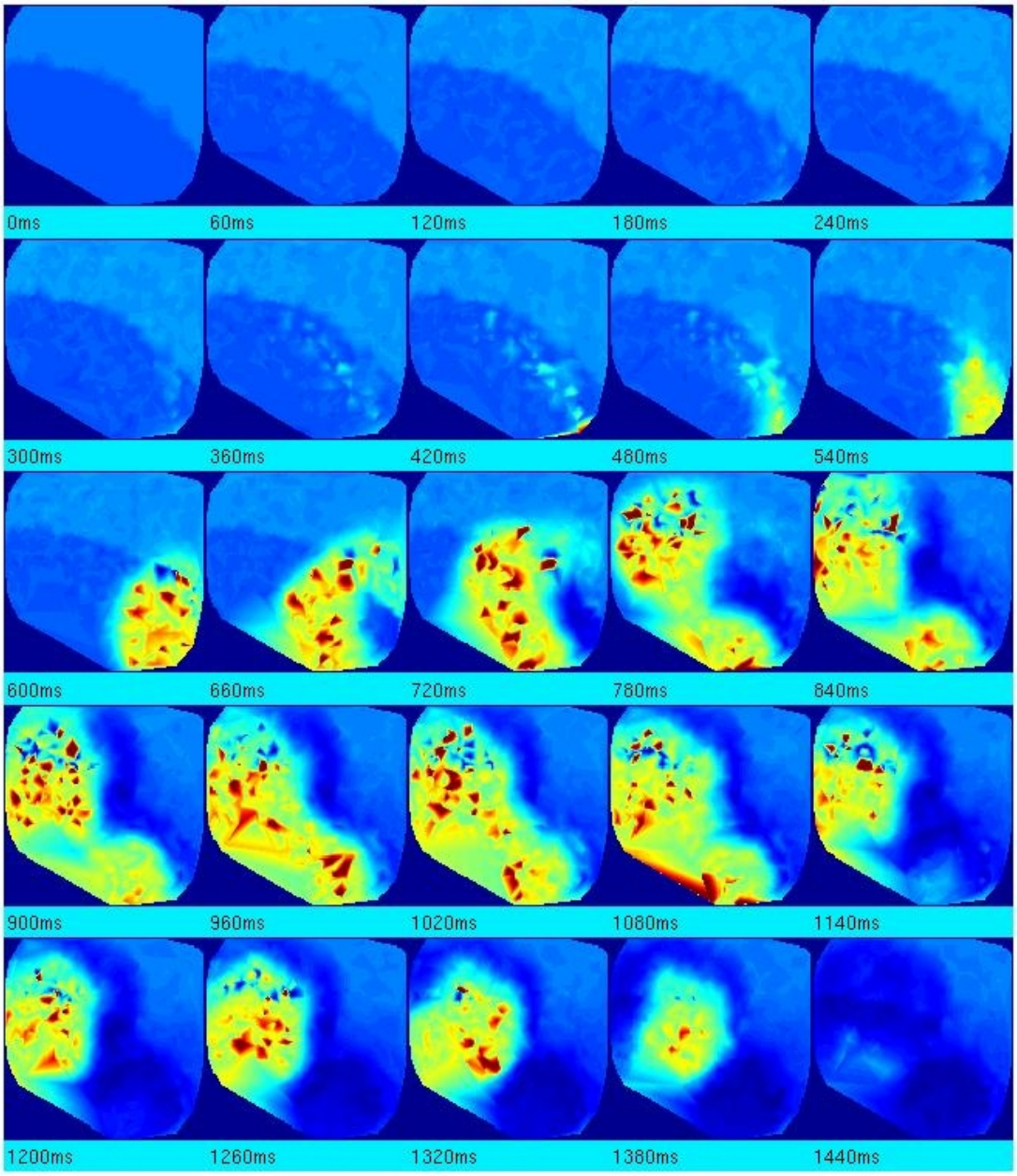


Figure 5.10: Noised FVSM Visual Cortex output for input movie 210°

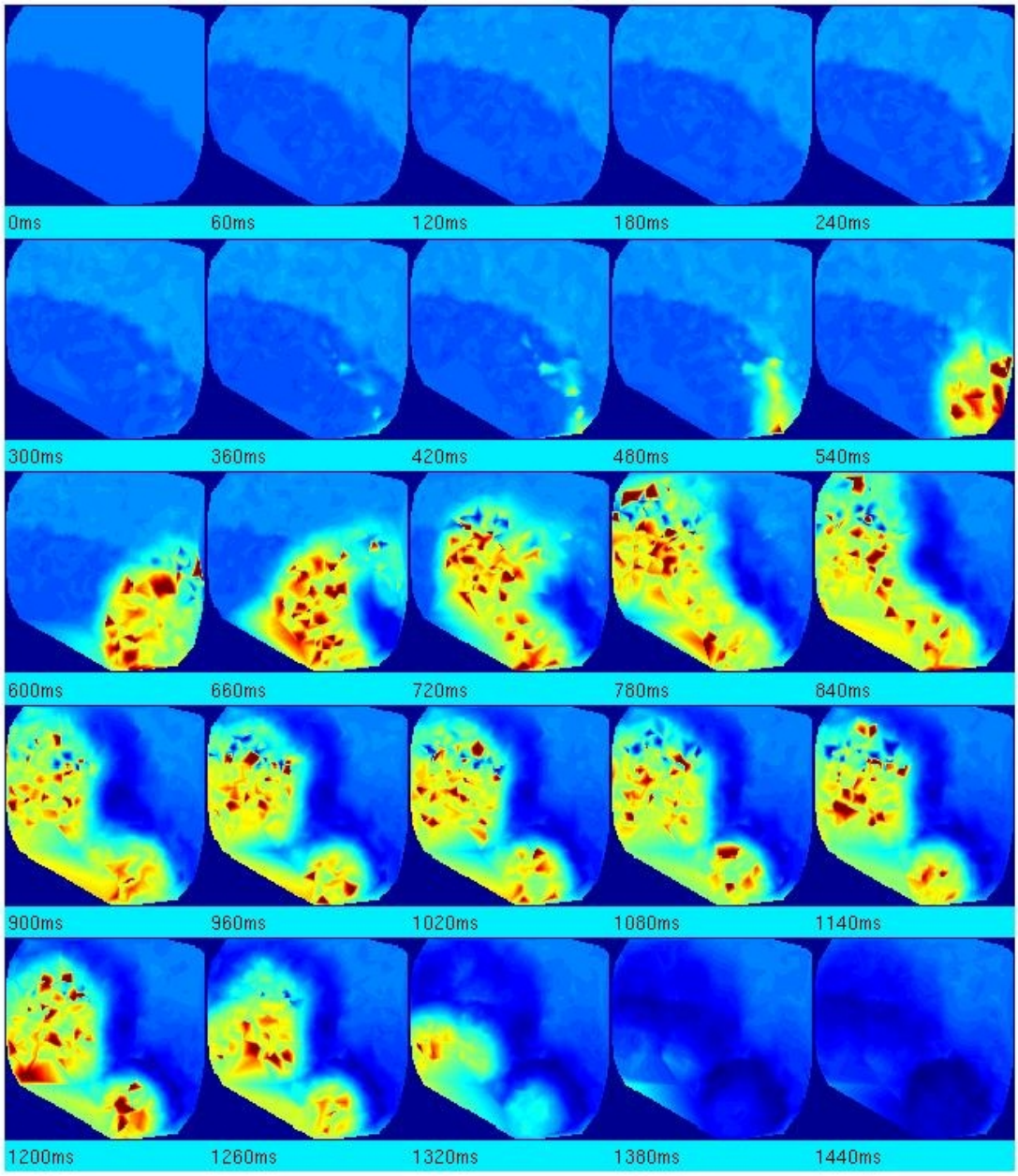


Figure 5.11: Noised FVSM Visual Cortex output for input movie 240°

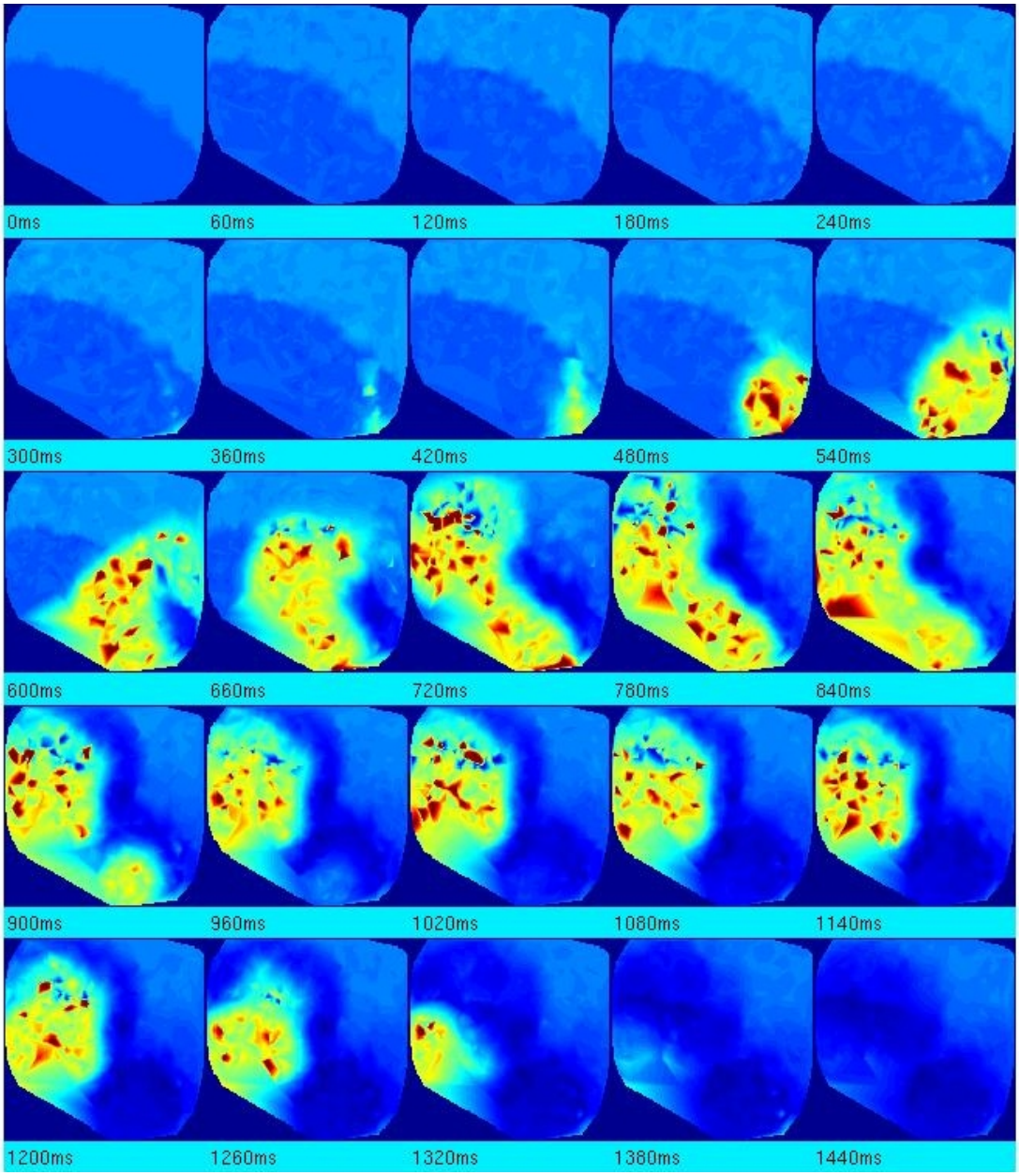


Figure 5.12: Noised FVSM Visual Cortex output for input movie 270°

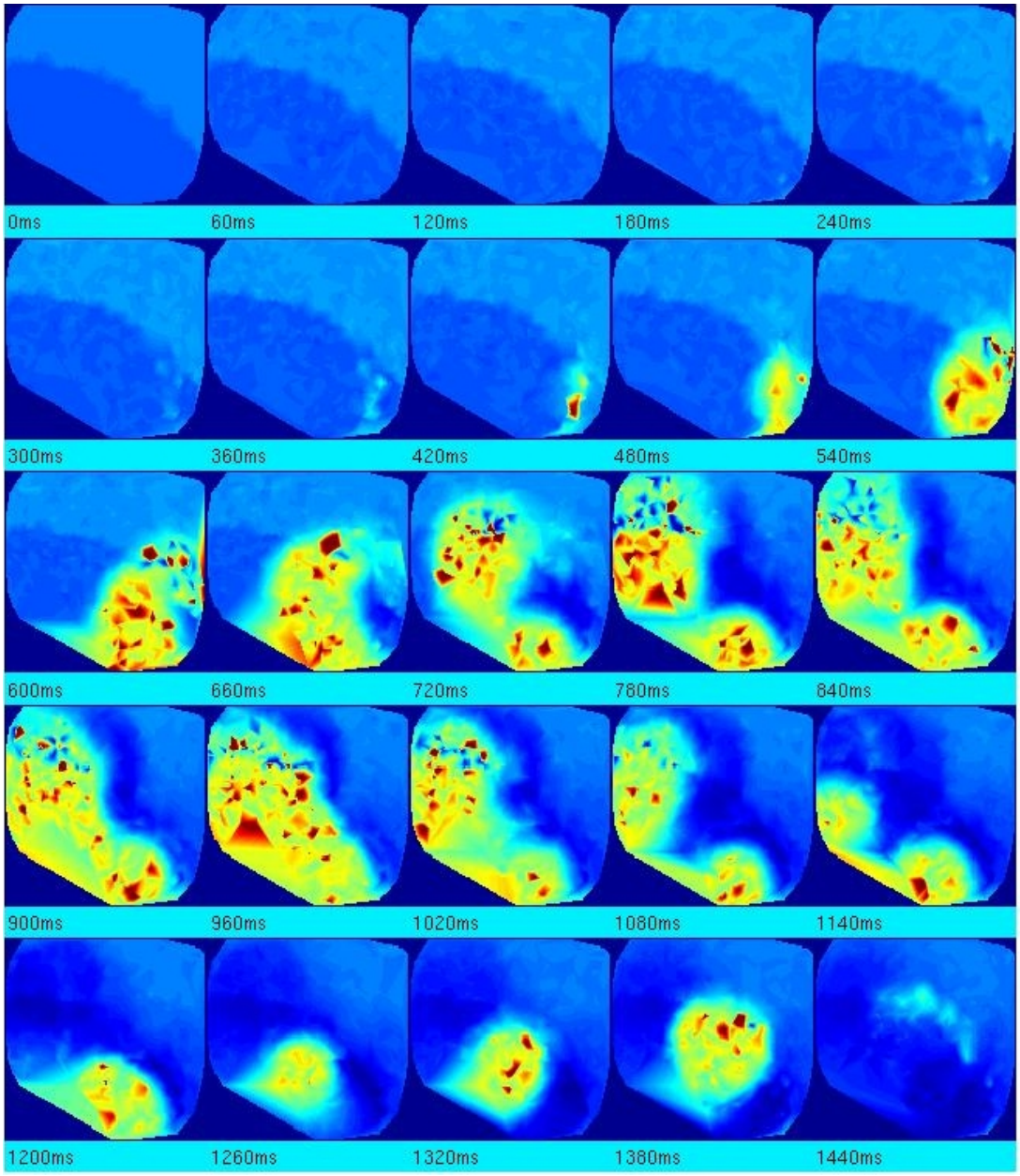


Figure 5.13: Noised FVSM Visual Cortex output for input movie 300°

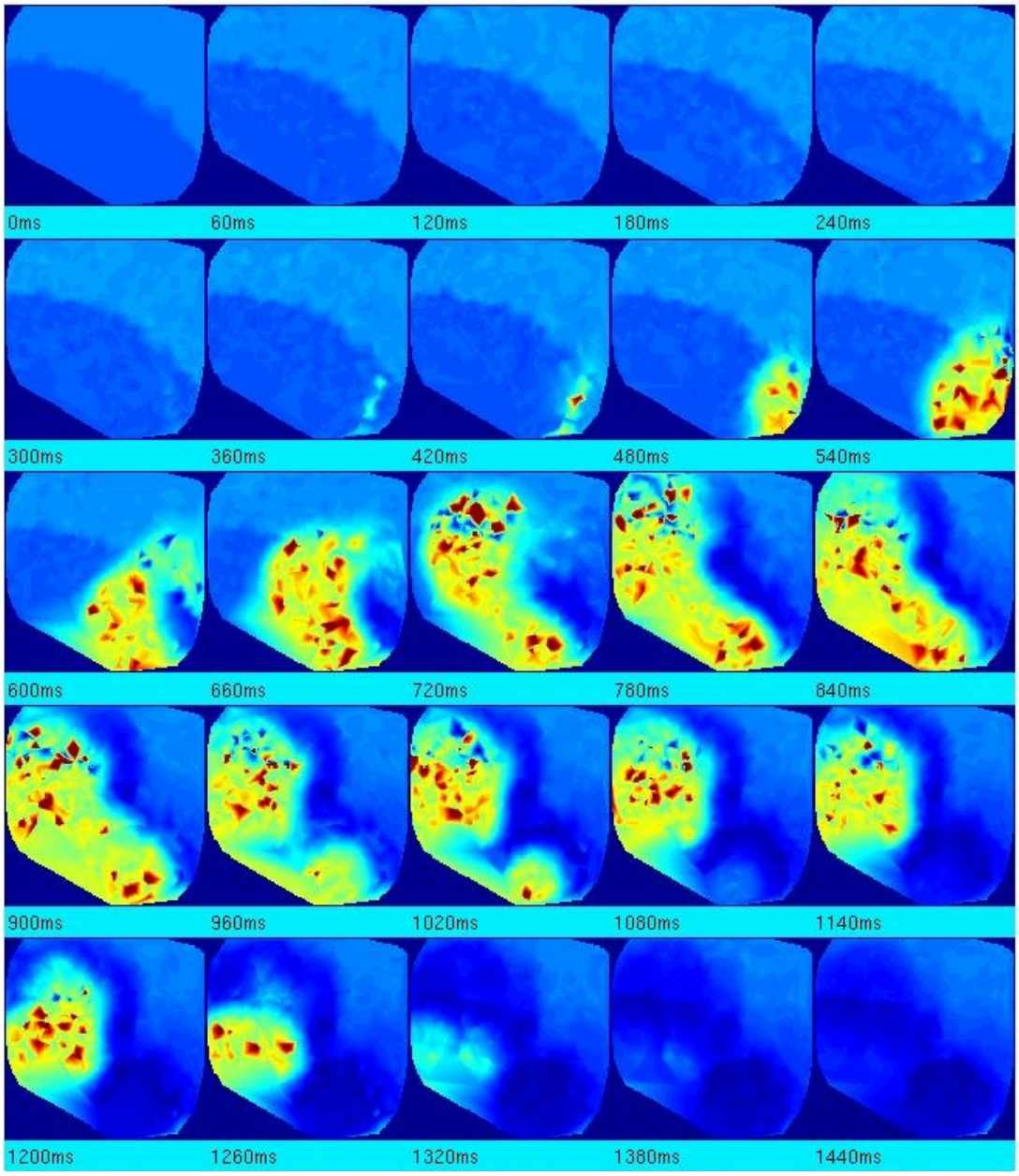
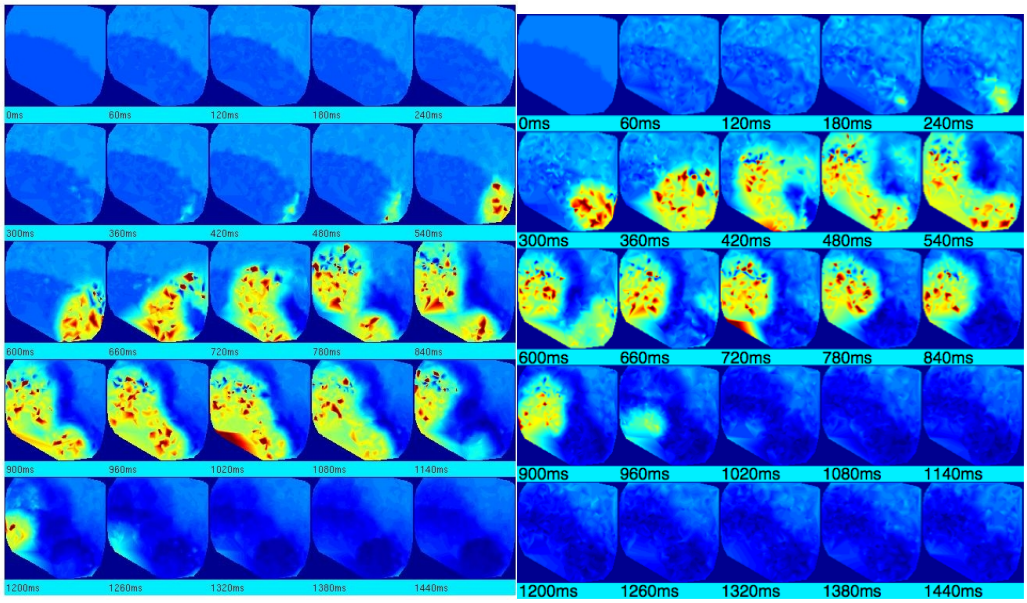


Figure 5.14: Noised FVSM Visual Cortex output for input movie 330°

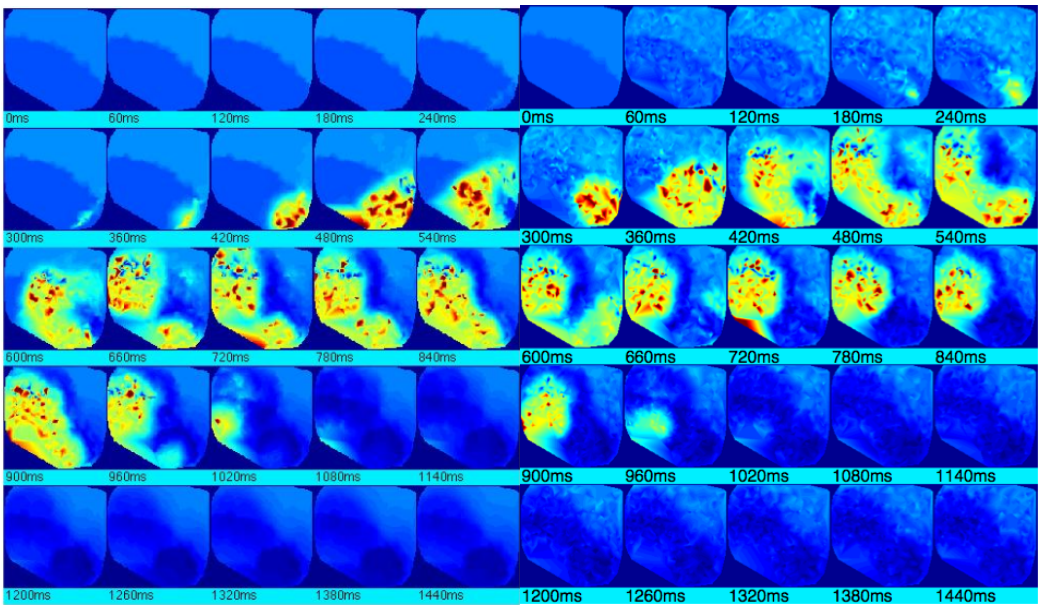
Consider the following comparison between the 0° movie output from the noised FVSM and the combined Retina/VC model used in [12]. In Figure 5.15, it is clear the FVSM cortical wave a) is delayed by about 100ms in comparison to the combined Retina/VC model wave b). The cause of this behavior is the inhibition layer in the FVSM LGC sub-model. This is illustrated by the deactivation of the inhibitory layer and removal of the Gaussian noise block in the retina.



(a) (b)

Figure 5.15: Comparison between a) the Noised FVSM cortical response to 0° input and b) response of the Retina/VC in [12] to 0° input.

Figure 5.16 b) shows this noiseless FVSM model result. Note its similarity to that shown in a), which is the cortex wave from the combined Retina/VC. The results are not identical, but this is no surprise since the geometric construction of the combined Retina/VC is different from the FVSM.



(a) (b)

Figure 5.16: Comparison between a) the Noiseless FVSM cortical response to 0° input and b) response of the Retina/VC model in [12] to 0° input. Notice the cortex noise in d).

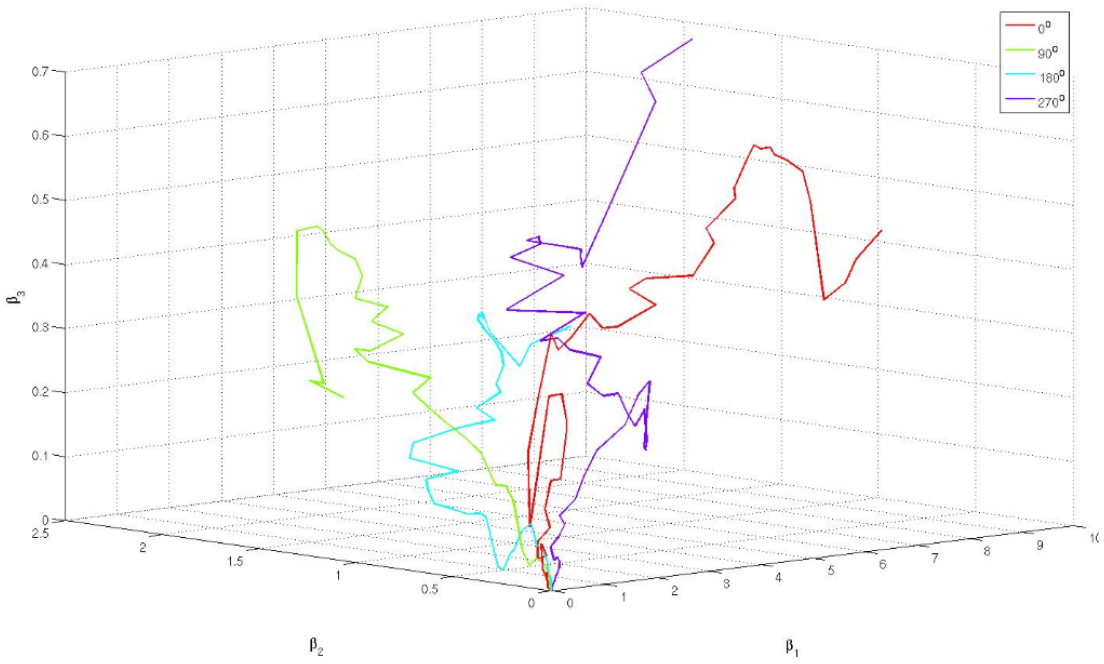
5.3 Angle Separability Tests

As was noted in the previous section, the cortical waves vary from simulation to simulation based on the noise components of the model and the angle movie input used. Thus, to get a better approximation of the cortical response to the movie inputs without noise influence, 60 iterations of each angle input were executed. Each simulation iteration used the GENESIS "randseed" command to prime the random number generators used in the FVSM. Upon completion of the simulations for each angle set, the GENESIS data was archived.

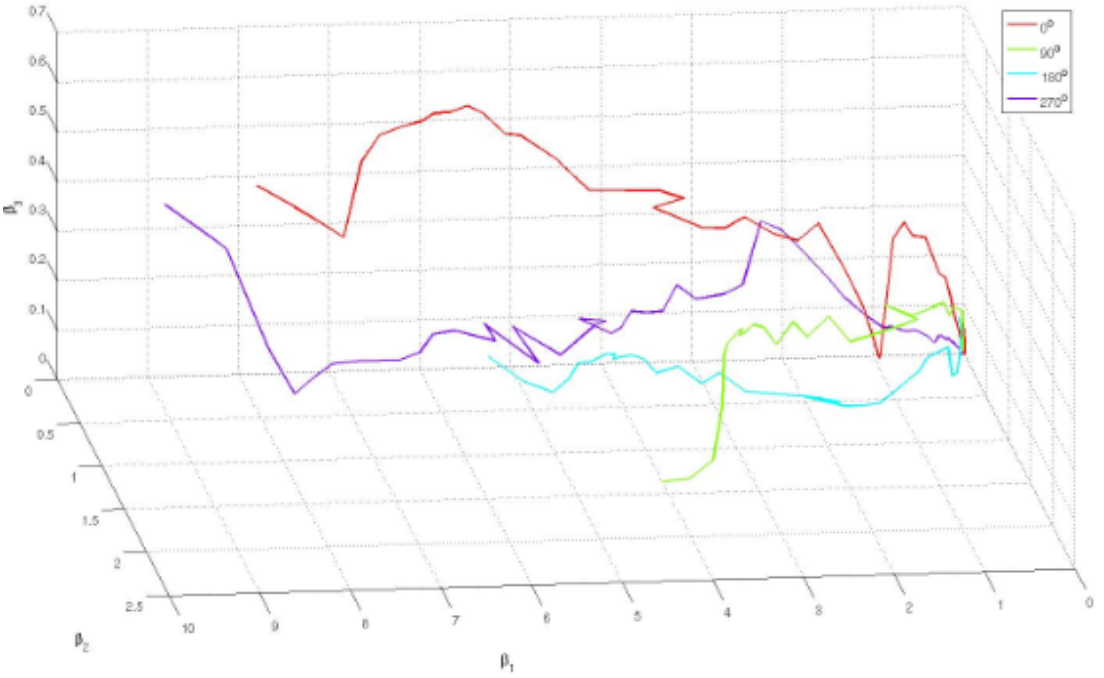
Data analysis of the neural activity in the cortex was done using MATLAB codes, provided in the appendix. First, each data set was filtered using *LPF.m*. Next, the resulting filtered data was put through Principal Component Analysis. The PCA code is also provided in the appendix, labeled as *PCA_calculation.m*.

The following diagrams illustrate the relationships among the PCA β components for each of the 12 test angles. More details about the process of PCA can be

found in [5], [12], and [16]. These three-dimensional plots are with respect to the β_1, β_2, and β_3 components. In our studies, the first six β components were computed, however according to work by Du, Ghosh, and Ulinski in [9], the first several of these components has proved to be sufficient for information extraction from the simulation runs. As a result, the same procedure is replicated here. This approach is further validated by its prior successful use in the aforementioned visual cortex model, a component of the FVSM.

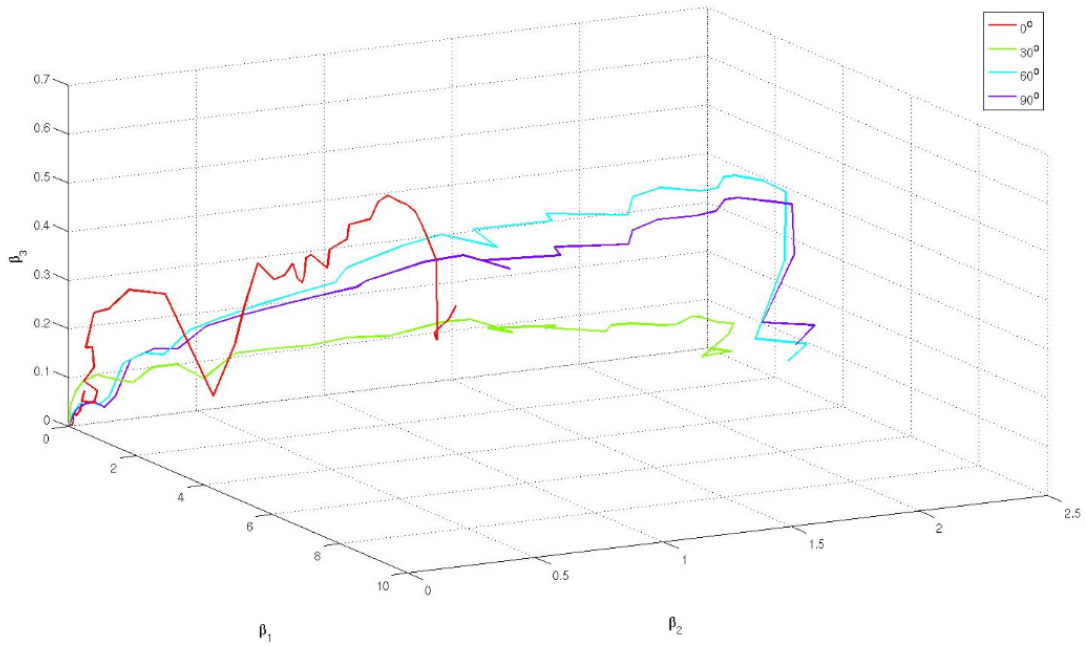


(a)

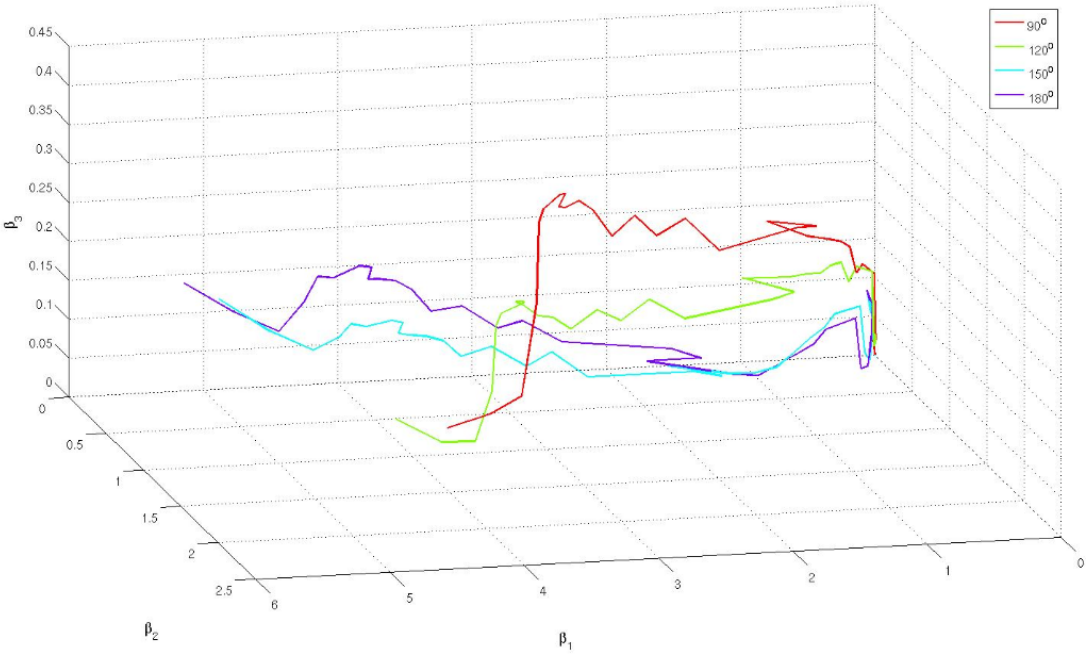


(b)

Figure 5.17: β components from PCA for the cardinal angle inputs.

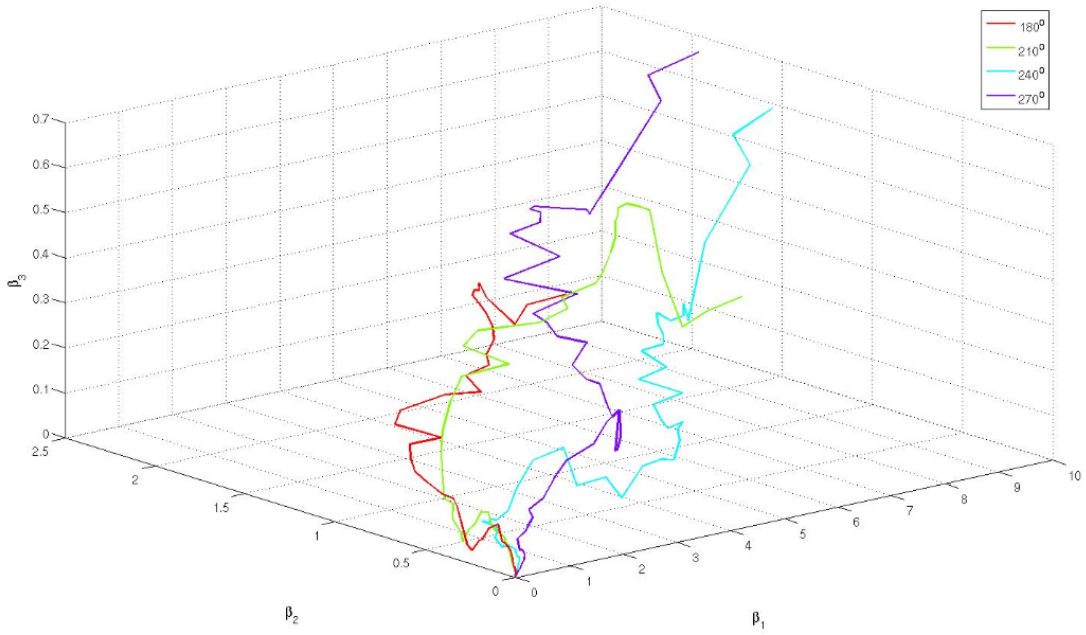


(a)

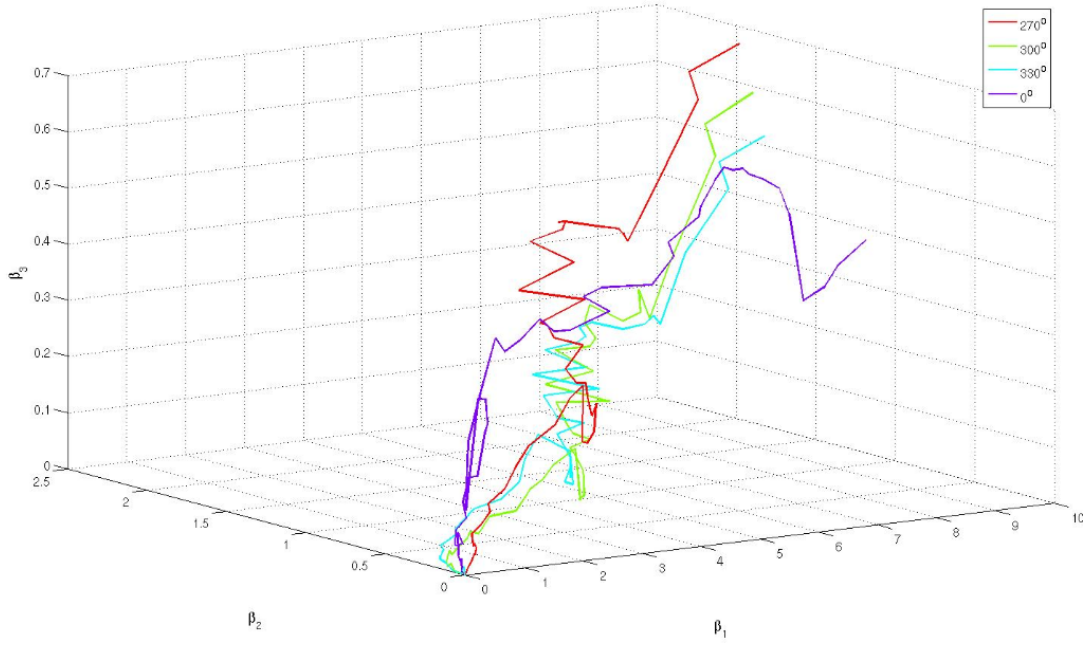


(b)

Figure 5.18: a) β components of angles 0°, 30°, 60°, 90°. b) Beta components of angles 90°, 120°, 150°, 180°



(a)



(b)

Figure 5.19: a) β components of angles 180°, 210°, 240°, 270°. b) Beta components of angles 270°, 300^0, 330°, 0°

CHAPTER VI
DISCUSSION AND CLOSING REMARKS

In this thesis, a novel model of the turtle lateral geniculate complex was constructed. This new computer model was then connected with models of the turtle visual cortex and retina to produce a fused visual system model of the turtle visual pathways. This model work was inspired and guided by former efforts at the Center for Bio-Cybernetics and Intelligent Systems both at Washington University in St. Louis and Texas Tech University in Lubbock.

The contribution of this work to the area of neuroscience is the use of inhibition to counteract Gaussian white noise in the turtle retina. Prior works in this area have attested to the noisy nature of retinal ganglion cells, and a method to address this problem in our combined models of the turtle retina and visual cortex was not available before undertaking this study. Results have confirmed the ability of neuropile inhibitory signals, combined with a correspondingly large radius of influence between the retina and neuropile cells, to suppress noise of a given variance in the retina. Biologically speaking, we know of no direct evidence to suggest the turtle neuropile cells actually provide such noise cancellation in the visual system, however studies such as these promise to open the door to further understanding of the interaction among the turtle visual system components.

Further studies are in progress regarding this work. Of keen desire is to improve angle separability as well as enhance the model's ability to filter noise at a wider range of variances. In studies by Dr. Ekanayake, additional movie inputs with differing speeds were investigated with respect to the retina. These datasets are currently being analyzed using the FVSM discussed in these pages.

Concluding this work, the study results presented herein seem to encourage further investigation of inhibition in the lateral geniculate complex as a tool for noise reduction in visual signals from the retina.

BIBLIOGRAPHY

[1] Rainey, W.T. and Ulinski, P. S., Morphology of Neurons in the Dorsal Lateral Geniculate Complex in Turtles of the Genera *Psuedemys* and *Chrysemys. Journal of Comparative Neurology*, 253: p440-465, 1986

[2] Anderson, Ronald C., Ghosh, B.K., and Ekanayake, M.P.B., , *Conference*, Munich, Germany, Sept 2011

[3] Ulinski, P.S., Unpublished Correspondence, Chicago, IL, 2008-2010

[4] Nenadic, Z., Ghosh, B.K., Ulinski, P.S., Propagating Waves in Visual Cortex: A Large-Scale Model of the Visual Cortex, *Journal of Computational Neuroscience*, vol 14, p.161-184, 2003

[5] Ekanayake, Parakrama, *Decoding the Speed and Motion Direction of Moving Targets Using a Turtle Retinal Patch Model*, PhD Dissertation, Dept. of Mathematics and Statistics, Texas Tech University, Lubbock, TX, 2011

[6] Cosans, Christopher E. and Ulinski, P.S., Spatial Organization of Axons in Turtle Visual Cortex: Intralamellar and Interlamellar Projections, *Journal of Comparative Neurology*, 296: p548-558, 1990

[7] Mulligan, K.A, and Ulinski, P.S., Orgainization of Geniculocortical Projections in Turtles: Isoazimuth Lamellae in the Visual Cortex, *Journal of Comparative Neurology*, 296:531-547, 1990

[8] Nenadic, Z., Ghosh, B.K., Signal Processing and Control Problems in the Brain: A Neuroscientific Perspective, *IEEE Control Systems Magazine*, Vol 21, no 4, p.28-41, Aug 2001

[9] Du, X., Ghosh, B.K., and Ulinski, P.S., Encoding and Decoding Target Locations with Waves in the Turtle Visual Cortex, *IEEE Transactions on Biomedical Engineering*, Vol 52, no 4, p566-577, 2005

[10] Ekanayake, M.P.B., Ghosh, B.K., and Ulinski, P.S., Motion Encoding and Decoding in the Turtle Retina, *10th European Control Conference*, Budapest, Hungary, August 2009

[11] Joseph, Jenner J., Bio-Inspired Encoding of Images with Spatiotemporal Cortical Activity Waves and Information Recovery Via Dynamical Modeling, PhD Dissertation, Washington University, St. Louis, MO, 2006

[12] Perera, Neshadha A.P., *Target Motion Discrimination with Model Retina and Cortex*, MS Thesis, Dept. of Mathematics and Statistics, Texas Tech University, Lubbock, TX, 2011

[13] Wang, Wenxue, Dynamics of the Turtle Visual Cortex and Design of Sensor Netowrks, PhD Dissertation, Washington University, St. Louis, MO, 2006

[14] Bower, J. M. and Beeman, D.. *The Book Of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* Springer Telos, 2nd Edition, 1998

[15] Kandel, Eric, Schwartz, James, and Jessell, Thomas, *Principles of Neural Science, 4th Edition*, McGraw-Hill Health Professions Division, New York, 1991

[16] I. T. Jolliffe, *Principal Component Analysis, 2nd Edition*, Springer Verlag, New York, NY, 2002

[17] The MathWorks. *MATLAB*, 2011 [Software]

[18] *The GEneral NEural SImulation System v2.3*, 1998 [Software]

[19] Purves D., Augustine GJ, Fitzpatric D, et. al., *Neuroscience, 2nd Edition*, Sinauer Associates, Sunderland, MA, 2001

[20] Mangles, Jennifer, Psychology Course Website *W1010x: Mind, Brain and Behavior*, Columbia University, 2003

[21] Nenadic, Zoran, Ghosh, Bijoy, Ulinski, Philip S., Modeling and Estimation Problems in the Turtle Visual Cortex, IEEE Transactions on BioMedical Engineering, Vol 49, No 8, Aug 2002

[22] National Institutes of Health website: http://www.ncbi.nlm.nih.gov/books/NBK11164/

APPENDIX

This appendix collects some of the major GENESIS and MATLAB scripts used in the construction and testing of the Fused Visual System Model, discussed in the main chapters of this document. All of these codes are referenced in the body of the thesis, but are considered too long to fit conveniently within the respective chapters. For each program, a brief description is provided and comments are included within the verbatim listing wherever appropriate.

The GENESIS scripts listed below are all components of the Fused Visual System Model, even though only one entry below bears that name. As can be seen in the codes, many "includes" are specified, which link scripts together to create the FVSM that underlies all the discussion in Chapters IV and V. Some of these includes are not provided since they were never modified from previous model work.

The MATLAB scripts have two purposes. First, MATLAB is used to assign locations for every cell in the FVSM. Second, MATLAB code is also instrumental in analyzing the resulting data from the Visual Cortex after testing of the FVSM. The first of these purposes is accomplished by *Generate LGC Coords*, which takes a matrix of cell densities and creates a random distribution of (x, y) coordinate pairs loosely satisfying the provided densities. This code creates the cellular layouts for both the LGC Cell Plate and LGC Neuropile Layers, as discussed in Sections 3.4.1 and 3.4.2. Although not explicitly discussed in this thesis, the coordinates used for the retina model and visual cortex model cells are developed using similar (though not identical) code. Analysis of the FVSM data, discussed in Chapter II, is accomplished by application of the remaining MATLAB codes provided in this appendix.

LGC Neuron

The GENESIS script below is discussed at length in Chapter III and is responsible for the construction of the cellular-level models for both the Cell Plate and Neuropile neuron classes of the LGC. Also included in the code are the parameters found in Tables 3.1, 3.2, and 3.3 back in Chapter II.

```
//*****************************************
// HEADER AND FILE INCLUDES
//*****************************************
include ../Library/phil_channel.g
//***************** END ********************
//*****************************************
//  CONSTAND DECLARATIONS AND PARAMETERS
//*****************************************
//-----------------------------
// CONSTANT DECLARATIONS
//-----------------------------
float PI = 3.14159
//-----------------------------
//-----------------------------
// CELLULAR ELECTRICAL PROPERTIES
//-----------------------------
float RM = 2.45 //3.450 //specific membrane RESISTANCE (ohms m^2)
float CM = .02429 //.00429 //specific membrane CAPACITANNCE
     (farads/m^2)
float RA = 1.389 //2.389 //specific AXIAL resistance (ohms m)
float EREST = -57.0 * 1e-3 //RESTING membrane potential (volts)
float Eleak = -57.0 * 1e-3 //membrane LEAKAGE potential (volts)
float Gleak = 0.027  //maximum LEAKAGE conductance (S/m^2)
float ENa = .045 //SODIUM reversal potential (volts)
float GNa = 600.00 * 1e-3/1e-4 //maximum SODIUM conductance
     (S/m^2)
```

```
float EK = -.90 //POTASSIUM reversal potential (volts)
float GK = 30.00 * 1e-3/1e-4 //maximum POTASSIUM conductance
     (S/m^2)
//-----------------------------
//-----------------------------
// COMPARTMENT DIMENSION DECLARATIONS
//-----------------------------
// Taken from Phil Ulinski's suggestions
// Soma is spherical
float soma_d = 15 * 1e-6
// All dendrites are cylindrical
// DENDRITE LEVEL 1
float dend1_l = 20 * 1e-6
float dend1_d = 5 * 1e-6
// DENDRITE LEVEL 2
float dend2_l = 100 * 1e-6
float dend2_d = 2 * 1e-6
float dend3_l = dend2_l
float dend3_d = dend2_d
// DENDRITE LEVEL 3 (terminal ends of dendrite tree)
float dend4_l = 50 * 1e-6
float dend4_d = 1 * 1e-6
float dend5_l = dend4_l
float dend5_d = dend4_d
float dend6_l = dend4_l
float dend6_d = dend4_d
float dend7_l = dend4_l
float dend7_d = dend4_d
//-----------------------------
//-----------------------------
// SURFACE AREA COMPUTATIONS
//-----------------------------
```

```
float soma_A = PI * (soma_d*soma_d-dend1_d*dend1_d/4)
float dend_level1_A = PI*dend1_d*dend1_l
float dend_level2_A = PI*dend2_d*dend2_l
float dend_level3_A = PI*dend4_d*dend4_l+PI*dend4_d*dend4_d/4
//TERMINAL END OF DENDRITE TREE
//---------------------------
//**************** END *********************

//*****************************************
// FUNCTIONS FOR LGC NEURON CREATION
//*****************************************
// See "includes" at file beginning for other functions
function link_compartments(receiver, sender)
// Link passed compartments together
// Signal travels " SENDER -> RECEIVER "
str receiver, sender
addmsg {sender} {receiver} RAXIAL Ra previous_state
addmsg {receiver} {sender} AXIAL previous_state
end
function construct_sphere_compartment(path, dia, area)
// Make the LGC soma with passed specifications
// Use Phil's Na/K Channels
str path
float dia, area
create compartment {path}
setfield   {path} \
Em {Eleak} \
Cm {CM*area} \
Rm {RM/area} \
Ra {RA/(dia*PI)}
end
function construct_cylind_compartment(path, len, dia, area)
```

```
// Make the LGC dendrite with specifications passed
// No Channels are used here
str path
float len, dia, area
create compartment {path}
setfield   {path} \
Em {Eleak} \
Rm {RM/area} \
Ra {RA*len/(dia*dia*PI)} \
Cm {CM*area}
end
function make_dendrite(path, len, dia, area)
str path
float len, dia, area
construct_cylind_compartment {path} {len} {dia} {area}
end
function make_soma(path, dia, area)
str path
float dia, area
construct_sphere_compartment {path} {dia} {area}
pushe {path}
make_potassium_channel {GK} {EK} {area}
pope
addmsg {path}/potassium_channel {path} CHANNEL Gk Ek
addmsg {path} {path}/potassium_channel VOLTAGE Vm
pushe {path}
make_sodium_channel {GNa} {ENa} {area}
pope
addmsg {path}/sodium_channel {path} CHANNEL Gk Ek
addmsg {path} {path}/sodium_channel VOLTAGE Vm
end
function make_lgc_neuron(parent, soma_only)
```

```
str parent
int soma_only
float r1, r2
float set_value
// Construct the LGN soma
make_soma {parent}/soma {soma_d} {soma_A}
if({soma_only} == 0)
// Construct the LGN dendrites
make_dendrite {parent}/dend1 {dend1_l} {dend1_d} {dend_level1_A}
make_dendrite {parent}/dend2 {dend2_l} {dend2_d} {dend_level2_A}
make_dendrite {parent}/dend3 {dend3_l} {dend3_d} {dend_level2_A}
make_dendrite {parent}/dend4 {dend4_l} {dend4_d} {dend_level3_A}
make_dendrite {parent}/dend5 {dend5_l} {dend5_d} {dend_level3_A}
make_dendrite {parent}/dend6 {dend6_l} {dend6_d} {dend_level3_A}
make_dendrite {parent}/dend7 {dend7_l} {dend7_d} {dend_level3_A}
// Ra Values for each link
r1 = {getfield {parent}/soma Ra}
r2 = {getfield {parent}/dend1 Ra}
set_value = (r1+r2)/2
setfield {parent}/dend1 Ra {set_value} // Soma Level
r1 = {getfield {parent}/dend2 Ra}
r2 = {getfield {parent}/dend1 Ra}
set_value = (r1+r2)/2
setfield {parent}/dend2 Ra {set_value} // Level 1
r1 = {getfield {parent}/dend3 Ra}
r2 = {getfield {parent}/dend1 Ra}
set_value = (r1+r2)/2
setfield {parent}/dend3 Ra {set_value} // Level 1
r1 = {getfield {parent}/dend2 Ra}
r2 = {getfield {parent}/dend4 Ra}
set_value =(r1+r2)/2
setfield {parent}/dend4 Ra {set_value} // Level 2
```

```
r1 = {getfield {parent}/dend2 Ra}
r2 = {getfield {parent}/dend5 Ra}
set_value = (r1+r2)/2
setfield {parent}/dend5 Ra {set_value} // Level 2
r1 = {getfield {parent}/dend3 Ra}
r2 = {getfield {parent}/dend6 Ra}
set_value = (r1+r2)/2
setfield {parent}/dend6 Ra {set_value} // Level 2
r1 = {getfield {parent}/dend3 Ra}
r2 = {getfield {parent}/dend7 Ra}
set_value = (r1+r2)/2
setfield {parent}/dend7 Ra {set_value} // Level 2
// Link compartments into LGN structure
link_compartments {parent}/soma {parent}/dend1  //Soma Level
link_compartments {parent}/dend1 {parent}/dend2 //Level 1
link_compartments {parent}/dend1 {parent}/dend3 //Level 1
link_compartments {parent}/dend2 {parent}/dend4 //Level 2
link_compartments {parent}/dend2 {parent}/dend5 //Level 2
link_compartments {parent}/dend3 {parent}/dend6 //Level 2
link_compartments {parent}/dend3 {parent}/dend7 //Level 2
end
end
//**************** END *********************
function make_pulsegen(pulse_gen_path, level, width, delay)
str pulse_gen_path
float level, width, delay
create pulsegen {pulse_gen_path}
setfield {pulse_gen_path} \
level1 {level} \
delay1 {delay} \
width1 {width}
end
```

```
function update_run
step {1.0} -time -background
end
```

Phil Channel

This GENESIS script produces the ionic channels used by the cellular models in the Retina Model, Visual Cortex Model, and LGC Model. It is not described in the main thesis, but is nonetheless vital to the successful execution of the FVSM.

```
//================= CONSTANTS ======================
int EXPONENTIAL =   1
int SIGMOID     =   2
int LINOID      =   3

//================= ACTIVE CHANNELS ================
//=========== By Phil S. Ulinski and Peiper ================
//==================================================

// Fast Sodium (Na)
function make_sodium_channel(Gmax, RevPot, SOMA_A)
float SOMA_A // the area of compartment m^2
float Gmax, RevPot
if ({exists sodium_channel})
return
end
create hh_channel sodium_channel
setfield  sodium_channel\
Ek  {RevPot} \ // V
Gbar { Gmax * SOMA_A } \ // S
Xpower 3.0 \
Ypower 1.0 \
X_alpha_FORM {LINOID} \
X_alpha_A -0.32e6 \ // 1/V-sec
X_alpha_B -4.00e-3 \ // V
X_alpha_V0 -34.67e-3  \ // V
X_beta_FORM {LINOID} \
```

```
X_beta_A 0.28e6 \ // 1/sec
X_beta_B 5.00e-3 \ // V
X_beta_V0 -6.67e-3 \ // V
Y_alpha_FORM {EXPONENTIAL} \
Y_alpha_A 0.128e3 \ // 1/sec
Y_alpha_B -18.00e-3 \ // V
Y_alpha_V0 -34.00e-3 \ // V
Y_beta_FORM {SIGMOID} \
Y_beta_A 4.00e3 \ // 1/sec
Y_beta_B -5.00e-3 \ // V
Y_beta_V0 -11.00e-3  // V
end

// Dealyed Rectified Potassium (K)
function make_potassium_channel(Gmax, RevPot, SOMA_A)
float SOMA_A // the area of compartment m^2
float Gmax, RevPot
if ({exists potassium_channel})
return
end

create hh_channel potassium_channel
     setfield potassium_channel\
Ek  {RevPot} \ // V
Gbar { Gmax * SOMA_A } \ // S
Xpower 4.0 \
Ypower 0.0 \
X_alpha_FORM {LINOID} \
X_alpha_A -0.032e6 \ // 1/V-sec
X_alpha_B -3.5e-3 \ // V
X_alpha_V0 -36.00e-3 \ // V
X_beta_FORM {EXPONENTIAL} \
```

```
X_beta_A 0.50e3 \ // 1/sec
X_beta_B -40.0e-3 \ // V
X_beta_V0 -41.00e-3   // V
end
```

The Fused Visual System Model

This is the main GENESIS script that simulates the FVSM, discussed and tested at length in this thesis. It is the noised version of the FVSM that is presented here. Note that some of the code lines are too long to fit on a single line. Such lines have been split to two lines with a five space indent on the second and successive splits.

```
echo {"---> Retina model by MPB Ekanayake"}
echo {"---> LGC model by Ronald C. Anderson"}
echo {"---> Visual Cortex model by ZORAN"}
echo {"*******************************************************"}
//==============================
// INCLUDING LIBRARY FUNCTIONS
//==============================
//====== FROM THE NGU MODEL=====
//==============================
include ../Library/make_synapse.g
include ../Library/make_spike.g
include ../Library/make_nmda.g
include ../Library/coord_reader.g
include ../Library/make_Vmgraph.g
include ../Library/make_control.g
include ../Library/gauss.g
include ../Library/noise.g
//==============================
//==== FROM THE RETINA MODEL ====
//==============================
// MPB Ekanayake
include ../Library/noise_source.g
include ../Library/retina_coords.g
//==============================
//======= LGN MODEL FILES =======
//==============================
```

```
// Ronald C. Anderson
include ../Library/LGC/lgc_neuron.g
//================================
//= END INITIAL INCLUDE FILE LIST =
//================================
echo {"================================================="}
echo {"STAGE 0::: SETTING PARAMETERS"}
echo {"================================================="}
int TRIAL = 0; // SEE  BASH SCRIPT FOR MAX (LIKELY 60)
int INPUT_ANGLE = 180; // Assume angle zero in case no args provided.
int FLAG = 0;  // Will become 1 if multiple trials assumed
int NO_ARGS = {argc} // Arguments passed to GENESIS if exist
if ({NO_ARGS} >= 1)
INPUT_ANGLE = {argv 1}
if ({NO_ARGS} >= 2)
TRIAL = {argv 2}
FLAG = 1  // Assume running multiple trials
if ({NO_ARGS} > 2)
echo {"Too many arguments specified.."}
end
end
else
echo {"No arguments passed"}
end
echo {"---> Input Angle: "@INPUT_ANGLE}
echo {"---> Trial Number: "@TRIAL}
if ({FLAG} == 1)
sh mkdir {"./my_results"}
sh mkdir {"./my_results/"@INPUT_ANGLE}
sh mkdir {"./my_results/"@INPUT_ANGLE@"/"@TRIAL}
echo {"Results will be saved to the batch directories."}
else
```

```
echo {"Results will be saved to the script directory."}
end
float var = 1.5
float variance = 4e-10
float sat = 3*variance
float retina_noiselevel = 3e-11 //10e-10  // 4e-10
//============ SYNAPSES FROM LGN - PARAMETERS ===========//
float lgn_lat_w = 1.87 //
float lgn_med_w = 0.25//
float lgn_stel_w = 0.25//0.21
float var_lat_r = 0.026 //mm
float var_med_r = 0.06   //mm
float var_stel_r = 0.025 //mm
//=========== OTHER SYNAPSES ============================//
float lat_lat_w = 0.95 * var
float lat_med_w = 1.20 * var
float lat_stel_w = 0.01 * var
float lat_hor_w = 0.02 * var
float med_lat_w = 0.3 * var
float med_med_w = 0.40 * var
float med_stel_w = 0.01 * var
float med_hor_w = 0.02 * var
float stel_lat_w = 1.9//1.89
float stel_med_w = 1.37//1.38
float stel_stel_w = 0.1
float hor_lat_w = 7.6//7.0
float hor_med_w = 5.5//5.7
float lat_lat_r = 0.250 //mm
float lat_med_r={lat_lat_r}
float lat_stel_r={lat_lat_r}
float lat_hor_r=0.25
float med_lat_r = 0.250 //mm
```

```
float med_med_r={med_lat_r}
float med_stel_r={med_lat_r}
float med_hor_r={med_stel_r}
float stel_lat_r = 0.35 //mm
float stel_med_r=0.35
float stel_stel_r={stel_med_r}
float hor_lat_r = 0.35 //mm
float hor_med_r = 0.35
//=========== SYNAPTIC PARAMETERS DEFINITIONS ============//
float Gs=5e-9
float E2_i_lgc = -55e-3
//AMPA2    Excitatory
float t2 = 3.0e-3
float tt2 = 0.3e-3
float E2 = 0.0e-3
//NMDA   Inhibitory
float tn = 80.0e-3
float ttn = 0.67e-3
float En = 0.0
//GABAa    Inhibitory
float ta = 1.7e-3
float tta = 1.7e-3
float Ia = -70.0e-3
//GABAb    Inhibitory
float tb = 500e-3
float ttb = 500e-3
float Ib = -90e-3
//Counting Variables
int i
int j
echo {"STAGE 0::: --- COMPLETED ---"}
```

```
//=====================================================//
//============================ RETINA ==================//
//=====================================================//
//==================== CREATING RETINAL CELLS ============//
include ../Library/retina_ganglion.g
echo {"================================================="}
echo {"STAGE 1::: CREATING MPB EKANAYAKE RETINA"}
echo {"================================================="}
create neutral /retina
create neutral /retina_noise
useclock /retina_noise/ {1}
create neutral /retina_noise_diff_amp
useclock /retina_noise_diff_amp/ {1}
create neutral /retina_noise/A_ON
useclock /retina_noise/A_ON {1}
create neutral /retina_noise/A_OFF
useclock /retina_noise/A_OFF {1}
create neutral /retina_noise/B_1
useclock /retina_noise/B_1 {1}
create neutral /retina_noise/B_2
useclock /retina_noise/B_2 {1}
create neutral /retina_noise/B_3
useclock /retina_noise/B_3 {1}
create neutral /retina_noise_diff_amp/A_ON
useclock /retina_noise_diff_amp/A_ON {1}
create neutral /retina_noise_diff_amp/A_OFF
useclock /retina_noise_diff_amp/A_OFF {1}
create neutral /retina_noise_diff_amp/B_1
useclock /retina_noise_diff_amp/B_1 {1}
create neutral /retina_noise_diff_amp/B_2
useclock /retina_noise_diff_amp/B_2 {1}
create neutral /retina_noise_diff_amp/B_3
```

```
useclock /retina_noise_diff_amp/B_3 {1}
openfile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat r
float ACTIVE_A_ON_CELL_COUNT  =
     {readfile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat}
float ACTIVE_A_OFF_CELL_COUNT =
     {readfile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat}
float ACTIVE_B_1_CELL_COUNT   =
     {readfile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat}
float ACTIVE_B_2_CELL_COUNT   =
     {readfile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat}
float ACTIVE_B_3_CELL_COUNT   =
     {readfile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat}
closefile ./data/RETINA_COORDS/EFFECTIVE_CELL_COUNT.dat
echo {""}
echo {"---> A_ON Cell Count: "@ACTIVE_A_ON_CELL_COUNT}
echo {"---> A_OFF Cell Count: "@ACTIVE_A_OFF_CELL_COUNT}
echo {"---> B_1 Cell Count: "@ACTIVE_B_1_CELL_COUNT}
echo {"---> B_2 Cell Count: "@ACTIVE_B_2_CELL_COUNT}
echo {"---> B_3 Cell Count: "@ACTIVE_B_3_CELL_COUNT}
echo {"---> Creating Retina A_ON Actives"}
create neutral /retina/A_ON
openfile ./data/RETINA_COORDS/A_ON_CELL_COORDS_ACTIVE.dat r
for (i=1; i<=ACTIVE_A_ON_CELL_COUNT; i=i+1)
create neutral /retina/A_ON/{"cell"@{i}}
makecell_ganglion /retina/A_ON/{"cell"@{i}}
make_spike /retina/A_ON/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
create gnoise /retina_noise/A_ON/{"r_noise"@{i}}
useclock /retina_noise/A_ON/{"r_noise"@{i}} {1}
create diffamp /retina_noise_diff_amp/A_ON/{"r_diff"@{i}}
useclock /retina_noise_diff_amp/A_ON/{"r_diff"@{i}} {1}
setfield /retina_noise_diff_amp/A_ON/{"r_diff"@{i}} gain
     {retina_noiselevel} saturation {{retina_noiselevel}*3}
```

```
addmsg /retina_noise/A_ON/{"r_noise"
     @{i}} /retina_noise_diff_amp/A_ON/{"r_diff"@{i}} PLUS output
addmsg /retina_noise_diff_amp/A_ON/{"r_diff"@{i}}
     /retina/A_ON/{"cell"@{i}}/soma INJECT output
assign_reitinal_cell_positions /retina/A_ON/{"cell"@{i}}/soma
     {readfile ./data/RETINA_COORDS/A_ON_CELL_COORDS_ACTIVE.dat}
end
closefile ./data/RETINA_COORDS/A_ON_CELL_COORDS_ACTIVE.dat
echo {"---> Creating Retina A_OFF Actives"}
create neutral /retina/A_OFF
openfile ./data/RETINA_COORDS/A_OFF_CELL_COORDS_ACTIVE.dat r
for (i=1; i<=ACTIVE_A_OFF_CELL_COUNT; i=i+1)
create neutral /retina/A_OFF/{"cell"@{i}}
makecell_ganglion /retina/A_OFF/{"cell"@{i}}
make_spike /retina/A_OFF/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
create gnoise /retina_noise/A_OFF/{"r_noise"@{i}}
useclock /retina_noise/A_OFF/{"r_noise"@{i}} {1}
create diffamp /retina_noise_diff_amp/A_OFF/{"r_diff"@{i}}
useclock /retina_noise_diff_amp/A_OFF/{"r_diff"@{i}} {1}
setfield /retina_noise_diff_amp/A_OFF/{"r_diff"@{i}} gain
     {retina_noiselevel} saturation {{retina_noiselevel}*3}
addmsg /retina_noise/A_OFF/{"r_noise"@{i}}
     /retina_noise_diff_amp/A_OFF/{"r_diff"@{i}} PLUS output
addmsg /retina_noise_diff_amp/A_OFF/{"r_diff"@{i}}
     /retina/A_OFF/{"cell"@{i}}/soma INJECT output
assign_reitinal_cell_positions /retina/A_OFF/{"cell"@{i}}/soma
     {readfile ./data/RETINA_COORDS/A_OFF_CELL_COORDS_ACTIVE.dat}
end
closefile ./data/RETINA_COORDS/A_OFF_CELL_COORDS_ACTIVE.dat
echo {"---> Creating Retina B1 Actives"}
create neutral /retina/B_1
openfile ./data/RETINA_COORDS/B_1_CELL_COORDS_ACTIVE.dat r
```

```
for (i=1; i<=ACTIVE_B_1_CELL_COUNT; i=i+1)
create neutral /retina/B_1/{"cell"@{i}}
makecell_ganglion /retina/B_1/{"cell"@{i}}
make_spike /retina/B_1/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
create gnoise /retina_noise/B_1/{"r_noise"@{i}}
useclock /retina_noise/B_1/{"r_noise"@{i}} {1}
create diffamp /retina_noise_diff_amp/B_1/{"r_diff"@{i}}
useclock /retina_noise_diff_amp/B_1/{"r_diff"@{i}} {1}
setfield /retina_noise_diff_amp/B_1/{"r_diff"@{i}}
     gain {retina_noiselevel} saturation {{retina_noiselevel}*3}
addmsg /retina_noise/B_1/{"r_noise"@{i}}
     /retina_noise_diff_amp/B_1/{"r_diff"@{i}} PLUS output
addmsg /retina_noise_diff_amp/B_1/{"r_diff"@{i}}
     /retina/B_1/{"cell"@{i}}/soma INJECT output
assign_reitinal_cell_positions /retina/B_1/{"cell"@{i}}/soma
     {readfile ./data/RETINA_COORDS/B_1_CELL_COORDS_ACTIVE.dat}
end
closefile ./data/RETINA_COORDS/B_1_CELL_COORDS_ACTIVE.dat
echo {"---> Creating Retina B2 Actives"}
create neutral /retina/B_2
openfile ./data/RETINA_COORDS/B_2_CELL_COORDS_ACTIVE.dat r
for (i=1; i<=ACTIVE_B_2_CELL_COUNT; i=i+1)
create neutral /retina/B_2/{"cell"@{i}}
makecell_ganglion /retina/B_2/{"cell"@{i}}
make_spike /retina/B_2/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
create gnoise /retina_noise/B_2/{"r_noise"@{i}}
useclock /retina_noise/B_2/{"r_noise"@{i}} {1}
create diffamp /retina_noise_diff_amp/B_2/{"r_diff"@{i}}
useclock /retina_noise_diff_amp/B_2/{"r_diff"@{i}} {1}
setfield /retina_noise_diff_amp/B_2/{"r_diff"@{i}}
     gain {retina_noiselevel} saturation {{retina_noiselevel}*3}
addmsg /retina_noise/B_2/{"r_noise"@{i}}
```

```
     /retina_noise_diff_amp/B_2/{"r_diff"@{i}} PLUS output
addmsg /retina_noise_diff_amp/B_2/
     {"r_diff"@{i}} /retina/B_2/{"cell"@{i}}/soma INJECT output
assign_reitinal_cell_positions /retina/B_2/{"cell"@{i}}/soma
     {readfile ./data/RETINA_COORDS/B_2_CELL_COORDS_ACTIVE.dat}
end
closefile ./data/RETINA_COORDS/B_2_CELL_COORDS_ACTIVE.dat
echo {"---> Creating Retina B3 Actives"}
create neutral /retina/B_3
openfile ./data/RETINA_COORDS/B_3_CELL_COORDS_ACTIVE.dat r
for (i=1; i<=ACTIVE_B_3_CELL_COUNT; i=i+1)
create neutral /retina/B_3/{"cell"@{i}}
makecell_ganglion /retina/B_3/{"cell"@{i}}
make_spike /retina/B_3/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
create gnoise /retina_noise/B_3/{"r_noise"@{i}}
useclock /retina_noise/B_3/{"r_noise"@{i}} {1}
create diffamp /retina_noise_diff_amp/B_3/{"r_diff"@{i}}
useclock /retina_noise_diff_amp/B_3/{"r_diff"@{i}} {1}
setfield /retina_noise_diff_amp/B_3/{"r_diff"@{i}}
     gain {retina_noiselevel} saturation {{retina_noiselevel}*3}
addmsg /retina_noise/B_3/{"r_noise"@{i}}
     /retina_noise_diff_amp/B_3/{"r_diff"@{i}} PLUS output
addmsg /retina_noise_diff_amp/B_3/{"r_diff"@{i}}
     /retina/B_3/{"cell"@{i}}/soma INJECT output
assign_reitinal_cell_positions /retina/B_3/{"cell"@{i}}/soma
     {readfile ./data/RETINA_COORDS/B_3_CELL_COORDS_ACTIVE.dat}
end
closefile ./data/RETINA_COORDS/B_3_CELL_COORDS_ACTIVE.dat
echo {"---> NOTICE: NOT Creating A_ON/A_OFF/B1/B2/B3 Inactives"}
echo {"---> Retina Created"}
echo {"STAGE 1::: --- COMPLETED ---"}
```

```
//=========================================================//
//======================== VISUAL CORTEX ==================//
//=========================================================//
echo {" "}
echo {" "}
echo {"=================================================="}
echo {"STAGE 2::: CREATING LGC AND VISUAL CORTEX CELLS"}
echo {"=================================================="}
//============ GETTING THE NUMBER OF INDIVIDUAL CELLS =======//
echo {"---> Setting # of Cells for LGC and Visual Cortex Models"}
str string, str1, str2, str3, str4, str5, str6, str7
openfile {"./data/No_cells1.dat"} r
string={readfile {"./data/No_cells1.dat"} -l}
str1={substring {string} 1  16}
str2={substring {string} 17  32}
str3={substring {string} 33  48}
str4={substring {string} 49  64}
str5={substring {string} 65  80}
str6={substring {string} 81  96}
str7={substring {string} 97  111}
float No_lgc={str6}
float No_lgn={str1}
float No_lateral={str2}
float No_medial={str3}
float No_stellate={str4}
float No_horizontal={str5}
float No_lgc_i={str7}
echo {"---> # of Cells Set"}
//=============== CREATING LGC CELLS =================//
include ../Library/geniculate.g
echo {"---> Creating Rinny LGC Cell Plates"}
create neutral /network_lgc
```

```
coord_reader ./data/lgn_coords.dat /network_lgc {No_lgc}
for (i=1; i<={No_lgc}; i=i+1)
make_lgc_neuron /network_lgc/{"cell"@{i}} {0}
make_spike /network_lgc/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
end
echo {"---> "@{No_lgc}@" Rinny LGC Cell Plates Created"}
echo {"---> Creating Rinny LGC Neuropile Cells"}
create neutral /network_lgc_i
coord_reader ./data/lgn_inhib_coords.dat /network_lgc_i {No_lgc_i}
for (i=1; i<={No_lgc_i}; i=i+1)
make_lgc_neuron /network_lgc_i/{"cell"@{i}} {1}
make_spike /network_lgc_i/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
end
echo {"---> "@{No_lgc_i}@" Rinny LGC Neuropiles Created"}
//============= CREATING GENICULATE CELLS ===============//
echo {"---> Creating Linear Geniculate Cells"}
create neutral /network_lgn
coord_reader ./data/lgn_coord.dat /network_lgn {No_lgn}
for (i=1; i<={No_lgn}; i=i+1)
make_lgc_neuron /network_lgn/{"cell"@{i}} {1}
make_spike /network_lgn/{"cell"@{i}}/soma spike {0.0} {0.01} {1}
end
echo {"---> "@{No_lgn}@" Linear Geniculate Cells Created"}
//============= CREATING LATERAL CELLS ==================//
include ../Library/lateral_adapt.g
echo {"---> Creating Lateral Cells"}
create neutral /network_lateral
coord_reader ./data/lateral_coord1.dat /network_lateral {No_lateral}
str i
for (i=1; i<={No_lateral}; i=i+1)
create gnoise {"noise_lateral"@{i}}
create diffamp {"diff_lateral"@{i}}
```

```
setfield ^ gain {variance} saturation {sat}
addmsg {"noise_lateral"@{i}} {"diff_lateral"@{i}} PLUS output
makecell_lateral /network_lateral/{"cell"@{i}}
make_spike /network_lateral/{"cell"@{i}}/soma spike
      {-0.02} {0.01} {1}
addmsg {"diff_lateral"@{i}} /network_lateral/{"cell"@{i}}
     /soma INJECT output
end
echo {"---> "@{No_lateral}@" Lateral Cells Created"}
//================== CREATING MEDIAL CELLS ===========//
include ../Library/medial_adapt.g
echo {"---> Creating Medial Cells"}
create neutral /network_medial
coord_reader ./data/medial_coord1.dat /network_medial {No_medial}
str i
for (i=1; i<={No_medial}; i=i+1)
create gnoise {"noise_medial"@{i}}
create diffamp {"diff_medial"@{i}}
setfield ^ gain {variance} saturation {sat}
addmsg {"noise_medial"@{i}} {"diff_medial"@{i}} PLUS output
makecell_medial /network_medial/{"cell"@{i}}
make_spike /network_medial/{"cell"@{i}}/soma spike {-0.01} {0.01} {1}
addmsg {"diff_medial"@{i}}
     /network_medial/{"cell"@{i}}/soma INJECT output
end
echo {"---> "@{No_medial}@" Medial Cells Created"}
//===================== CREATING STELLATE CELLS ==============//
include ../Library/stellate.g
echo {"---> Creating Stellate Cells"}
create neutral /network_stellate
coord_reader ./data/stellate_coord1.dat /network_stellate {No_stellate}
str i
```

```
for (i=1; i<={No_stellate}; i=i+1)
create gnoise {"noise_stellate"@{i}}
create diffamp {"diff_stellate"@{i}}
setfield ^ gain {variance} saturation {sat}
addmsg {"noise_stellate"@{i}} {"diff_stellate"@{i}} PLUS output
makecell_stellate /network_stellate/{"cell"@{i}}
make_spike /network_stellate/{"cell"@{i}}   //
     /soma spike {-0.02} {0.01} {1}
addmsg {"diff_stellate"@{i}} /network_medial/
     {"cell"@{i}}/soma INJECT output
end
echo {"---> "@{No_stellate}@" Stellate Cells Created"}
//================ CREATING HORIZONTAL CELLS =================//
include ../Library/horizontal.g
echo {"---> Creating Horizontal Cells"}
create neutral /network_horizontal
coord_reader ./data/horizontal_coord1.dat /network_horizontal
     {No_horizontal}
str i
for (i=1; i<={No_horizontal}; i=i+1)
create gnoise {"noise_horizontal"@{i}}
create diffamp {"diff_horizontal"@{i}}
setfield ^ gain {variance} saturation {sat}
addmsg {"noise_horizontal"@{i}}
     {"diff_horizontal"@{i}} PLUS output
makecell_horizontal /network_horizontal/{"cell"@{i}}
make_spike /network_horizontal/{"cell"@{i}}
     /soma spike {-0.02} {0.01} {1}
addmsg {"diff_horizontal"@{i}}
     /network_medial/{"cell"@{i}}
     /soma INJECT output
end
```

```
echo {"---> "@{No_horizontal}@" Horizontal Cells Created"}
echo {"STAGE 2::: --- COMPLETED ---"}
//=============================================================//
//================= CONNECTIONS FROM RETINA ===================//
//=============================================================//
echo {" "}
echo {" "}
echo {"======================================================"}
echo {"STAGE 3::: CONNECTING RETINA TO RINNY LGC MODEL"}
echo {"======================================================"}
//echo {"---> Retina Connections Skipped"}
//============ CONNECTIONS FROM A ON CELLS ================//
float gang_gang_w = 3 ;
int count = 0;
float tdelay = 1e-3 ;
float Gs=5e-9
float t2 = 3.0e-3
float tt2 = 0.3e-3
float E2 = 0.0e-3
float A_ON_CELL_COUNT = ACTIVE_A_ON_CELL_COUNT;
float A_OFF_CELL_COUNT = ACTIVE_A_OFF_CELL_COUNT;
float B_1_CELL_COUNT = ACTIVE_B_1_CELL_COUNT;
float B_2_CELL_COUNT = ACTIVE_B_2_CELL_COUNT;
float B_3_CELL_COUNT = ACTIVE_B_3_CELL_COUNT;
float x_coord_diff
float connect_rad = 15
float connect_rad_inhib_ret = 130 //(scaled)
float map_scale_fact = 4
float retina_patch_size = 50
float lgc_patch_size = 200
float gang_lgc_w = 9  // 9 Changed via function
float gang_lgc_inhib = .60  // .55 Changed via function
```

```
for(i=1; i<=No_lgc; i=i+1)
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}} x}
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float distance = {sqrt  {(lgn_x_coord)**2 + (lgn_y_coord)**2} }
if(distance <= {lgc_patch_size + connect_rad})
count = count + 1;
end
end
echo {"---> Connection Parameters..."}
echo {"---> LGC Patch Size: "@ {lgc_patch_size}}
echo {"---> Retina Patch Size: "@ {retina_patch_size}}
echo {"---> # of Possible LGC Cellplates: "@ {count}}
echo {"---> Ganglion to LGC weight: "@ {gang_lgc_w}}
count = 0;
for(i=1; i<=No_lgc_i; i=i+1)
float lgn_x_coor = {getfield /network_lgc_i/{"cell"@{i}} x}
float lgn_y_coor = {getfield /network_lgc_i/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float distance = {sqrt  {(lgn_x_coord)**2 + (lgn_y_coord)**2} }
if(distance <= {lgc_patch_size})
count = count + 1;
end
end
echo {"---> # of Possible LGC Neuropile Cells: "@ {count}}
echo {"---> Ganglion to LGC weight: "@ {gang_lgc_inhib}}
count = 0;
echo {"---> Cellplates: Connecting A_ON Retinal Cells"}
for (i=1; i<=No_lgc; i=i+1)
for (j=1; j<=A_ON_CELL_COUNT; j=j+1)
```

```
float gang_x_coor = {getfield /retina/A_ON/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/A_ON/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad} >= distance)
count = count + 1;
make_synapse /network_lgc/{"cell"@{i}}/dend6
     {"gang_ON_lgc"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/A_ON/{"cell"@{j}}/soma/spike
     /network_lgc/{"cell"@{i}}/dend6/{"gang_ON_lgc"@{j}} SPIKE
setfield /network_lgc/{"cell"@{i}}/dend6/{"gang_ON_lgc"@{j}} \
synapse[{{getfield /network_lgc/{"cell"@{i}}
     /dend6/{"gang_ON_lgc"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend6/
     {"gang_ON_lgc"@{j}} nsynapses} -1}].weight {gang_lgc_w}
end
end
end
echo {"---> # Cellplate to A_ON Connections: "@ {count}}
count = 0;
echo {"---> Neuropile: Connecting A_ON Retinal Cells"}
for (i=1; i<=No_lgc_i; i=i+1)
for (j=1; j<=A_ON_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/A_ON/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc_i/{"cell"@{i}} x}
```

```
float gang_y_coor = {getfield /retina/A_ON/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc_i/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad_inhib_ret} >= distance)
count = count + 1;
make_synapse /network_lgc_i/{"cell"@{i}}/soma
     {"gang_ON_lgc_i"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/A_ON/{"cell"@{j}}/soma/spike /network_lgc_i/
     {"cell"@{i}}/soma/{"gang_ON_lgc_i"@{j}} SPIKE
setfield /network_lgc_i/{"cell"@{i}}/soma/{"gang_ON_lgc_i"@{j}} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_ON_lgc_i"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_ON_lgc_i"@{j}} nsynapses} -1}].weight {gang_lgc_inhib}
end
end
end
echo {"---> # Neuropile to A_ON Connections: "@ {count}}
//========= CONNECTIONS FROM A OFF CELLS ==========//
echo {"---> Cellplate: Connecting A_OFF Retinal Cells"}
count = 0;
for (i=1; i<=No_lgc; i=i+1)
for (j=1; j<=A_OFF_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/A_OFF/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/A_OFF/{"cell"@{j}}/soma y}
```

```
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad} >= distance)
count = count + 1;
make_synapse /network_lgc/{"cell"@{i}}
     /dend7 {"gang_OFF_lgc"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/A_OFF/{"cell"@{j}}/soma/spike /network_lgc/
     {"cell"@{i}}/dend7/{"gang_OFF_lgc"@{j}} SPIKE
setfield /network_lgc/{"cell"@{i}}/dend7/{"gang_OFF_lgc"@{j}} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend7/
     {"gang_OFF_lgc"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend7/
     {"gang_OFF_lgc"@{j}} nsynapses} -1}].weight {gang_lgc_w}
end
end
end
echo {"---> # Cellplate to A_OFF Connections: "@ {count}}
count = 0;
echo {"---> Neuropile: Connecting A_OFF Retinal Cells"}
for (i=1; i<=No_lgc_i; i=i+1)
for (j=1; j<=A_OFF_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/A_OFF/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc_i/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/A_OFF/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc_i/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
```

```
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad_inhib_ret} >= distance)
count = count + 1;
make_synapse /network_lgc_i/{"cell"@{i}}/soma
     {"gang_OFF_lgc_i"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/A_OFF/{"cell"@{j}}/soma/spike
     /network_lgc_i/{"cell"@{i}}
     /soma/{"gang_OFF_lgc_i"@{j}} SPIKE
setfield /network_lgc_i/{"cell"@{i}}/soma/{"gang_OFF_lgc_i"@{j}} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_OFF_lgc_i"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_OFF_lgc_i"@{j}} nsynapses} -1}].weight
     {gang_lgc_inhib}
end
end
end
echo {"---> # Neuropile to A_OFF Connections: "@ {count}}
echo {"---> Cellplate: Connecting B1 Retinal Cells"}
count = 0;
for (i=1; i<=No_lgc; i=i+1)
for (j=1; j<=B_1_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/B_1/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/B_1/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
```

```
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad} >= distance)
count = count + 1;
make_synapse /network_lgc/{"cell"@{i}}/dend5
     {"gang_B1_lgc"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/B_1/{"cell"@{j}}/soma/spike
     /network_lgc/{"cell"@{i}}/dend5/{"gang_B1_lgc"@{j}} SPIKE
setfield /network_lgc/{"cell"@{i}}/dend5/{"gang_B1_lgc"@{j}} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend5/
     {"gang_B1_lgc"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend5/
     {"gang_B1_lgc"@{j}} nsynapses} -1}].weight {gang_lgc_w}
end
end
end
echo {"---> # Cellplate to B1 Connections: "@ {count}}
count = 0;
echo {"---> Neuropile: Connecting B1 Retinal Cells"}
for (i=1; i<=No_lgc_i; i=i+1)
for (j=1; j<=B_1_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/B_1/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc_i/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/B_1/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc_i/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
```

```
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad_inhib_ret} >= distance)
count = count + 1;
make_synapse /network_lgc_i/{"cell"@{i}}/soma
     {"gang_B1_lgc_i"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/B_1/{"cell"@{j}}/soma/spike
     /network_lgc_i/{"cell"@{i}}/soma/{"gang_B1_lgc_i"@{j}} SPIKE
setfield /network_lgc_i/{"cell"@{i}}/soma/{"gang_B1_lgc_i"@{j}} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_B1_lgc_i"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_B1_lgc_i"@{j}} nsynapses} -1}].weight {gang_lgc_inhib}
end
end
end
echo {"---> # Neuropile to B1 Connections: "@ {count}}
echo {"---> Cellplate: Connecting B2 Retinal Cells"}
count = 0;
for (i=1; i<=No_lgc; i=i+1)
for (j=1; j<=B_2_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/B_2/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/B_2/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
```

```
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad} >= distance)
count = count + 1;
make_synapse /network_lgc/{"cell"@{i}}
     /dend5 {"gang_B2_lgc"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/B_2/{"cell"@{j}}/soma/spike /network_lgc/
     {"cell"@{i}}/dend5/{"gang_B2_lgc"@{j}} SPIKE
setfield /network_lgc/{"cell"@{i}}/dend5/{"gang_B2_lgc"@{j}} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend5/
     {"gang_B2_lgc"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc/{"cell"@{i}}
     /dend5/{"gang_B2_lgc"@{j}} nsynapses} -1}].weight {gang_lgc_w}
end
end
end
echo {"---> # Cellplate to B2 Connections: "@ {count}}
count = 0;
echo {"---> Neuropile: Connecting B2 Retinal Cells"}
for (i=1; i<=No_lgc_i; i=i+1)
for (j=1; j<=A_OFF_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/B_2/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc_i/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/B_2/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc_i/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
```

```
if ({connect_rad_inhib_ret} >= distance)
count = count + 1;
make_synapse /network_lgc_i/{"cell"@{i}}/soma
     {"gang_B2_lgc_i"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/B_2/{"cell"@{j}}/soma/spike /network_lgc_i/
     {"cell"@{i}}/soma/{"gang_B2_lgc_i"@{j}} SPIKE
setfield /network_lgc_i/{"cell"@{i}}/soma/{"gang_B2_lgc_i"@{j}} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_B2_lgc_i"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_B2_lgc_i"@{j}} nsynapses} -1}].weight {gang_lgc_inhib}
end
end
end
echo {"---> # Neuropile to B2 Connections: "@ {count}}
echo {"---> Cellplate: Connecting B3 Retinal Cells"}
count = 0;
for (i=1; i<=No_lgc; i=i+1)
for (j=1; j<=B_3_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/B_3/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/B_3/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad} >= distance)
count = count + 1;
```

```
make_synapse /network_lgc/{"cell"@{i}}/dend4
     {"gang_B3_lgc"@{j}} {Gs} {E2} {t2} {tt2}
addmsg /retina/B_3/{"cell"@{j}}/soma/spike /network_lgc/
     {"cell"@{i}}/dend4/{"gang_B3_lgc"@{j}} SPIKE
setfield /network_lgc/{"cell"@{i}}/dend4/{"gang_B3_lgc"@{j}} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend4/
     {"gang_B3_lgc"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/dend4/
     {"gang_B3_lgc"@{j}} nsynapses} -1}].weight {gang_lgc_w}
end
end
end
echo {"---> # Cellplate to B3 Connections: "@ {count}}
count = 0;
echo {"---> Neuropile: Connecting B3 Retinal Cells"}
for (i=1; i<=No_lgc_i; i=i+1)
for (j=1; j<=A_OFF_CELL_COUNT; j=j+1)
float gang_x_coor = {getfield /retina/B_3/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc_i/{"cell"@{i}} x}
float gang_y_coor = {getfield /retina/B_3/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc_i/{"cell"@{i}} y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float gang_x_coord = gang_x_coor * map_scale_fact;
float gang_y_coord = gang_y_coor * map_scale_fact;
float distance = {sqrt  {(gang_x_coord -
     lgn_x_coord)**2 + (gang_y_coord - lgn_y_coord)**2} }
tdelay = {{{{abs {gang_y_coord} }/100} + 1} * 1e-3}
if ({connect_rad_inhib_ret} >= distance)
count = count + 1;
make_synapse /network_lgc_i/{"cell"@{i}}/soma
     {"gang_B3_lgc_i"@{j}} {Gs} {E2} {t2} {tt2}
```

```
addmsg /retina/B_3/{"cell"@{j}}/soma/spike /network_lgc_i/
     {"cell"@{i}}/soma/{"gang_B3_lgc_i"@{j}} SPIKE
setfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_B3_lgc_i"@{j}} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}/soma/
     {"gang_B3_lgc_i"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc_i/{"cell"@{i}}
     /soma/{"gang_B3_lgc_i"@{j}} nsynapses} -1}]
     .weight {gang_lgc_inhib}
end
end
end
echo {"---> # Neuropile to B3 Connections: "@ {count}}
count = 0;
echo {"STAGE 3::: --- COMPLETED ---"}
echo {" "}
echo {" "}

//=====================================================//
//========= CONNECTIONS IN THE VISUAL CORTEX ==========//
//=====================================================//
echo {"============================================="}
echo {"STAGE 4::: CONNECTING LGC MODEL TO LINEAR LGN MODEL"}
echo {"============================================="}
//============== Connections from LGC to LGN ================//
int m, n
int cell_counter=0;
float x_lgc
float y_lgc
float lgc_lgn_w = 1.1
float map_x_coord_1, map_x_coord_2
openfile {"./data/lgn_lgn_map.dat"} r
```

```
echo {"---> Setting RostralCaudal Linear Mapping Rinny LGC -> LGN"}
string={readfile {"./data/lgn_lgn_map.dat"} -1}
map_x_coord_1={string}
for (m=1; m <= {No_lgc}; m = m + 1)
x_lgc={getfield /network_lgc/{"cell"@{m}} x}
y_lgc={getfield /network_lgc/{"cell"@{m}} y}
if ({map_x_coord_1 - connect_rad} <= x_lgc && x_lgc <= map_x_coord_1)
cell_counter = cell_counter+1
make_synapse /network_lgn/{"cell"@{1}}/soma \
{"lgc_lgn"@{m}} {Gs} {E2} {t2} {tt2}
addmsg /network_lgc/{"cell"@{m}}/soma/spike \
network_lgn/{"cell"@{1}}/soma/{"lgc_lgn"@{m}} SPIKE
setfield /network_lgn/{"cell"@{1}}/soma/{"lgc_lgn"@{m}} \
synapse[{{getfield /network_lgn/{"cell"@{1}}/soma/{"lgc_lgn"@{m}} \
nsynapses} -1}].delay {.0001} \
synapse[{{getfield /network_lgn/{"cell"@{1}}/soma/{"lgc_lgn"@{m}} \
nsynapses} -1}].weight {lgc_lgn_w}
end
end
for (n=2; n <= {No_lgn}; n=n+1)
string={readfile {"./data/lgn_lgn_map.dat"} -1}
map_x_coord_2={string}
for (m=1; m <= {No_lgc}; m = m + 1)
x_lgc={getfield /network_lgc/{"cell"@{m}} x}
y_lgc={getfield /network_lgc/{"cell"@{m}} y}
if (x_lgc <= map_x_coord_2 && map_x_coord_1 < x_lgc)
cell_counter = cell_counter+1
make_synapse /network_lgn/{"cell"@{n}}/soma \
{"lgc_lgn"@{m}} {Gs} {E2} {t2} {tt2}
addmsg /network_lgc/{"cell"@{m}}/soma/spike \
network_lgn/{"cell"@{n}}/soma/{"lgc_lgn"@{m}} SPIKE
setfield /network_lgn/{"cell"@{n}}/soma/{"lgc_lgn"@{m}} \
```

```
synapse[{{getfield /network_lgn/{"cell"@{n}}/soma/{"lgc_lgn"@{m}} \
nsynapses} -1}].delay {.0001} \
synapse[{{getfield /network_lgn/{"cell"@{n}}/soma/{"lgc_lgn"@{m}} \
nsynapses} -1}].weight {lgc_lgn_w}
end
end
map_x_coord_1={map_x_coord_2}
end
for (m=1; m <= {No_lgc}; m = m + 1)
x_lgc={getfield /network_lgc/{"cell"@{m}} x}
y_lgc={getfield /network_lgc/{"cell"@{m}} y}
if (map_x_coord_2 <= x_lgc && x_lgc <= {map_x_coord_2 + connect_rad})
cell_counter = cell_counter+1
make_synapse /network_lgn/{"cell"@{201}}/soma \
{"lgc_lgn"@{m}} {Gs} {E2} {t2} {tt2}
addmsg /network_lgc/{"cell"@{m}}/soma/spike \
network_lgn/{"cell"@{201}}/soma/{"lgc_lgn"@{m}} SPIKE
setfield /network_lgn/{"cell"@{201}}/soma/{"lgc_lgn"@{m}} \
synapse[{{getfield /network_lgn/{"cell"@{201}}/soma/{"lgc_lgn"@{m}} \
nsynapses} -1}].delay {.0001} \
synapse[{{getfield /network_lgn/{"cell"@{201}}/soma/{"lgc_lgn"@{m}} \
nsynapses} -1}].weight {lgc_lgn_w}
end
end
closefile {"./data/lgn_lgn_map.dat"}
echo {"---> "@{cell_counter}@" LGC cells found and connected to LGN"}
echo {"---> Establishing inhibitory LGC to LGC connections"}
float connect_rad_lgc_lgc = 140;
float lgc_lgc_i_w = 0.7;
count = 0;
for (i=1; i<=No_lgc; i=i+1)
for (j=1; j<=No_lgc_i; j=j+1)
```

```
float gang_x_coor = {getfield /network_lgc_i/{"cell"@{j}}/soma x}
float lgn_x_coor = {getfield /network_lgc/{"cell"@{i}}/soma x}
float gang_y_coor = {getfield /network_lgc_i/{"cell"@{j}}/soma y}
float lgn_y_coor = {getfield /network_lgc/{"cell"@{i}}/soma y}
float lgn_y_coord = lgn_y_coor - 500;
float lgn_x_coord = lgn_x_coor - 450;
float lgn_x_coord2 = gang_x_coor - 450;
float lgn_y_coord2 = gang_y_coor - 500;
float distance = {sqrt  {(lgn_x_coord2 -
     lgn_x_coord)**2 + (lgn_y_coord2 - lgn_y_coord)**2} }
tdelay = {{{{abs {lgn_y_coord2} }/100} + 1} * 1e-3}
if ({connect_rad_lgc_lgc} >= distance)
count = count + 1;
make_synapse /network_lgc/{"cell"@{i}}/soma
     {"lgc_i_lgc"@{j}} {Gs} {E2_i_lgc} {t2} {tt2}
addmsg /network_lgc_i/{"cell"@{j}}/soma/spike
     /network_lgc/{"cell"@{i}}/soma/{"lgc_i_lgc"@{j}} SPIKE
setfield /network_lgc/{"cell"@{i}}/soma/{"lgc_i_lgc"@{j}} \
synapse[{{getfield /network_lgc/{"cell"@{i}}/soma/
     {"lgc_i_lgc"@{j}} nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lgc/{"cell"@{i}}
     /soma/{"lgc_i_lgc"@{j}} nsynapses} -1}].weight {lgc_lgc_i_w}
end
end
end
count = 0;
echo {"---> Inhibitory connections within Rinny LGC created"}
echo {"STAGE 4::: --- COMPLETED ---"}
echo {" "}
echo {" "}

//=====================================================//
```

```
//============= CONNECTIONS FROM LGN ================//
int i, j
echo {"================================================"}
echo {"STAGE 5::: ESTABLISH CONNECTIONS WITHIN CORTEX"}
echo {"================================================"}
int No_varicosity = 1374
float x_pre, y_pre, x_post, y_post, x_lgn, dist1,
     dist2, dist, tdelay, grade
float y_lgn = 0;
float lgn_index
openfile {"./data/varicosity_coord.dat"} r
echo {"---> Reading LGN Varicosity & Linking LGN to Visual Cortex"}
for (i=1; i <= {No_varicosity}; i = i + 1)
string={readfile {"./data/varicosity_coord.dat"} -l}
str1={substring {string} 1 15}
str2={substring {string} 16 31}
str3={substring {string} 32 47}
x_pre = {str1};
y_pre = {str2};
lgn_index = {str3};
x_lgn={getfield /network_lgn/{"cell"@{lgn_index}} x}
dist1 = {sqrt  {(x_pre - x_lgn)**2 + (y_pre - y_lgn)**2} }
//Connecting LGN to LATERAL ********************
for (j=1; j<= {No_lateral}; j = j + 1)
x_post = {getfield /network_lateral/{"cell"@{j}} x}
y_post = {getfield /network_lateral/{"cell"@{j}} y}
dist2={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist2 <= var_lat_r)
dist=dist1+dist2
tdelay= dist * 1e-3/ 0.18 //sec
make_synapse /network_lateral/{"cell"@{j}}/basal2 \
{"lgn_lat"@{i}} {Gs} {E2} {t2} {tt2}
```

```
addmsg /network_lgn/{"cell"@{lgn_index}}/soma/spike \
network_lateral/{"cell"@{j}}/basal2/{"lgn_lat"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal2/{"lgn_lat"@{i}} \
synapse[{{getfield /network_lateral/
     {"cell"@{j}}/basal2/{"lgn_lat"@{i}} \
     nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal2/{"lgn_lat"@{i}} \
     nsynapses} -1}].weight {lgn_lat_w}
end
end
//Connecting LGN to MEDIAL ****************
for (j=1; j<= {No_medial}; j = j + 1)
x_post = {getfield /network_medial/{"cell"@{j}} x}
y_post = {getfield /network_medial/{"cell"@{j}} y}
dist2={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist2 <= var_med_r)
dist=dist1+dist2
tdelay= dist * 1e-3/ 0.18 //sec
make_synapse /network_medial/{"cell"@{j}}/dend3 \
{"lgn_med"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_lgn/{"cell"@{lgn_index}}/soma/spike \
/network_medial/{"cell"@{j}}/dend3/{"lgn_med"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}/dend3/{"lgn_med"@{i}} \
synapse[{{getfield /network_medial
     /{"cell"@{j}}/dend3/{"lgn_med"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend3/{"lgn_med"@{i}} \
nsynapses} -1}].weight {lgn_med_w}
end
end
```

```
//Connecting LGN to STELLATE ******************
for (j=1; j<= {No_stellate}; j = j + 1)
x_post = {getfield /network_stellate/{"cell"@{j}} x}
y_post = {getfield /network_stellate/{"cell"@{j}} y}
dist2={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist2 <= var_stel_r)
dist=dist1+dist2
tdelay= dist * 1e-3/ 0.18 //sec
 make_synapse /network_stellate/{"cell"@{j}}/dend16 \
{"lgn_stel"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_lgn/{"cell"@{lgn_index}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend16/{"lgn_stel"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend16/{"lgn_stel"@{i}} \
synapse[{{getfield /network_stellate
     /{"cell"@{j}}/dend16/{"lgn_stel"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
    /dend16/{"lgn_stel"@{i}} \
nsynapses} -1}].weight {lgn_stel_w}
end
end
end
echo {"---> LGN Connections Complete"}
//========= CONNECTIONS FROM LATERAL =======//
echo {"---> Connecting Lateral Cells In Visual Cortex"}
for (i=1; i<={No_lateral}; i=i+1)
x_pre = {getfield /network_lateral/{"cell"@{i}} x}
y_pre = {getfield /network_lateral/{"cell"@{i}} y}
//Connecting LATERAL to LATERAL ******************
for (j=1; j<={No_lateral}; j=j+1)
x_post = {getfield /network_lateral/{"cell"@{j}} x}
y_post = {getfield /network_lateral/{"cell"@{j}} y}
```

```
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= lat_lat_r && i!=j)
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre}
     {y_post} {lat_lat_r/var}}
//==== AMPA ******
make_synapse /network_lateral/{"cell"@{j}}/basal3 \
{"lat_lat_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/basal3/{"lat_lat_a"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal3/{"lat_lat_a"@{i}} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal3/{"lat_lat_a"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral
     /{"cell"@{j}}/basal3/{"lat_lat_a"@{i}} \
nsynapses} -1}].weight {lat_lat_w * grade}
//==== NMDA ******
make_nmda /network_lateral/{"cell"@{j}}/basal3 \
{"lat_lat_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/basal3/{"lat_lat_n"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal3/{"lat_lat_n"@{i}} \
synapse[{{getfield /network_lateral
     /{"cell"@{j}}/basal3/{"lat_lat_n"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral
     /{"cell"@{j}}/basal3/{"lat_lat_n"@{i}} \
nsynapses} -1}].weight {0.03 * var * grade}
end
end
//Connecting LATERAL to MEDIAL *****************
```

```
for (j=1; j<={No_medial}; j=j+1)
x_post = {getfield /network_medial/{"cell"@{j}} x}
y_post = {getfield /network_medial/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= lat_med_r)
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre} {y_post} {lat_med_r/var}}
//==== AMPA ******
make_synapse /network_medial/{"cell"@{j}}/dend8 \
{"lat_med_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend8/{"lat_med_a"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}/dend8/{"lat_med_a"@{i}} \
synapse[{{getfield /network_medial
     /{"cell"@{j}}/dend8/{"lat_med_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial
     /{"cell"@{j}}/dend8/{"lat_med_a"@{i}}\
nsynapses} -1}].weight {lat_med_w * grade}
//==== NMDA ******
make_nmda /network_medial/{"cell"@{j}}/dend8 \
{"lat_med_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend8/{"lat_med_n"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}/dend8/{"lat_med_n"@{i}} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend8/{"lat_med_n"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend8/{"lat_med_n"@{i}}\
nsynapses} -1}].weight {0.06 * var * grade}
end
```

```
end
//Connecting LATERAL to STELLATE ************
for (j=1; j<={No_stellate}; j=j+1)
x_post = {getfield /network_stellate/{"cell"@{j}} x}
y_post = {getfield /network_stellate/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= lat_stel_r)
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre} {y_post} {lat_stel_r/var}}
//==== AMPA ******
make_synapse /network_stellate/{"cell"@{j}}/dend8 \
{"lat_stel_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend8/{"lat_stel_a"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend8/{"lat_stel_a"@{i}} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend8/{"lat_stel_a"@{i}} \
     nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend8/{"lat_stel_a"@{i}}\
     nsynapses} -1}].weight {lat_stel_w * grade}
//==== NMDA ******
make_nmda /network_stellate/{"cell"@{j}}/dend8 \
{"lat_stel_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend8/{"lat_stel_n"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend8
     /{"lat_stel_n"@{i}} \
synapse[{{getfield /network_stellate
     /{"cell"@{j}}/dend8/{"lat_stel_n"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate
```

```
     /{"cell"@{j}}/dend8/{"lat_stel_n"@{i}}\
nsynapses} -1}].weight {0.1 * var * grade} //0.1
end
end
//Connecting LATERAL to HORIZONTAL ******************
for (j=1; j<={No_horizontal}; j=j+1)
x_post = {getfield /network_horizontal/{"cell"@{j}} x}
y_post = {getfield /network_horizontal/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= lat_hor_r)
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre}
     {y_post} {lat_hor_r/var}}
//==== AMPA ******
make_synapse /network_horizontal/{"cell"@{j}}/dend2 \
{"lat_hor_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_horizontal/{"cell"@{j}}/dend2/{"lat_hor_a"@{i}} SPIKE
setfield /network_horizontal/{"cell"@{j}}/dend2/{"lat_hor_a"@{i}} \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
     /dend2/{"lat_hor_a"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
     /dend2/{"lat_hor_a"@{i}}\
nsynapses} -1}].weight {lat_hor_w * grade}
//==== NMDA ******
make_nmda /network_horizontal/{"cell"@{j}}/dend2 \
{"lat_hor_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_lateral/{"cell"@{i}}/soma/spike \
/network_horizontal/{"cell"@{j}}/dend2/{"lat_hor_n"@{i}} SPIKE
setfield /network_horizontal/{"cell"@{j}}/dend2/{"lat_hor_n"@{i}} \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
```

```
     /dend2/{"lat_hor_n"@{i}} \
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
     /dend2/{"lat_hor_n"@{i}}\
nsynapses} -1}].weight {0.19 * var * grade} //0.19
end
end
end
echo {"---> Lateral Cells Connected"}
//==== CONNECTIONS FROM MEDIAL ==========//
echo {"---> Connecting Medial Cells in Visual Cortex"}
for (i=1; i<={No_medial}; i=i+1)
x_pre = {getfield /network_medial/{"cell"@{i}} x}
y_pre = {getfield /network_medial/{"cell"@{i}} y}
//Connecting MEDIAL to LATERAL ***************************
for (j=1; j<={No_lateral}; j=j+1)
x_post = {getfield /network_lateral/{"cell"@{j}} x}
y_post = {getfield /network_lateral/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= med_lat_r )
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre} {y_post} {med_lat_r/var}}
//==== AMPA ******
make_synapse /network_lateral/{"cell"@{j}}/basal4 \
{"med_lat_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/basal4/{"med_lat_a"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal4/{"med_lat_a"@{i}} \
synapse[{{getfield /network_lateral
      /{"cell"@{j}}/basal4/{"med_lat_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral
```

```
     /{"cell"@{j}}/basal4/{"med_lat_a"@{i}}\
nsynapses} -1}].weight {med_lat_w * grade}
//==== NMDA ******
make_nmda /network_lateral/{"cell"@{j}}/basal4 \
{"med_lat_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/basal4/{"med_lat_n"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal4
     /{"med_lat_n"@{i}} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal4/{"med_lat_n"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal4/{"med_lat_n"@{i}}\
nsynapses} -1}].weight {0.0167 * var * grade}
end
end
//Connecting MEDIAL to MEDIAL **********************
for (j=1; j<={No_medial}; j=j+1)
x_post = {getfield /network_medial/{"cell"@{j}} x}
y_post = {getfield /network_medial/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= med_med_r && i!=j )
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre}
     {y_post} {med_med_r/var}}
//==== AMPA ******
make_synapse /network_medial/{"cell"@{j}}/dend7 \
{"med_med_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend7/{"med_med_a"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}/dend7/{"med_med_a"@{i}} \
```

```
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend7/{"med_med_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend7/{"med_med_a"@{i}}\
nsynapses} -1}].weight {med_med_w * grade}
//==== NMDA ******
make_nmda /network_medial/{"cell"@{j}}/dend7 \
{"med_med_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend7/{"med_med_n"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}/dend7/
     {"med_med_n"@{i}} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend7/{"med_med_n"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}/dend7
     /{"med_med_n"@{i}}\
nsynapses} -1}].weight {0.0167 * var * grade}
end
end
//Connecting MEDIAL to STELLATE ******************
for (j=1; j<={No_stellate}; j=j+1)
x_post = {getfield /network_stellate/{"cell"@{j}} x}
y_post = {getfield /network_stellate/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= med_stel_r)
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre}
     {y_post} {med_stel_r/var}}
//==== AMPA ******
make_synapse /network_stellate/{"cell"@{j}}/dend3 \
```

```
{"med_stel_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend3/{"med_stel_a"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend3/
     {"med_stel_a"@{i}}  \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend3/{"med_stel_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend3/{"med_stel_a"@{i}}\
nsynapses} -1}].weight {med_stel_w * grade}
//==== NMDA ******
make_nmda /network_stellate/{"cell"@{j}}/dend3 \
{"med_stel_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend3/{"med_stel_n"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend3/
     {"med_stel_n"@{i}}  \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend3/{"med_stel_n"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend3/{"med_stel_n"@{i}}\
nsynapses} -1}].weight {0.11 * var * grade} //0.1
end
end
//Connecting MEDIAL to HORIZONTAL ***************
for (j=1; j<={No_horizontal}; j=j+1)
x_post = {getfield /network_horizontal/{"cell"@{j}} x}
y_post = {getfield /network_horizontal/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= med_hor_r)
```

```
tdelay= dist * 1e-3/ 0.05 //sec
grade = {gauss {x_pre} {x_post} {y_pre} {y_post} {med_hor_r/var}}
//==== AMPA ******
make_synapse /network_horizontal/{"cell"@{j}}/dend3 \
{"med_hor_a"@{i}} {Gs} {E2} {t2} {tt2}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_horizontal/{"cell"@{j}}/dend3/
     {"med_hor_a"@{i}} SPIKE
setfield /network_horizontal/{"cell"@{j}}/dend3
     /{"med_hor_a"@{i}}  \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
     /dend3/{"med_hor_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
     /dend3/{"med_hor_a"@{i}}\
nsynapses} -1}].weight {med_hor_w * grade}
//==== NMDA ******
make_nmda /network_horizontal/{"cell"@{j}}/dend3 \
{"med_hor_n"@{i}} {Gs} {En} {tn} {ttn}
addmsg /network_medial/{"cell"@{i}}/soma/spike \
/network_horizontal/{"cell"@{j}}/dend3/{"med_hor_n"@{i}} SPIKE
setfield /network_horizontal/{"cell"@{j}}/dend3/{"med_hor_n"@{i}}  \
synapse[{{getfield /network_horizontal/{"cell"@{j}}
     /dend3/{"med_hor_n"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_horizontal
     /{"cell"@{j}}/dend3/{"med_hor_n"@{i}}\
nsynapses} -1}].weight {0.22 * var * grade} //0.2
end
end
end
echo {"---> Medial Cells Connected"}
```

```
//======= CONNECTIONS FROM STELLATE=========//
echo {"---> Connecting Stellate Cells in Visual Cortex"}
float Stel_lat_w, Stel_med_w, s_gabab_lat, s_gabab_med
for (i=1; i<={No_stellate}; i=i+1)
x_pre = {getfield /network_stellate/{"cell"@{i}} x}
y_pre = {getfield /network_stellate/{"cell"@{i}} y}
if (x_pre > 1.0 && y_pre < 0.6)
Stel_lat_w = 7.8//7.7
Stel_med_w = 2.3//2.34
s_gabab_lat = 0.02//0.02
s_gabab_med = 0.01//0.01
else
Stel_lat_w = stel_lat_w
Stel_med_w =  stel_med_w
s_gabab_lat = 0.002//0.002
s_gabab_med = 0.001//0.001
end
//Connecting STELLATE to LATERAL ********************
for (j=1; j<={No_lateral}; j=j+1)
x_post = {getfield /network_lateral/{"cell"@{j}} x}
y_post = {getfield /network_lateral/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= stel_lat_r)
tdelay= dist * 1e-3/ 0.05 //sec
//==== GABAa ******
make_synapse /network_lateral/{"cell"@{j}}/apical3 \
{"stel_lat_a"@{i}} {Gs} {Ia} {ta} {tta}
addmsg /network_stellate/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/apical3/{"stel_lat_a"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/apical3/{"stel_lat_a"@{i}} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /apical3/{"stel_lat_a"@{i}}\
```

```
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /apical3/{"stel_lat_a"@{i}}\
nsynapses} -1}].weight {Stel_lat_w}
//==== GABAb ******
make_synapse /network_lateral/{"cell"@{j}}/apical3 \
{"stel_lat_b"@{i}} {Gs} {Ib} {tb} {ttb}
addmsg /network_stellate/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/apical3/{"stel_lat_b"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/apical3/{"stel_lat_b"@{i}} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /apical3/{"stel_lat_b"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /apical3/{"stel_lat_b"@{i}}\
nsynapses} -1}].weight {s_gabab_lat}
end
end
//Connecting STELLATE to MEDIAL ************************
for (j=1; j<={No_medial}; j=j+1)
x_post = {getfield /network_medial/{"cell"@{j}} x}
y_post = {getfield /network_medial/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= stel_med_r)
tdelay= dist * 1e-3/ 0.05 //sec
//==== GABAa ******
make_synapse /network_medial/{"cell"@{j}}/dend3 \
{"stel_med_a"@{i}} {Gs} {Ia} {ta} {tta}
addmsg /network_stellate/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend3
     /{"stel_med_a"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}
```

```
     /dend3/{"stel_med_a"@{i}} \
synapse[{{getfield /network_medial
     /{"cell"@{j}}/dend3/{"stel_med_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial
     /{"cell"@{j}}/dend3/{"stel_med_a"@{i}}\
nsynapses} -1}].weight {Stel_med_w}
//==== GABAb ******
make_synapse /network_medial/{"cell"@{j}}/dend3 \
{"stel_med_b"@{i}} {Gs} {Ib} {tb} {ttb}
addmsg /network_stellate/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend3
     /{"stel_med_b"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}
     /dend3/{"stel_med_b"@{i}} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend3/{"stel_med_b"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend3/{"stel_med_b"@{i}}\
nsynapses} -1}].weight {s_gabab_med}
end
end
//Connecting STELLATE to STELLATE **********************
for (j=1; j<={No_stellate}; j=j+1)
x_post = {getfield /network_stellate/{"cell"@{j}} x}
y_post = {getfield /network_stellate/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= stel_stel_r && i!=j)
tdelay= dist * 1e-3/ 0.05 //sec
//==== GABAa ******
make_synapse /network_stellate/{"cell"@{j}}/dend11 \
```

```
{"stel_stel_a"@{i}} {Gs} {Ia} {ta} {tta}
addmsg /network_stellate/{"cell"@{i}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend11
     /{"stel_stel_a"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend11
     /{"stel_stel_a"@{i}} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend11/{"stel_stel_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate/{"cell"@{j}}
     /dend11/{"stel_stel_a"@{i}}\
nsynapses} -1}].weight {stel_stel_w}
//==== GABAb ******
make_synapse /network_stellate/{"cell"@{j}}/dend11 \
{"stel_stel_b"@{i}} {Gs} {Ib} {tb} {ttb}
addmsg /network_stellate/{"cell"@{i}}/soma/spike \
/network_stellate/{"cell"@{j}}/dend11
     /{"stel_stel_b"@{i}} SPIKE
setfield /network_stellate/{"cell"@{j}}/dend11
     /{"stel_stel_b"@{i}} \
synapse[{{getfield /network_stellate
     /{"cell"@{j}}/dend11/{"stel_stel_b"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_stellate
     /{"cell"@{j}}/dend11/{"stel_stel_b"@{i}}\
nsynapses} -1}].weight {2.5e-4}
end
end
end
echo {"---> Stellate Cells Connected"}
//======= CONNECTIONS FROM HORIZONTAL ==============//
echo {"---> Connecting Horizontal Cells in Visual Cortex"}
```

```
float Hor_lat_w, Hor_med_w, h_gabab_lat, h_gabab_med
for (i=1; i<={No_horizontal}; i=i+1)
x_pre = {getfield /network_horizontal/{"cell"@{i}} x}
y_pre = {getfield /network_horizontal/{"cell"@{i}} y}
if (x_pre > 1.0 && y_pre < 0.6)
Hor_lat_w = 24.0//23.5
Hor_med_w = 8.7//
h_gabab_lat = 0.02//0.02
h_gabab_med = 0.01//0.01
else
Hor_lat_w = hor_lat_w
Hor_med_w = hor_med_w
h_gabab_lat = 0.002//0.002
h_gabab_med = 0.002//0.002
end
//Connecting HORIZONTAL to LATERAL *************************
for (j=1; j<={No_lateral}; j=j+1)
x_post = {getfield /network_lateral/{"cell"@{j}} x}
y_post = {getfield /network_lateral/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= hor_lat_r)
tdelay= dist * 1e-3/ 0.05 //sec
//==== GABAa ******
make_synapse /network_lateral/{"cell"@{j}}/basal4 \
{"hor_lat_a"@{i}} {Gs} {Ia} {ta} {tta}
addmsg /network_horizontal/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/basal4/{"hor_lat_a"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal4/{"hor_lat_a"@{i}} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal4/{"hor_lat_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
```

```
     /basal4/{"hor_lat_a"@{i}}\
nsynapses} -1}].weight {Hor_lat_w}
//==== GABAb ******
make_synapse /network_lateral/{"cell"@{j}}/basal4 \
{"hor_lat_b"@{i}} {Gs} {Ib} {tb} {ttb}
addmsg /network_horizontal/{"cell"@{i}}/soma/spike \
/network_lateral/{"cell"@{j}}/basal4/{"hor_lat_b"@{i}} SPIKE
setfield /network_lateral/{"cell"@{j}}/basal4
     /{"hor_lat_b"@{i}} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal4/{"hor_lat_b"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_lateral/{"cell"@{j}}
     /basal4/{"hor_lat_b"@{i}}\
nsynapses} -1}].weight {h_gabab_lat}
end
end
//Connecting HORIZONTAL to MEDIAL ************************
for (j=1; j<={No_medial}; j=j+1)
x_post = {getfield /network_medial/{"cell"@{j}} x}
y_post = {getfield /network_medial/{"cell"@{j}} y}
dist={sqrt  {(x_pre - x_post)**2 + (y_pre - y_post)**2} }
if   (dist <= hor_med_r)
tdelay= dist * 1e-3/ 0.05 //sec
//==== GABAa ******
make_synapse /network_medial/{"cell"@{j}}/dend9 \
{"hor_med_a"@{i}} {Gs} {Ia} {ta} {tta}
addmsg /network_horizontal/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend9
     /{"hor_med_a"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}
     /dend9/{"hor_med_a"@{i}} \
```

```
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend9/{"hor_med_a"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend9/{"hor_med_a"@{i}}\
nsynapses} -1}].weight {Hor_med_w}
//==== GABAb ******
make_synapse /network_medial/{"cell"@{j}}/dend9 \
{"hor_med_b"@{i}} {Gs} {Ib} {tb} {ttb}
addmsg /network_horizontal/{"cell"@{i}}/soma/spike \
/network_medial/{"cell"@{j}}/dend9/{"hor_med_b"@{i}} SPIKE
setfield /network_medial/{"cell"@{j}}/dend9/{"hor_med_b"@{i}} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend9/{"hor_med_b"@{i}}\
nsynapses} -1}].delay {tdelay} \
synapse[{{getfield /network_medial/{"cell"@{j}}
     /dend9/{"hor_med_b"@{i}}\
nsynapses} -1}].weight {h_gabab_med}
end
end
end
echo {"---> Horizontal Cells Connected"}
echo {"STAGE 5::: --- COMPLETED ---"}
echo {" "}
echo {" "}
echo {" "}
echo {" "}
echo {"MODEL CONFIGURATION COMPLETE..."}
echo {" "}
//=================Only modify below here=========
float tmax = 1.50
float dt = 0.00005
```

```
echo {"Initiating simulation: 0 to "@{tmax}@" sec with dt="@{dt}}
int N_of_steps = tmax/dt + 1
echo {"---> Number of steps to run: "@{N_of_steps}}
setclock 0 {dt}
setclock 1 {0.001}
//================= READ THE INPUT FILE ========
echo {"---> Setting Input Signals for Simulation"}
create neutral /in
create neutral /out
create neutral /in/A_ON
create neutral /out/A_ON
for (i=1; i<=ACTIVE_A_ON_CELL_COUNT; i=i+1)
create disk_in /in/A_ON/{"g_exc"@{i}}
create disk_in /in/A_ON/{"g_inh"@{i}}
setfield  /in/A_ON/{"g_exc"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}
     @"/g_exc_cell_"@{i}@".dat"} \
leave_open 0
setfield  /in/A_ON/{"g_inh"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}
     @"/g_inh_cell_"@{i}@".dat"} \
leave_open 0
useclock /in/A_ON/{"g_exc"@{i}} 1
useclock /in/A_ON/{"g_inh"@{i}} 1
addmsg /in/A_ON/{"g_exc"@{i}}/ /retina/A_ON/{"cell"@{i}}
     /soma CHANNEL val[0][0] {0}
addmsg /in/A_ON/{"g_inh"@{i}}/ /retina/A_ON/{"cell"@{i}}
     /soma CHANNEL val[0][0] {-0.07}
```

```
end
//=======================================
create neutral /in/A_OFF
create neutral /out/A_OFF
for (i=1; i<=ACTIVE_A_OFF_CELL_COUNT; i=i+1)
create disk_in /in/A_OFF/{"g_exc"@{i}}
create disk_in /in/A_OFF/{"g_inh"@{i}}
setfield  /in/A_OFF/{"g_exc"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}@"/g_exc_cell_"
     @{{i}+{ACTIVE_A_ON_CELL_COUNT}}@".dat"} \
leave_open 0
setfield  /in/A_OFF/{"g_inh"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}@"/g_inh_cell_"
     @{{i}+{ACTIVE_A_ON_CELL_COUNT}}@".dat"} \
leave_open 0
useclock /in/A_OFF/{"g_exc"@{i}} 1
useclock /in/A_OFF/{"g_inh"@{i}} 1
addmsg /in/A_OFF/{"g_exc"@{i}}/ /retina/A_OFF/{"cell"@{i}}
     /soma CHANNEL val[0][0] {0}
addmsg /in/A_OFF/{"g_inh"@{i}}/ /retina/A_OFF/{"cell"@{i}}
     /soma CHANNEL val[0][0] {-0.07}
end
create neutral /in/B_1
create neutral /out/B_1
for (i=1; i<=ACTIVE_B_1_CELL_COUNT; i=i+1)
create disk_in /in/B_1/{"g_exc"@{i}}
create disk_in /in/B_1/{"g_inh"@{i}}
setfield  /in/B_1/{"g_exc"@{i}} \
```

```
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}
     @"/g_exc_cell_"@{{i}+{ACTIVE_A_ON_CELL_COUNT}
     +{ACTIVE_A_OFF_CELL_COUNT}}@".dat"} \
leave_open 0
setfield  /in/B_1/{"g_inh"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}
     @"/g_inh_cell_"@{{i}
     +{ACTIVE_A_ON_CELL_COUNT}+{
     ACTIVE_A_OFF_CELL_COUNT}}@".dat"} \
leave_open 0
useclock /in/B_1/{"g_exc"@{i}} 1
useclock /in/B_1/{"g_inh"@{i}} 1
addmsg /in/B_1/{"g_exc"@{i}}/ /retina/B_1/{"cell"@{i}}
     /soma CHANNEL val[0][0] {0}
addmsg /in/B_1/{"g_inh"@{i}}/ /retina/B_1/{"cell"@{i}}
     /soma CHANNEL val[0][0] {-0.07}
end
create neutral /in/B_2
create neutral /out/B_2
for (i=1; i<=ACTIVE_B_2_CELL_COUNT; i=i+1)
create disk_in /in/B_2/{"g_exc"@{i}}
create disk_in /in/B_2/{"g_inh"@{i}}
setfield  /in/B_2/{"g_exc"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}@"
     /g_exc_cell_"@{{i}+{ACTIVE_A_ON_CELL_COUNT}+
     {ACTIVE_A_OFF_CELL_COUNT}+
```

```
     {ACTIVE_B_1_CELL_COUNT}}@".dat"} \
leave_open 0
setfield  /in/B_2/{"g_inh"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"
     @{INPUT_ANGLE}@"/g_inh_cell_"@{{i}
    +{ACTIVE_A_ON_CELL_COUNT}+
     {ACTIVE_A_OFF_CELL_COUNT}+
     {ACTIVE_B_1_CELL_COUNT}}@".dat"} \
leave_open 0
useclock /in/B_2/{"g_exc"@{i}} 1
useclock /in/B_2/{"g_inh"@{i}} 1
addmsg /in/B_2/{"g_exc"@{i}}/ /retina/B_2/{"cell"@{i}}
     /soma CHANNEL val[0][0] {0}
addmsg /in/B_2/{"g_inh"@{i}}/ /retina/B_2/{"cell"@{i}}
     /soma CHANNEL val[0][0] {-0.07}
end
create neutral /in/B_3
create neutral /out/B_3
for (i=1; i<=ACTIVE_B_3_CELL_COUNT; i=i+1)
create disk_in /in/B_3/{"g_exc"@{i}}
create disk_in /in/B_3/{"g_inh"@{i}}
setfield  /in/B_3/{"g_exc"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}
     @"/g_exc_cell_"@{{i}+{ACTIVE_A_ON_CELL_COUNT}+
     {ACTIVE_A_OFF_CELL_COUNT}+
     {ACTIVE_B_1_CELL_COUNT}
     +{ACTIVE_B_2_CELL_COUNT}}@".dat"} \
leave_open 0
```

```
setfield  /in/B_3/{"g_inh"@{i}} \
nx 1 \
ny 1 \
filename {"../SPEED10/ANGLE"@{INPUT_ANGLE}
     @"/g_inh_cell_"@{{i}
     +{ACTIVE_A_ON_CELL_COUNT}+
     {ACTIVE_A_OFF_CELL_COUNT}+
     {ACTIVE_B_1_CELL_COUNT}+
     {ACTIVE_B_2_CELL_COUNT}}@".dat"} \
leave_open 0
useclock /in/B_3/{"g_exc"@{i}} 1
useclock /in/B_3/{"g_inh"@{i}} 1
addmsg /in/B_3/{"g_exc"@{i}}/ /retina/B_3/{"cell"@{i}}
     /soma CHANNEL val[0][0] {0}
addmsg /in/B_3/{"g_inh"@{i}}/ /retina/B_3/{"cell"@{i}}
     /soma CHANNEL val[0][0] {-0.07}
end
// ================================================
echo {"---> Creating Output File: Cortex"}
create asc_file /output/{"cortex_response"}.dat
setfield ^ flush 1 leave_open 1
for (i=1; i <= {No_lateral}; i = i + 1)
addmsg  /network_lateral/{"cell"@{i}}/soma
     /output/{"cortex_response"}.dat   SAVE  Vm
end
for (i=1; i <= {No_medial}; i = i + 1)
addmsg  /network_medial/{"cell"@{i}}/soma
     /output/{"cortex_response"}.dat   SAVE  Vm
end
for (i=1; i <= {No_stellate}; i = i + 1)
addmsg  /network_stellate/{"cell"@{i}}/soma
     /output/{"cortex_response"}.dat   SAVE  Vm
```

```
end
for (i=1; i <= {No_horizontal}; i = i + 1)
addmsg  /network_horizontal/{"cell"@{i}}/soma
     /output/{"cortex_response"}.dat   SAVE  Vm
end
useclock /output/{"cortex_response"}.dat 1
echo {"---> Creating Output File: LGN"}
create asc_file /output/{"LGN_response"}.dat
setfield ^ flush 1 leave_open 1
for (i=1; i <= {No_lgn}; i = i + 1)
addmsg  /network_lgn/{"cell"@{i}}/soma  /output/
     {"LGN_response"}.dat   SAVE  Vm
end
useclock /output/{"LGN_response"}.dat 1
echo {"---> Creating Output File: Rinny LGC Neuropile"}
create asc_file /output/{"LGC_np_response"}.dat
setfield ^ flush 1 leave_open 1
for (i=1; i <= {No_lgc_i}; i = i + 1)
addmsg  /network_lgc_i/{"cell"@{i}}/soma  /output/
     {"LGC_np_response"}.dat   SAVE  Vm
end
useclock /output/{"LGC_np_response"}.dat 1
echo {"---> Creating Output File: Rinny LGC Cellplate"}
create asc_file /output/{"LGC_cp_response"}.dat
setfield ^ flush 1 leave_open 1
for (i=1; i <= {No_lgc}; i = i + 1)
addmsg  /network_lgc/{"cell"@{i}}/soma  /output/
     {"LGC_cp_response"}.dat   SAVE  Vm
end
useclock /output/{"LGC_cp_response"}.dat 1
//================================================
echo {"---> Creating Output File: MPB Retina"}
```

```
create asc_file /output/{"retina_response"}.dat
setfield ^ flush 1 leave_open 1
for (i=1; i <= {ACTIVE_A_ON_CELL_COUNT}; i = i + 1)
addmsg  /retina/A_ON/{"cell"@{i}}/soma
     /output/{"retina_response"}.dat   SAVE  Vm
end
for (i=1 ; i <= {ACTIVE_A_OFF_CELL_COUNT}; i = i + 1)
addmsg  /retina/A_OFF/{"cell"@{i}}/soma
     /output/{"retina_response"}.dat   SAVE  Vm
end
for (i=1; i <= {ACTIVE_B_1_CELL_COUNT}; i = i + 1)
addmsg  /retina/B_1/{"cell"@{i}}/soma
     /output/{"retina_response"}.dat   SAVE  Vm
end
for (i=1; i <= {ACTIVE_B_2_CELL_COUNT}; i = i + 1)
addmsg  /retina/B_2/{"cell"@{i}}/soma
     /output/{"retina_response"}.dat   SAVE  Vm
end
for (i=1; i <= {ACTIVE_B_3_CELL_COUNT}; i = i + 1)
addmsg  /retina/B_3/{"cell"@{i}}/soma
     /output/{"retina_response"}.dat   SAVE  Vm
end
echo {" "}
useclock /output/{"retina_response"}.dat 1
//================================================
randseed
check
reset
step {N_of_steps}
echo {" "}
echo {" "}
echo {"SIMULATION COMPLETE..."}
```

```
exit
```

Generate LGC Cell Coordinates

This MATLAB script takes as input one of the distribution matrices specified in Appendix D and returns a '.dat' file containing randomly assigned (x, y) coordinate pairs loosely satisfying the distribution described by that matrix. It produces the LGC Cell Plate coordinates and LGC Neuropile coordinates used in the FVSM code. For reference, these '.dat' files are also included in Appendix D.

```
% CELLULAR DISTRIBUTION GENERATOR


function [cell_coords,cell_count] = cell_dist(distr,len,height)
close all;
clc;
cell_count = 0;
dimens = size(distr);
part = 0;
height_ctr = 0;
cell_coords = [];


for i=1:dimens(1)
    height_ctr_max = height_ctr + height;
    h_min = height_ctr;
    h_max = height_ctr_max;

    for m=1:dimens(2)
        whole = floor(distr(i,m));
        cell_count = cell_count + whole;
        part = distr(i,m) - whole + part;
        l_min = (m - 1)*len;
        l_max = m*len;
        x = l_min + (l_max - l_min).*rand(1,whole);
```

```
        y = h_min + (h_max - h_min).*rand(1,whole);
    %    c_index = cell_count:(cell_count + whole);

        for k=1:whole
         cell_coords = [cell_coords; x(k)-50 900-y(k) 0.0;];
        end
    end
    height_ctr = height + height_ctr;
end
cell_coords = cell_coords';
fid = fopen('lgn_coords.dat', 'w');
fprintf(fid,'%16.7e %15.7e %15.7e\n',cell_coords);
fclose(fid);

lgn_lgn_map = [252:1.98:648]; %[4.23:4.23:854.23];

fid = fopen('lgn_lgn_map.dat', 'w');
fprintf(fid,'%16.7e\n',lgn_lgn_map);
fclose(fid);

figure(1); title('density map');
contour3(distr);



figure(2); title('model cell locations');
scatter(cell_coords(1,:),cell_coords(2,:),'blue')
xlabel('microns');
ylabel('microns');
```

LPF.g

This MATLAB codes performs low-pass filtering on the cortex response data files.

```
clc
close all
clear all
NREP = 60 ; % Number of repetitions
THRESHOLD = -0.034;
%ANGLES = 120:30:120;
ANGLES = [0 30 60 90 120 150 180 210 240 270 300 330];
tic
k10 = 10;
k50 = 50;
k100 = 100;
k500 = 500;
destinationDIR10 = './LPF/LPF_10';
destinationDIR50 = './LPF/LPF_50';
destinationDIR100 = './LPF/LPF_100';
destinationDIR500 = './LPF/LPF_500';
destinationDIR_SPIKE = './LPF_Results';
sourceDIR = '/home/ronaande/Desktop/
     SIMS/noisy_rinny_results/SPEED10';
srcFILEPREFIX = 'cortex_response';
mkdir(destinationDIR10);
mkdir(destinationDIR50);
mkdir(destinationDIR100);
mkdir(destinationDIR500);
n = 2 ;
LPF_SYS10 = zpk([],-k10*ones(1,n), k10^n) ;
LPF_SYS50 = zpk([],-k50*ones(1,n), k50^n) ;
LPF_SYS100 = zpk([],-k100*ones(1,n), k100^n) ;
```

```
LPF_SYS500 = zpk([],-k500*ones(1,n), k500^n) ;
for ANGLE = ANGLES
mkdir([destinationDIR10 '/ANGLE' num2str(ANGLE)]);
mkdir([destinationDIR50 '/ANGLE' num2str(ANGLE)]);
mkdir([destinationDIR100 '/ANGLE' num2str(ANGLE)]);
mkdir([destinationDIR500 '/ANGLE' num2str(ANGLE)]);
mkdir([destinationDIR_SPIKE '/ANGLE' num2str(ANGLE)]);
for REP = 1:NREP
DATA = load([sourceDIR '/' num2str(ANGLE) '/' num2str(REP)
     '/' srcFILEPREFIX '.dat']);
[ROWS,COLS] = size(DATA) ;
T_VEC = DATA(:,1) ;
LPF_DATA10 = zeros(ROWS,COLS) ;
LPF_DATA50 = zeros(ROWS,COLS) ;
LPF_DATA100 = zeros(ROWS,COLS) ;
LPF_DATA500 = zeros(ROWS,COLS) ;
LPF_DATA(:,1) = T_VEC ;
for J = 2:COLS
RAW_DATA = DATA(:,J) ;
THRESH_DATA = (RAW_DATA >= THRESHOLD) ;
SPIKE_DATA = ((THRESH_DATA
     - [0 ; THRESH_DATA(1:(end-1))]) > 0);
LPF_TEMP10 = lsim(LPF_SYS10,SPIKE_DATA,T_VEC);
LPF_TEMP50 = lsim(LPF_SYS50,SPIKE_DATA,T_VEC);
LPF_TEMP100 = lsim(LPF_SYS100,SPIKE_DATA,T_VEC);
LPF_TEMP500 = lsim(LPF_SYS500,SPIKE_DATA,T_VEC);
LPF_DATA10(:,J) = LPF_TEMP10;
LPF_DATA50(:,J) = LPF_TEMP50;
LPF_DATA100(:,J) = LPF_TEMP100;
LPF_DATA500(:,J) = LPF_TEMP500;
end
save([ destinationDIR10 '/ANGLE' num2str(ANGLE)
```

```
      '/' srcFILEPREFIX num2str(ANGLE) '_' num2str(REP)
      '.mat'],'LPF_DATA10');
save([ destinationDIR50 '/ANGLE' num2str(ANGLE) '/'
     srcFILEPREFIX num2str(ANGLE) '_' num2str(REP)
     '.mat'],'LPF_DATA50');
save([ destinationDIR100 '/ANGLE' num2str(ANGLE) '/'
     srcFILEPREFIX num2str(ANGLE) '_' num2str(REP) '
     .mat'],'LPF_DATA100');
save([ destinationDIR500 '/ANGLE' num2str(ANGLE) '/'
     srcFILEPREFIX num2str(ANGLE) '_' num2str(REP) '
     .mat'],'LPF_DATA500');
save([ destinationDIR_SPIKE '/ANGLE' num2str(ANGLE) '/'
     srcFILEPREFIX num2str(ANGLE) '_' num2str(REP) '.mat'],'
     SPIKE_DATA');
disp(['Angle: ' num2str(ANGLE) ' Repetition: ' num2str(REP) '
     Completed']); toc
end
end
```

PCA.m

This code performs the PCA analysis on the low-pass filtered cortex activity data.

```
tic
ROUNDING = 1 ; % In fractions of a milli second
CELL_START = 1 ; % Where to start
NCELL = 744 ; % Number of cells
NIP = 12 ; % Number of inputs
NREP = 60 ; % Number of repetitions
CELL_IDX = (1:NCELL) + 1 ; % NCELL ;
ANGLES = [0 30 60 90 120 150 180 210 240 270 300 330];%358 ;
T_SHIFT = 10 ; % In milli seconds
T_WIN = 100 ; % In milli seconds
T_F = 1500 ; % In milli seconds
dT = 1/ROUNDING ;
T_VEC = 0:dT:(T_F-(dT/2)) ;
signif_coords_a = 10 ;
signif_coords_b = 6 ;
K = 10;
dir_string = ['./LPF/LPF_' num2str(K)]
save_dir = [ './PCA/FVSM'] ;
mkdir(save_dir)
ALPHA_TEMP = './ALPHA/' ;
FILE_NAME = 'angle_' ;
for T_J = 0:(T_SHIFT):(T_F-T_WIN-1)
disp(T_J)
clear beta_coords
CovMatrix = zeros(length(CELL_IDX)) ;
T_a = ceil(T_J/dT)+1 ;
T_b = floor((T_WIN+T_J)/dT)+1 ;
for ANGLE = ANGLES
for REP = 1:NREP
```

```
load([dir_string '/ANGLE' num2str(ANGLE)
     '/cortex_response' num2str(ANGLE) '_' num2str(REP) '.mat']);
LPF_DATA = LPF_DATA10;
Y0 = LPF_DATA(T_a:T_b,CELL_IDX) - ones(length(T_a:T_b),1)
     *mean(LPF_DATA(T_a:T_b,CELL_IDX)) ;
CovMatrix = CovMatrix + Y0'*Y0 ;
clear Y0
end
end
disp([ num2str(T_J) 'ms Covariance matrix calculated...' ]) ; toc
CovMatrix = CovMatrix/(T_WIN*NIP*NREP) ;
[eVec_a,eVal_a] = eigs(CovMatrix,signif_coords_a) ;
eVec_a = eVec_a*sign(eVal_a) ;
mkdir(ALPHA_TEMP)
alpha_coords = zeros(length(T_a:T_b),signif_coords_a*NREP) ;
for ANGLE = ANGLES
for REP = 1:NREP
load([dir_string '/ANGLE' num2str(ANGLE)
     '/cortex_response' num2str(ANGLE) '_' num2str(REP) '.mat']);
LPF_DATA = LPF_DATA10;
Y0 = LPF_DATA(T_a:T_b,CELL_IDX) ;
temp_alpha = Y0*eVec_a ;
alpha_coords(:,(1:signif_coords_a)+
     (signif_coords_a*REP)) = temp_alpha ;
end
save([ALPHA_TEMP 'ALPHA_' num2str(ANGLE)
      '_T_' num2str(T_J) '.mat'],'alpha_coords') ;
end
disp( [ num2str(T_J) 'ms Alpha strand completed... ...' ]) ; toc
N_ALPHA = numel(temp_alpha) ;
clear Y0 temp_a alpha_coords T_VEC CovMatrix
     eVal_a eVec_a temp_alpha
```

```
BigAlphaMatrix = zeros(NIP*NREP,N_ALPHA) ;
q = 0 ;
for ANGLE = ANGLES
load([ALPHA_TEMP 'ALPHA_' num2str(ANGLE)
      '_T_' num2str(T_J) '.mat']) ;
for REP = 1:NREP
q = q + 1 ;
ALPHA = alpha_coords(:,(1:signif_coords_a)
     +(signif_coords_a*REP)) ;
TEMP =  reshape(ALPHA,1,numel(ALPHA)) ;
BigAlphaMatrix(q,:) = TEMP - mean(TEMP) ;
end
end
CovBigAlphaMatrix = (BigAlphaMatrix' * BigAlphaMatrix)/
     (NIP*NREP) ;
clear ALPHA TEMP
[eVec_b,eVal_b,C_FLAG] =
     eigs(CovBigAlphaMatrix,signif_coords_b) ;
eVec_b = eVec_b*sign(eVal_b) ;
clear CovBigAlphaMatrix
beta_coords = BigAlphaMatrix * eVec_b ;
clear BigAlphaMatrix
if (flag)
error(['eigs did not converge at T_J = ' num2str(T_J)])
end
save([save_dir '/PCA_POINTS_' num2str(T_J) '.mat']
     ,'beta_coords')
clear eVal_b eVec_b
disp([ num2str(T_J) 'ms Beta Strand Completed ... ... ...']) ; toc
end
imagesc(rand(250))
```