The Role of Correlations in the Development of the Visual System

Rocio Aguilar-Chongtay

Department of Artificial Intelligence

University of Edinburgh

1992

Abstract

In this project a mathematical model of *Ocular dominance column development* proposed by [Miller *et al* 89] was implemented. Miller assumes in his model that this development is driven by correlations involving two eyes' activity patterns. Miller only considers very simple forms of these correlations and it was assumed also that there are no correlations between the two eyes. The latter assumption has been proposed to be not biologically plausible [Dayan & Goodhill 92, Goodhill 91]. Recent theoretical analysis of Miller's model presented by [Dayan & Goodhill 92, Goodhill 91] suggests that by using more realistic correlations, a failure to simulate ocular dominance can be produced in Miller's model. In this project empirical exploration of the parameter space of the model is made. The results are analysed and compared in order to evaluate whether the empirical work supports these theoretical predictions.

Acknowledgements

I would firstly like to thank my supervisor, Geoff Goodhill, for his enormous amount of time, effort, and over all his patient guidance and help during the course of this project.

I would also like to thank Gillian Hayes for her time, contributions, comments and supervision to reach the goals of this project.

Deep thanks are due to Illona Bellos and Anthony Blake who provided useful comments on this dissertation.

Thanks to all those who contributed in my life this year through their friendship specially to the members of Vasteca group.

I wish to express my very great thanks to Jose A. Arevalo Ramirez for providing an enthusiastic and invaluable emotional support at any time.

I would also like to thank to the Dirección General de Asuntos del Personal Académico (DGAPA), and the Instituto de Fisiología Celular from the National Autonomous University of México for providing the financial support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this project a mathematical model of *Ocular dominance column development* proposed by K. Miller [Miller *et al* 89] was implemented. Recent analysis of Miller's model by P.S. Dyan and G. Goodhill [Dayan & Goodhill 92, Goodhill 91] suggests that the use of more realistic correlations than those used by Miller (such as positive correlation between eyes) could lead to a failure of the model to exhibit the phenomenon of *ocular dominance*. So far though, the analysis of [Dayan & Goodhill 92, Goodhill 91] has been entirely theoretical. This leads to the motivation of this project, namely, to implement Miller's model and to explore its parameter space empirically using parameters such as those proposed in Miller's work and the more realistic ones as proposed by [Dayan & Goodhill 92, Goodhill 91]. The results obtained from the implemented simulation are compared with both Miller's results and the theoretical predictions of [Dayan & Goodhill 92, Goodhill 91] in order to evaluate whether the theoretical predictions could be supported empirically.

1.1 A General Framework for the Project

1.1.1 Ocular Dominance Column Development

The *Ocular dominance column development* modeled by Miller is a phenomenon which occurs during development of the visual system within the brain. The features of interest are those involved in the formation of synaptic connections in the visual path between the *retina*, the *lateral geniculate nucleus* (LGN) and *primary visual cortex* (particularly the visual systems of cats and monkeys). These connections between the retinae, LGN and visual cortex vary during the early developmental stages in these kinds of organisms. The optic nerve fibres from the two retinae are connected to both of the LGNs. From each of the LGNs connections originating from both eyes are made to the visual cortex cells. Initially these connections are distributed in a homogeneous manner. After some time, these connections to the primary visual cortex are segregated into patches in which there are more connections from one eye than from the other. The way in which these patches are formed are as vertically stacked columns of cells with similar response properties. This process of segregation leads to one eye becoming dominant. This property is known as ocular dominance.

1.1.2 Brain, Connectivity, and Models

The brain of a vertebrate is formed from a very large number of functional units called *neurons* which are highly interconnected. Given the scale of this interconnectivity, genetic information alone would be insufficient to drive the development of each specific connection in the brain. Alternatively it has been proposed that the genetic information available in DNA encodes dynamical rules which drive the development of the brain's connections, to form particular patterns [Miller *et al* 89]. One interesting connectivity pattern observed in many parts of the visual system is that neighbouring cells (neurons) in one structure are connected to neighbouring cells in the next structure. This is known as the topographic map property.

The development of the visual system has been well studied by neuroscientists. One of the reasons for this is that standard neuroanatomical and neurophysiological techniques allow them to observe the process of connectivity alteration. This results in an increasing amount of biological information which in turn facilitates the modelling of these processes.

In Artificial Intelligence (AI), the brain has been studied as an information processing device. These studies include attempts to model some of the functionality of the brain by means of computer programs. During this AI research the brain has been viewed as an artificial interconnected neural network (*neural net*), which is able to carry out various information processing tasks simultaneously.

In trying to explain how certain properties of the visual system develop, several models have emerged. In particular some of them are based on a learning rule proposed by Hebb (as cited in [Goodhill 91]). This rule proposes that the connections between structures are reinforced in accord to the coincidences in their pre- and post-synaptic activity.

This project focuses on one of these Hebb-rule-based models, namely Miller's model of ocular dominance column development. Another Hebb-rule-based model, closely related to Miller's model is Linsker's model of visual receptive field development [Linsker 88]. Miller proposed his model to analyse the development of ocular dominance patches as a consequence of competition between the two eyes' activity patterns. In Miller's model it is assumed that connection development is driven by correlations in activity in the visual input. However, the results produced in [Miller *et al* 89] used only a very simple form of these correlations. In particular, it was assumed that there are no correlations between the two eyes. But, by introducing positive correlations between eyes, it has been predicted by [Dayan & Goodhill 92, Goodhill 91] that using Miller's model, ocular dominance cannot be simulated. This is the more important aspect being tested in the implementation. The other aspect is the observation of the effects on topographic refinement by manipulating various parameters.

1.2 Project Objectives

The main objectives of this project were:

- To implement Miller's model.
- To explore Miller's model empirically by observing the effect of varying parameters in it, such as introducing different correlations within and between the two eyes.
- To implement a graphic program to visualise the convergence from initial to final state, in order to analyse the results.
- To plot the results in graph form[1] in order to facilitate analysis (commercially available tools such as Xgraph and Mat-lab were used for this purpose).
- The comparison of the empirically obtained results with those presented in Miller's paper and the theoretical predictions from [Dayan & Goodhill 92, Goodhill 91], and the analysis of these results.

[1]Where graph means a curve plotter on an x,y Cartesian plane rather an interconnected node-link network as used in graph theory.

1.3 Overview of Dissertation

In the second chapter a general biological background overview is presented. A brief description of brain organisation is given, which is a very simplified version, to provide a context for the following chapter on biological data for models of ocular dominance. The third chapter contains brief descriptions of one model of map formation, and some models of ocular dominance. A more detailed description of Miller's model is also given in Chapter 3. The implementation of Miller's model is described in Chapter 4. Chapter 5 and 6 contain a description of the simulation results. The analysis of these results is given in Chapter 7 and finally some conclusions are presented in Chapter 8. Figure 1–1 provides a layout of the dissertation structure.

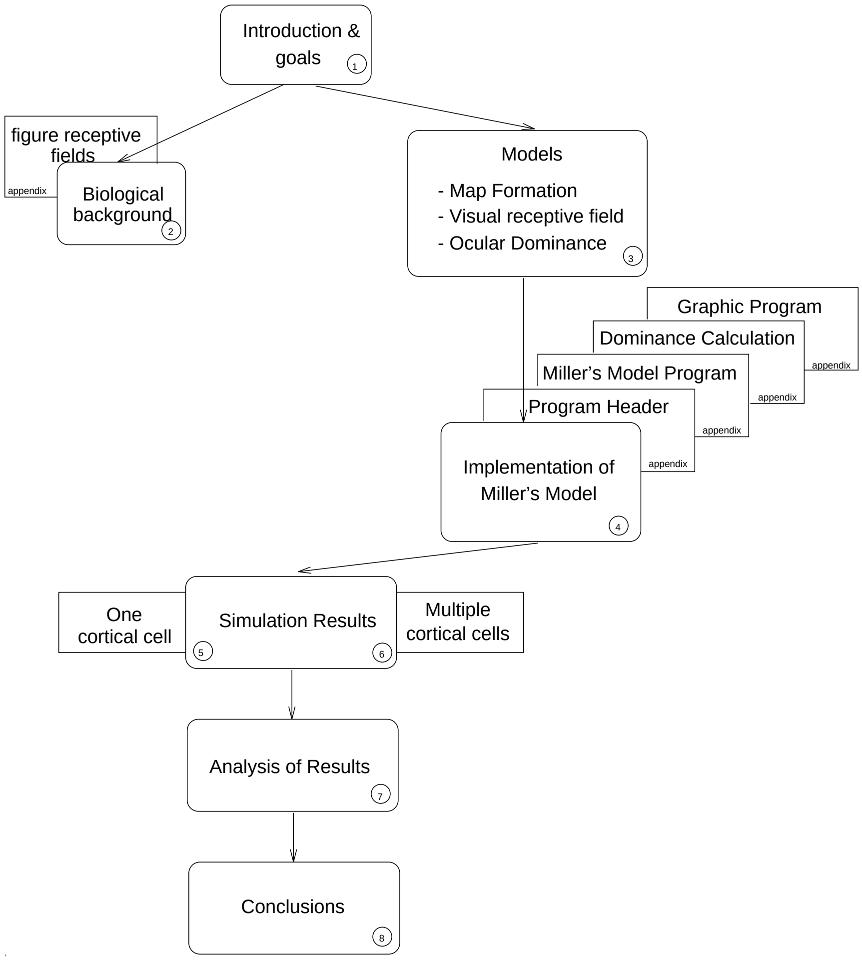

Figure 1–1: Organisation of the information in this dissertation.

Chapter 2

Biological Background

In this chapter the biological bases of ocular dominance in the visual pathway of cats and monkeys are described. We discuss the functional types of cortical cells involved, some aspects of ocular dominance stripe formation and some examples of stripe development under abnormal conditions.

2.1 Brain and Visual System

The brain's main functions involve perception and the processing and analysis of information. Each specific task is carried out by a specific region of the brain. These regions are formed by groups of interconnected brain cells. The interconnections between these neurons affect the different functions performed by each region. Given the huge scale of this interconnectivity, genetic information available in DNA is not sufficient to completely determine the configurations that result during brain development. The development of the brain, in general, and the development of the visual system, in particular, have been studied by neuroscientists. The development of the visual system has also attracted the interest of theoreticians because it is possible to compare the results of theoretical (or computer based) models with empirically observed biological information. Biological data is readily available because of the relative ease with which changes in the visual system under different conditions can be observed using standard anatomical and physiological techniques.

The first three sequential components of the visual path in which visual information is processed are as follows:

- Firstly, visual information in the form of light images, enters through the cell layers (*retinae*) of the receptors, namely the eyes.

- It then passes to the output nerve fibres of the eyes in the form of electrical signals. These group of fibres are divided in two subgroups at the optic chiasm to supply information from both eyes to each of the two *lateral geniculate nuclei (LGN)*, which are situated in the left and the right hemispheres.

- From there connections project to a specific layer (*layer IV*) in the *primary visual cortex* (see figure 2–1).

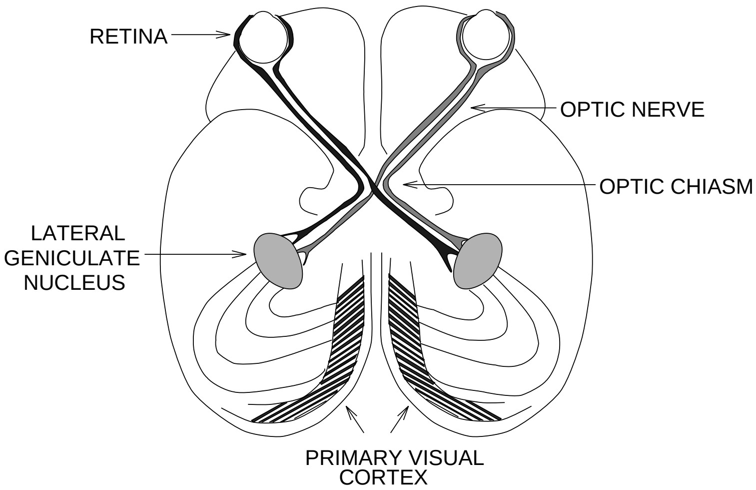

Figure 2–1: The visual path.

Retinal cells have different responses to different shapes of stimuli. Some retinal cells, for example have a stronger response to circularly shaped stimuli. To produce these responses, the optimal size of the stimuli also varies from cell to cell. The retinal cells have two types of receptive field: (a) the *on-center* receptive field which responds more strongly to a spot of light covering the central part of it and (b) the *off-center* receptive field types, which respond more strongly to stimuli of annular form. One remarkable feature of these fields is that the centre and surround interact in such way that they exhibit inhibitory and excitatory regions with mutual antagonistic action. Stimulation of *on-center* receptive fields produces a simultaneous inhibitive response in *off-center* receptive fields. In a similar way, if the stimulus received is annular then the *on-center* types are inhibited and the *off-center* types have a stronger response. These cell responses are transmitted through the optic nerve fibres to the LGN.

The right and left lateral geniculate nuclei are layered structures, and the interconnections between the retinae and LGN are *ordered topographically*. This means these interconnections are organised in such way that neighbouring cells in the retinae are connected with neighbouring cells in the LGN, [Kuffler *et al* 84, pages 26-44].

LGN neurons have similar receptive fields to those found in the cells of the retinae. If they receive a highly contrasting illumination, a strong response is obtained, and if the illumination is diffuse, a weak response is obtained. Visual information passes from different LGN layers to the primary visual cortex.

2.1.1 Primary Visual Cortex

As in the retinae and LGN, the cells in the primary visual cortex require a contrasting illumination to produce any significant response. These cells are organised into functional groups depending on their selective response to particular stimuli (visual pattern input). There are two main cell types, namely *simple cells* and *complex cells*.

The **simple cells** are those which have *orientation specificity*. This means that they respond to a well-defined line oriented in some direction within the stimulus at a particular position. Some simple cells also require a bar of defined length in order to obtain an optimum response.

Complex cells, as with the simple ones, require specific orientation stimuli but they do not have on-off areas. Then these cells can produce responses to stimuli with a well defined orientation but they do not need the stimuli to have a well-defined position. They respond to moving edges; some complex cells require lines with definite ends to produce a response [Kuffler *et al* 84, pages 49-57].

2.1.2 Structural Organisation

The visual cortex is formed by cells with similar response properties stacked vertically to form columns. The connections from the LGN to cells in some of these columns have input information from both eyes (binocularity) and they are grouped by functional similarity (orientation selectivity), where there is a column of cells for each possible orientation in each position. There are also some columns whose inputs are *exclusively* from one eye or the other (monocularity).

The structures known as hypercolumns are groups of cell columns with all the possible orientations represented and with inputs from one eye (monocular) and from both eyes (binoculars) (see figure 2–2), [Hubel & Wiesel 77], [Kuffler *et al* 84, pages 80-87]. These hypercolumns represent the functional unit in the primary visual cortex.

The way in which this final organisation is developed in the mammalian visual system has been shown to be influenced by the environment in the first days of life in some species or in the few days before birth in others, [Hubel & Wiesel 77]. In the case of ocular dominance, the connections between the LGN and cortical cells are initially in the form of diffuse overlapping projections without any apparent preference. But in the first days in which the organism receives light stimuli these connections are rearranged in columns which can be driven by only one eye or the other, some remaining with connections from both eyes [Kuffler *et al* 84, pages 533-535]. Figure 2–3 shows an ocular dominance stripe pattern after [Hubel & Wiesel 77] is shown.

2.1.3 Induced Abnormal Developments

In trying to understand how the visual system develops, several experimental works have been carried out by neuroscientists on the subject of ocular dominance

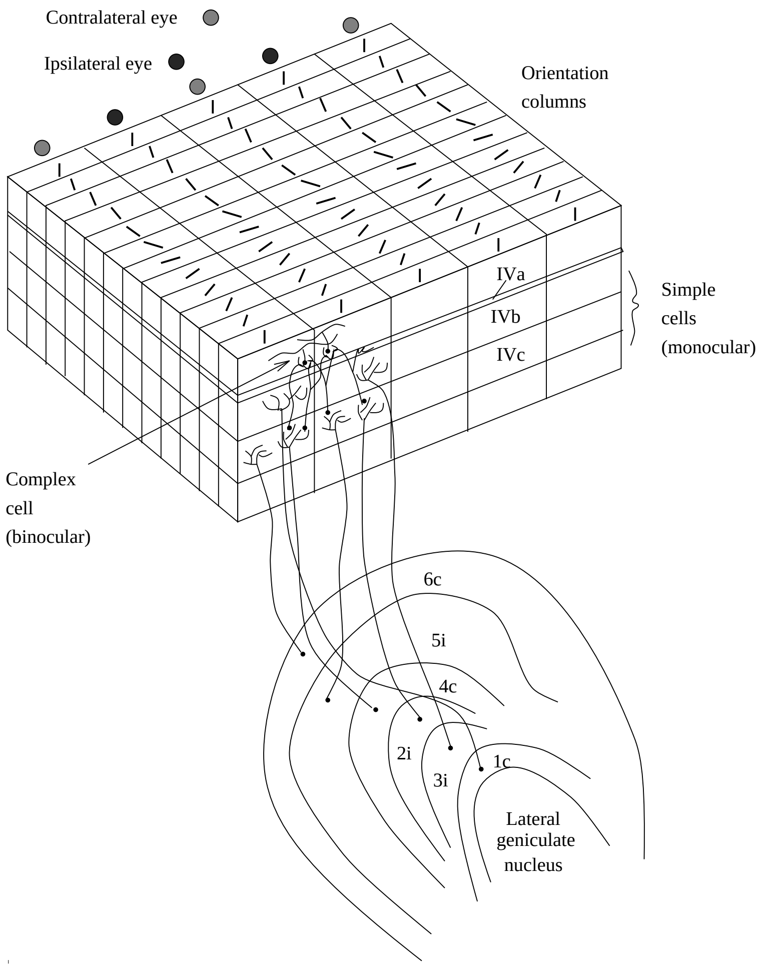

Figure 2–2: Visual cortex hypercolumns and LGN; c = contralateral and i = ipsilateral e.f. 6c, 5i, (after [Kuffler et al 84]).

development. In [Hubel & Wiesel 77] they have made an important contribution to the study of the relationship between neuronal activities and the subsequent rearrangement of synaptic connections.

They have shown that several biological mechanisms such as synaptic plasticity and activity-dependent mechanisms are involved in ocular dominance development. In some of their experiments they used monocular deprivation, i.e. one eye was sewn closed (or removed) during the *critical period* in which some developmental plasticity is still occurring (during weeks 4-5 in kittens and the first 6 weeks in monkeys), [Kuffler *et al* 84, pages 543-544]. After the critical period they reopened the sewn eye and it was observed that cortical cells only responded

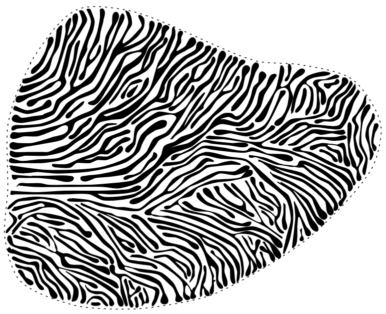

Figure 2–3: Ocular dominance stripes (after [Hubel and Wiesel 77]).

to stimuli from the eye which was maintained open, that means they lost their response ability to the closed eye. Anatomically the projections to the cortical cells from the open eye almost totally cover the primary visual cortex, producing an abnormal ocular dominance column formation. This process could be reversed if the eye that was deprived was reopened during the *critical period*. In contrast, binocular deprivation has little effect, and thus it appears that ocular dominance column formation is due to competition between the closed eye and open eye [Hubel & Wiesel 77], [Kuffler *et al* 84, page 547].

In another experiment to investigate the role of correlated activity in ocular dominance, Hubel and Wiesel preserved the same activity in both eyes, but destroyed the correlations between them with artificial strabismus, which was produced by altering the eye muscles causing the direction in which they were pointing to be different, resulting in each eye perceiving different parts of the scene. The alteration produced by this experiment in the cortical ocular dominance organisation was that almost all the cells were driven monocularly, [Kuffler *et al* 84, page 548].

The role of correlated activity between the two eyes in the ocular dominance segregation was also illustrated by [Stryker & Harris 86]. They produced the same effect as Hubel and Wiesel's strabismus experiment, blocking pharmacologically the neuronal activity in both eyes initially, and then using asynchronous electrical stimulation in the nerves from both eyes.

In ocular dominance segregation, the participation of cortical activity was proved by [Stryker & Harris 86]. They used pharmacological blocking in the pre- and post-synaptic activity in the cortex during the segregation, while monocular deprivation was produced. This experiments showed that the abnormal segregation observed with monocular deprivation alone was prevented [Miller 89, page 7].

Chapter 3

Ocular Dominance Segregation Models

There are several models which have been proposed attempting to explain how the properties of the visual system such as topography, orientation selectivity and ocular dominance develop. In this chapter Willshaw and von der Malsburg's model for topography and some models of ocular dominance segregation such as Swindale's, Bienenstock, Cooper and Munro's, Linsker's and Miller's models are described.

3.1 Previous Models

The mechanisms involved in the dominance stripe formation have been formalised into several computational models. Some of them are based on modified versions of a rule proposed by Hebb cited in [Goodhill 91]. This is a learning rule where connections between neurons are reinforced if pre- and post-synaptic activity takes place simultaneously.

The following is a brief review of some computational models based on the Hebbian rule addressing ocular dominance stripe formation and topography in the visual system.

3.1.1 Willshaw and von der Malsburg's Model

This model was proposed by [Willshaw & von derMalsburg 76] to work by self-organisation. It is a model of map formation rather than ocular dominance, but it is important to include it because it explores the idea that correlated activity could drive visual cortical development. Willshaw and von der Malsburg represented the pre- and post-synaptic sheets as two dimensional sheets of cells. The theory of topographical mapping is based in the topography property found in different structures of the nervous system, as mentioned for the cell organisation in the visual system, in which neighbouring cells in one sheet project connections to neighbouring cells in a second one. In this model the distances between cells in the pre-synaptic sheet are coded in the form of correlations in the electrical activity of these cells. This code is used to produce connections between neighbouring elements in the pre-synaptic sheet to the neighbouring elements in the post-synaptic sheet in a topographical way. The following assumptions were made in Willshaw and von der Malsburg's model:

1. The activity in neighbouring elements in the sheets are mutually reinforced because in the model it is assumed that there are short range excitatory connections.

2. On the other hand, longer range inhibitory connections are assumed; in this way the spreading of activity was prevented from getting too large.

3. Modifiable synapses interconnect the two sheets. As in Hebb's work these synapses are of the correlation type.

4. If some cells are active, and they are grouped together, then they fire strongly.

5. Changes are proportional to the activity product in the pre- and post-synaptic cells.

6. Synaptic strengths are prevented from growing infinitely large by limiting the total synaptic strength for each cell.

7. A fixed orientation is assumed for the map in the earliest stages of development. This is assumed to be provided genetically.

The initial conditions have an important role in determining the final orientation of the map. The model introduces *polarity markers*, which are cells from a specific pre-synaptic region initially connected to a post-synaptic region in the orientation required.

3.1.2 Swindale's Model

This model proposed by [Swindale 80] is based on the interactions between synapses in a cortical array as follows:

- In synapses with inputs from the **same** eye (known as RR interactions within for the right eye and LL for the interactions within the left eye) there is a property of *excitation-inhibition* between synapses where a synapse reinforces others which are in a range of 200 μm and inhibits those in a range of distances between 200-600 μm.

- In synapses from the **opposite** eye (known as RL and LR) the process is exactly reversed: the effect is inhibitory for distances shorter than 200 μm and if the distance is longer than this then the effect is excitatory.

It is assumed that the initial synapses strength values are random and intermixed, the model only takes the eye of synapse origin into account without considering the retinotopic position. The growth values are allowed to be positive or negative, and the synapse strength is increased or decreased according to these values. Synaptic strength is also limited within a given range. With this model it is possible to reproduce ocular dominance stripe formation and also other phenomena such as the monocular deprivation effect, the critical period and the binocular deprivation effect.

3.1.3 Bienenstock, Cooper and Munro's Model

A mathematical theory of the development of stimulus selectivity in the cortex was proposed by [Bienenstock *et al* 82]. This model uses a variation of a Hebb rule which takes into account:

- The reinforcement and weakening mechanisms of a synapse,
- The pre- and post-synaptic activity at each time step and
- A term which represents the varying time-averaged post-synaptic activity of the cell.

The authors proposed a temporal competition mechanism instead of the spatial competition mechanism:

- In the spatial competition mechanism two effects occur simultaneously: the increase of the strength of some synapses in a neuron and the decrease of the strength of other synapses in the same neuron.

- The temporal competition mechanism is proposed as being between input patterns (instead of different synapses) and the sizeof the post-synaptic response compared with a variable threshold value will determine whether the synaptic strength will decrease or increase.

In this model the change in a specific synapse strength depends, in a non-linear way, on the post-synaptic activity, producing an environmental selectivity effect. The selectivity uses the average response of the cell as a threshold value. Then if the post-synaptic activity is below this threshold the synaptic strength is weakened, and it is reinforced if that post-synaptic activity is above the threshold. This model also uses a decay factor such that if there is no input, the strength will decay to zero. Bienenstock *et al* obtained simulations of effects produced in the visual cortex such as binocular cells with both eyes' inputs well correlated, monocular cells with binocular deprivation conditions, and orientation selectivity.

3.2 Linsker's Model

Despite the fact that Linsker's model is a model of receptive field development, and Miller's is, as mentioned before, a model of ocular dominance, these two models are closely related: both are based on a different modification of the Hebbian rule. It is possible to say that given the simple simulation in Linsker's model of the layers in the visual path from one eye, a simple form of Miller's model is a roughly binocular version of Linsker's one. For this reason Linsker model is described in more detail in the following section.

In this model proposed by [Linsker 88] using a very simple model of receptive field development, he found that it was possible to produce a simulation of cells with response properties similar to those in the initial stages in the mammalian visual system.

Linsker's model consist of units in a network which are organised into multiple two dimensional abstract layers, the units (elements) of which are not connected to other units in the same layer but only with units in the previous layer. Each layer then develops independently as shown in figure 3–1, these abstract layers are the representation of layers in the visual system an the units represent cells.

This series of layers uses a particular form of Hebbian learning rule, (described below) to develop one layer after the other. Each unit sees a slightly different bit of input layer, where the connection strengths between layers is increased or weakened according to the degree of simultaneous pre-synaptic and post-synaptic activity.

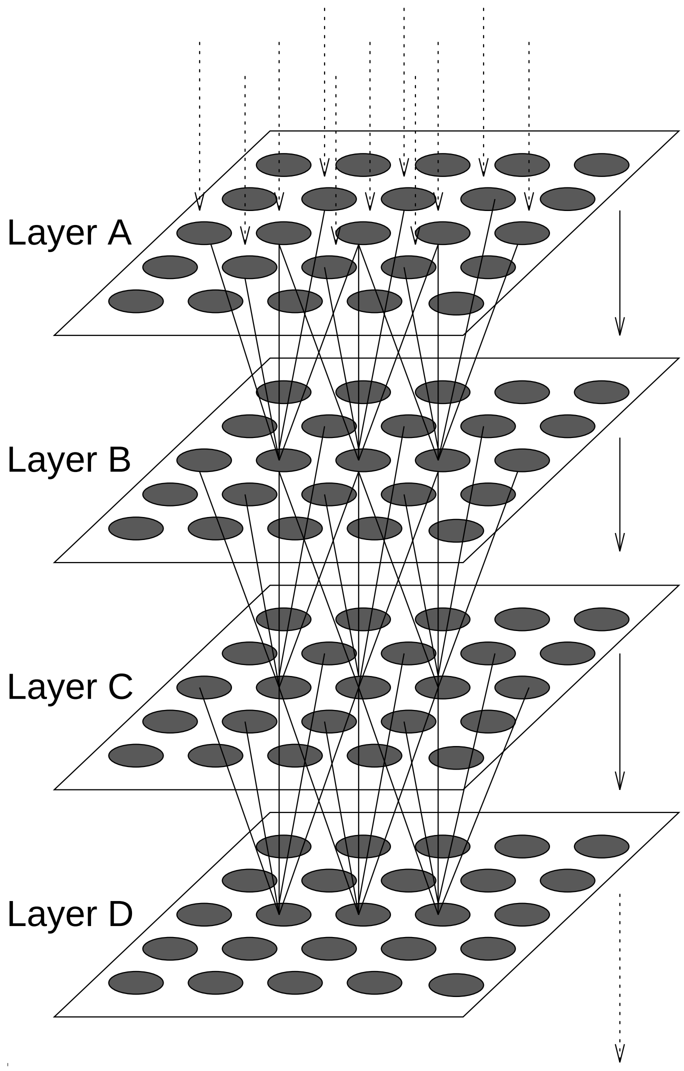

Figure 3–1: Linsker's model representations, a multilayered self-adaptative network, (after [Linsker, 1988]).

In the equations used in Linsker's model[1], correlations in layer $\mathcal{L}$ drive Hebb development in layer $\mathcal{M}$; then the activity in a unit $O^{\mathcal{M}}$ in $\mathcal{M}$ is calculated as:

$$O^{\mathcal{M}} = a + \sum_{n \epsilon \mathcal{L}} w_n V_n^{\mathcal{L}} \tag{3.1}$$

where a is a constant (which is assumed to be the same in all the units), w_n are the synaptic strengths and V_n are the activities in layer $\mathcal{L}$.

The following is the modified Hebbian rule used for this model:

$$\dot{w}_n = (O^{\mathcal{M}} - c_2)(V_n^{\mathcal{L}} - c_1) + c_3 \tag{3.2}$$

where c_1, c_2, c_3 are constants.

Then substituting for $O^{\mathcal{M}}$ from (equation 3.2)

$$\dot{w}_n = \left(\sum_{m \epsilon \mathcal{L}} w_m V_m^{\mathcal{L}} + a - c_2 \right) (V_n^{\mathcal{L}} - c_1) + c_3 \tag{3.3}$$

Finding the average rate of change of w_n in terms of:

1. By the average over the input activity at a synapse V_m in layer L, and it is assumed that it is the same for all the synapses.

2. The covariance matrix $Q_{nm}^{\mathcal{L}}$ of the activities of input cells n and m obtained by the following time average (denoted by $\langle ... \rangle$): $Q_{nm}^{\mathcal{L}} = \langle (V_n^{\mathcal{L}} - \bar{V}_n^{\mathcal{L}})(V_m^{\mathcal{L}} - \bar{V}_m^{\mathcal{L}}) \rangle$.

3. Combining constants in k_1 and k_2, (the value of the total connection strength is determined by these constants) and finally

[1]Math in section 3.2 after [Goodhill 91]

4. Assuming the uniformity of statistics of the patterns,

the following equation is obtained:

$$\dot{w_n} = \sum_{m \epsilon \mathcal{L}} (Q^{\mathcal{L}}_{nm} + k_2) w_m + k_1 \qquad (3.4)$$

Some features of this model are:

- A fixed topography is assumed,
- the weights are prevented from exceeding some limit, using a saturation constraint.

resulting in various receptive field structures depending on the parameters used, e.g. center-surround cell types and orientation-selective cells (as those described in Chapter 2).

To summarise, Linsker's model shows how a series of layers of units evolve according to a simple Hebbian type learning rule into various receptive fields. Linsker focused on the case where there is not input but random activity of the cells of layer A. Learning is driven by the correlation activities in each layer; resulting in various receptive field structures. This model contributes to the belief that using a simple Hebbian mechanism, it is possible to produce a significant organisation of individual receptive fields.

3.3 Miller's Model

Miller's model was proposed to analyse and simulate the development of ocular dominance patches as a consequence of competition between the activity patterns of the two eyes. Also taken into account is the fact that this development in biological systems is accompanied by the formation of monocular receptive fields for cortical neurons, topographic refinement of these receptive fields, and the progressive segregation of LGN axon arbors to patches.

This mathematical model is based (as is Linsker's) on a type of Hebbian learning rule, where the correlation between pre-synaptic units determines the form of receptive field development.

Miller focused his model on four biological features which are thought to play a role in the organisation of the ocular dominance patches. These are quoted below [Miller etal 1989, pp 605-606]:

1. *The patterns of initial connectivity of the geniculocortical afferents (inputs from geniculate to visual cortex) onto the cortical cells. These patterns involve the spread of afferent arbors and of cortical dendrites and are described by an "arbor function", A.*

2. *The patterns of activity in the afferents. These patterns are described by a set of four "correlation functions,"* C^{LL}, C^{RR}, C^{LR}, C^{RL}. *They describe correlations in activity between afferents serving the same eye, left or right (*C^{LL} *and* C^{RR}*) or serving different eyes (*C^{LR} *and* C^{RL}*).*

3. *Influences acting laterally within the cortex, whereby synapses on one cell can influence the competition occurring on nearby cells. These influences, described by a "cortical interaction function," I, may occur through cortical synaptic connections or diffusion of modulatory substances.*

4. *Constraints limiting the total synaptic strength supported by an afferent or cortical cell.*

Miller wanted to demonstrate how, from an initially uniform state, these features participate in ocular dominance segregation and patch formation. He described the development of structure within individual cortical receptive fields and geniculate axon terminal arborisations, using a combination of simple Hebbian type rules with some assumptions in a model of geniculate correlation.

The main assumptions in Miller's model are:

(a) fixed topography,

(b) each input line is multiplied by the weight on that line producing a linear weighted output of post-synaptic unit(s), i.e. each output of a cell is a linear weighted sum of its inputs,

(c) the changes in strength are considered to be driven by the averaged activity because it is assumed that the synapse's strength changes more slowly than the afferent activity patterns do,

(d) the three functions (arbor, correlation and interaction) are considered static during the development of a pattern of ocular dominance segregation.

3.3.1 Formulation of Miller's Model

The LGN for the left and right eyes are represented as two different square arrays. Miller originally used a size of 25 x 25. The cortex is represented as another square array with the same size. In figure 3–2 a representation example with 7x7 square arrays is shown.

The number of synapses between LGN and cortex are determined by the arbor function (or "arboreal/dendritic function")[2], which is a decreasing function of the retinotopic distance between geniculate and cortical cells.

---

[2]This name is given in view of the fact that each LGN cell has synapses to cortical cells with an anatomical connectivity in an arboreal manner.

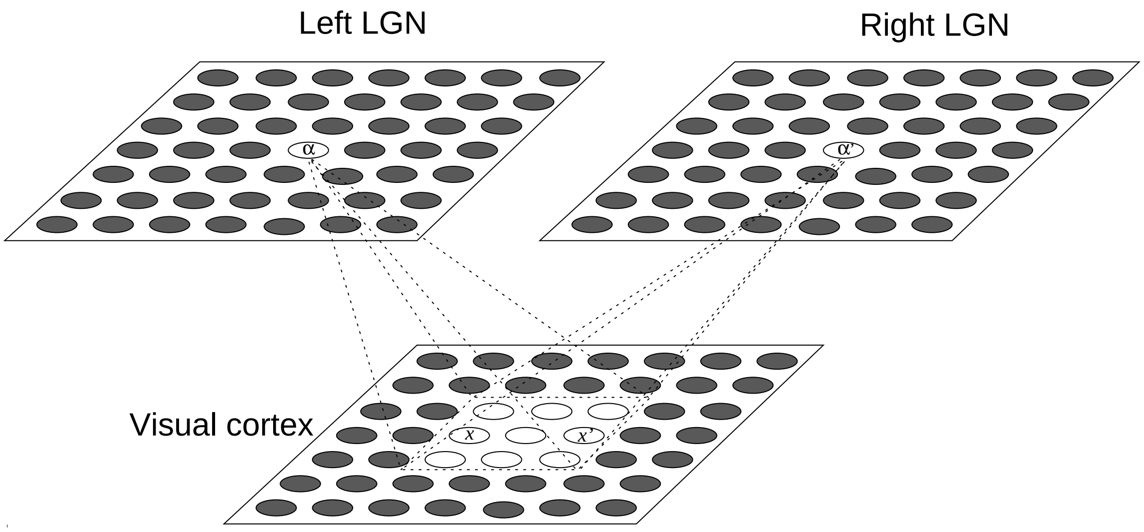

Figure 3–2: Miller's model representation, 7x7 square arrays for left and right LGN and visual cortex; in this example each LGN cell arborises to contact a 3x3 square of cortical cells.

The correlation in firing between the left- or right-eye input cells with retinotopic position α and the the left- or right-eye input cells with retinotopic position α', are described by the correlation function C (see figure 3–3).

In the figure 3–3 the elements of Miller's model are represented as follows (using the notations of [Miller 90]):

1. α and α' are LGN positions

2. x and x' are positions in the cortex retinotopically corresponding to α and α'.

3. LGN cells project synapses to several cortical cells with the retinotopically corresponding cortical position as the centre. In the figure the LGN cell and the cortical cells to which the synapses are projected are surrounded by dotted and dashed squares.

4. $A(y - \alpha)$ is given by the distance between y and the particular x that corresponding retinotopically to α. It is held fixed during the development of

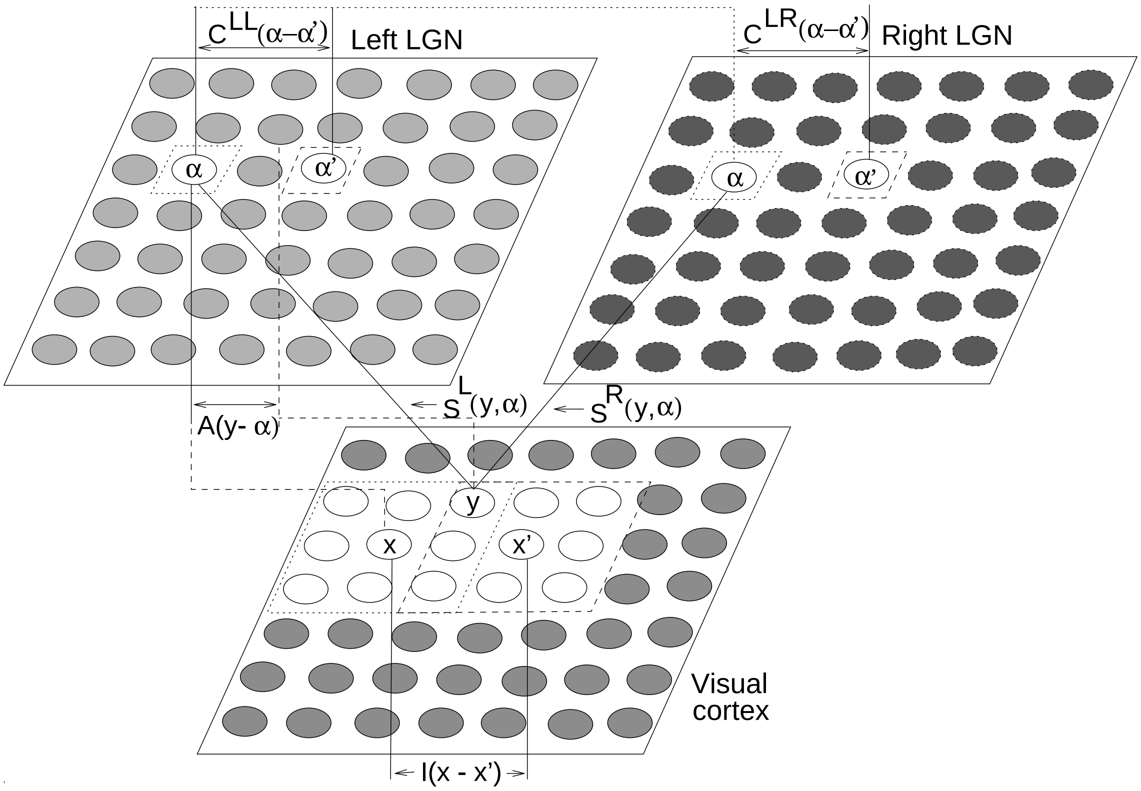

Figure 3–3: Representation of the elements of Miller's model (after Miller 1990).

the model. The number of synapses from α to the cortical cell position y is proportional to the arbor function $A(y - \alpha)$.

5. $S^L(y, \alpha)$ and $S^R(y, \alpha)$ are the total synaptic strength for the left and right eye respectively; they change during the development of the model.

6. $C^{LL}(\alpha - \alpha')$ gives the correlation between afferents α and α' both from the left eye.

7. $C^{LR}(\alpha - \alpha')$ gives the correlation between left-eye afferent α and right-eye afferent α'.

8. $I(x - x')$ gives the sign and strength of the interaction across the cortex. This influence consist in that the activation at point x' influences growth of synapses at the point x.

Miller's equation for cortical synaptic plasticity which expresses the changes in synaptic strengths with time, is as follows:

$$\frac{d(S^L(x,\alpha,t))}{dt} = \lambda A(x-\alpha)\sum_{y,\beta} I(x-y)[C^{LL}(\alpha-\beta)S^L(y,\beta,t) + C^{LR}(\alpha-\beta)S^R(y,\beta,t)] - DECAY^L(x,\alpha,t) \tag{3.5}$$

where S^{eye}(x,α,t) represents the total strength at time t in the left or right eye. The letters x, y represent positions for the cortical cells and the letters α, β for the LGN cells. $C^{eye-eye}(\alpha-\beta)$ is the correlation function. $I(x-y)$ represents the cortical interaction function, $A(x-\alpha)$ represents the arbor function. There is a similar equation for the right eye which just has L and R interchanged.

Miller made the following assumptions: each LGN cell arborised to a 7x7 square of cortical cells, using random values between 0.8 and 1.2 as initial strength on synapses (the inputs are assumed to be similar for both eyes). The synaptic strength was limited to a range between 0.0 and 8.0. Correlation between eyes was set to zero and constraints were applied to limit the total synaptic strength supported by an input cell or by a cortical cell.

Summary of Miller's Basic Simulation Results

In the results obtained with this model where the initial conditions were such that the cortical cells received inputs from both eyes, the topographic refinement effect consists in the concentration of the inputs from each eye on the central part of the receptive fields. The ocular dominance patches are formed as a consequence segregation of both eyes' synapses: inputs from one eye are very strong whereas inputs from the other are lost. These patches are produced because the segregation results in complementary regions driven by one eye or the other.

With this model, several features related to ocular dominance development were reproduced using different combinations of correlations, cortical interactions and arbor ranges, such as monocular deprivation, the critical period and strabismus. It has been shown that the afferent activity determines the degree of monocularity and the variation in intracortical interactions or arbor functions determine the stripe width [Miller *et al* 89].

Chapter 4

Model of Ocular Dominance Implementation

Miller's model, as described in Chapter 3 equation 3.5, was implemented in C and run on a Sun-4/25 SLC. The C language was used for its superior speed and space management efficiency and its numerical processing capability compared to other available alternatives (such as LISP or PROLOG). Because of the amount of processing required for a full test run for multiple cortical cells, the majority of testing was done on Sun 4/25s where a test could run for up to 15 hours. Runtime on Sun 4/20s could exceed 33 hours.

The results for the overall dominance values were plotted using a graphic package called xgraph in order to illustrate various points. To observe other properties such as topographic refinement some results were plotted using the package PRO - MATLAB ver 3.5. To observe the changes in all the units in the grid simultaneously for every ten iterations, a specific graphic program was implemented which used an Xwindows tool called Xwidgets.

4.1 Program Implementation: Representation with 4D and 2D Structures

As mentioned in the previous chapter, the arbor, interaction and correlation functions each have two associated indices. Each of these indices represents a cell body's position in the LGN or in the visual cortex. Cell layers (such as those in the LGN or visual cortex) can be viewed as two-dimensional matrices where each cell's position has an x and a y index. The above functions thus require a pair of arguments (arg1, arg2) where arg1 and arg2 are the x, y pairs of any two cells within the 2-D matrix. Therefore the function can be written as func$(x1, y1, x2, y2)$. In order to represent the necessary data structures to contain the values of the above functions, two alternative schemes were considered. These are:

1. The obvious choice was a four-dimensional structure. This 4D structure was implemented as a two-dimensional array structure where each component represented one of the two cell position indices (i.e. an X/Y pair) associated with the functions. Each of these indices are also two-dimensional, thus resulting in a 4D structure.

2. An alternative which actually was chosen for the final implementation of this program was a 2D array structure. This data structure was represented as a one-dimensional array. The component of this array was a two-dimensional index of as many cells as there were in the simulated sheets. For example an n x m simulated sheet is represented as a 1D array of (n x m) two-dimensional indices.

Both of these schemes were implemented and tested. Reasons and details are given below.

4.2 The 2D Structure Representation Version

The second of the above two representations (2D) was chosen for the model implementation. The motivation of this decision concerned the amount of memory needed to store (in 4D representation) the tables of gaussian distribution values for interaction and correlation functions[1]. These were calculated only once and then used in look-up tables for the calculation of the derivatives. The 2D version facilitates the representation of those tables in a way which reduced the amount of memory used to store them. This will be described in more detail in the *gauss* function description. The 4D method had the advantage of speed over the 2D method, but the cost in memory was too high for the available hardware. The disadvantage in speed of the 2D method stems from the need to calculate the interaction and correlation functions during the calculation of derivatives at each iteration. Despite this it solved the memory problem.

Each of the functions and data structures used in the implementation are discussed in the next section.

4.3 Functions and Data Structures

This section provides details of all the functions and data structures used in the implementation (see the source code in **??**, D, C and **??** for details).

4.3.1 *find_position* (generates an array of cell positions)

The two-dimensional indices representing the position of the cells located in the simulated cell sheets were used to store *x,y* coordinate values, i.e. the row and

[1] It was not mentioned until this point the that interaction and correlation functions are of the gaussian form.

column position in the cell sheets. The particular cell positions used to calculate the arbor, correlation and interaction functions during the model simulations were the same for each of the functions. For this reason a position table was generated to be consulted during any of the calculations of these functions. The *find_position* function was implemented to generate a table with all the possible positions in the grid. It was done first for the positions in the LGN then for the positions in the cortex. These are done separately because the LGN grids used a different size than the grid used for the cortex for some simulations. The generation of the position table was as follows:

A two-dimensional array was used as the data structure. The first index is a cell number from 0 to the total number of cells and the second index indicates whether the array element contains an x or y coordinate.

The data declaration is therefore:

$$position[SQR\_GRID][DIM]$$

where SQR_GRID is the total number of all possible two-dimensional cell positions in the grid and DIM is for the row (x) or column (y) coordinate. The result of the *mod* operator between the cell number and the total number of cells is assigned to the x coordinate element and the result of the div operator between them is assigned to the y coordinate element using the following equations, for each LGN cell first and then for each cortical cell:

$$position[cell\_number][x] = cell\_number \ mod \ CELL\_GRID$$
$$position[cell\_number][y] = cell\_number \ div \ CELL\_GRID$$

where $CELL\_GRID$ is the length of a side of the SQR_GRID.

4.3.2 *gauss* (generates look-up tables of gaussian values)

In order to calculate the correlation function, i.e. the correlation between afferents with retinotopic positions, tables of gaussian values of the distances between afferents were generated. Different factors (widths) were used in the gaussian calculation for each correlation function and for the interaction function. This allowed the variation of these factors during parameter space exploration.

The gaussian calculation was done in this 2D representation using the following equation:

$$e^{-x^2/spread\_factor^2} * e^{-y^2/spread\_factor^2} \tag{4.1}$$

Which is equivalent to that used in the old 4D version:

$$e^{-(x^2+y^2)/(spread\_factor^2)} \tag{4.2}$$

the gaussian values calculation given by the formula 4.1, is represented in the code as follows:

$$exp^{(-(dist)^2/factor^2))} \tag{4.3}$$

Where the term $spread\_factor^2$ is the standard deviation width.

Looping over LGN cells using distances between the cells, the gaussian values calculated were stored in a one-dimensional array with an index which represents distance between cell positions in the LGN or cortical sheets.

The gaussian values were calculated for the following four correlation arrays (see figure 4–1):

- CorLL[dist] for distances between cells in the left LGN.
- CorRR[dist] for distances between cells in the right LGN.
- CorLR[dist] for distances between left and right LGN cells.
- CorRL[dist] for distances between right and left LGN cells.

where *dist* is the distance between cells.

The influence of cortical cells upon one another's growth was denoted by the interaction function. gaussian values for this interaction function were generated and the values were stored in an array

$$Inter[dist]$$

$$\left(\begin{array}{c:c} C^{LL} & C^{LR} \\ \hdashline C^{RL} & C^{RR} \end{array}\right)$$

Figure 4–1: Correlation within and between eyes.

4.3.3 *CorreInt* (computes correlations and interactions)

This function is used to calculate both correlation function and interaction functions. This function has three arguments; a and b are the numbers of the two cells involved and *funct* is the type of function to be calculated (i.e. correlation function (LL, RR, LR, RL) or interaction function).

Then given a pair of cells involved in the function:

1. The x,y coordinate positions are consulted and the absolute value of the difference between their x's and y's are assigned to the variables *xdiff* and *ydiff* respectively.

2. In a case statement the gaussian values for their cells' distance is selected from their respective array and is returned for use in the main equation at each time step.

4.3.4 *arborF* (generates a look-up table for the arbor function)

The arbor function synapses between LGN and cortex were denoted by an array of size SQR_GRID.

$$arbor[SQR\_GRID]$$

In this implementation a flat arbor function with a value of 1 is used. The function is a simple loop for all the LGN cells assigning a value of 1 to all the elements in the arbor array.

4.3.5 *build_weight_matrix* (initialises the weight matrix)

This function is used to generate initial small random values and initialise the weights array with them. It has two subroutines: the first one generates the random values, within a given range. In the second routine which calls the first one, the random values are assigned to the weights array for left and right eye using double loop. The weights for each cell in the cortex or LGN were stored in a three-dimensional array where the first component represented the left or right eyes (EYE), the second is the cell position in the LGN (SQR_LGN) and the third is the total cell position in the cortex (SQR_CX).

The data declaration is thus:

$$weight\_matrix[EYE][SQR\_LGN][SQR\_CX]$$

4.3.6 *weight_changes* (calculates the derivatives)

In this function Miller's model equation 3.5 is calculated as follows:

Looping over each cortical cell and for each LGN cell the following calculations were done:

1. The right side of equation 3.5 is calculated and accumulated in the *changesl* and *changesr* variables in the following way:

$$changesl+ = I(x,y) * (C^{LL} * S^{L}(b,y) + C^{LR} * S^{R}(b,y)) \tag{4.4}$$

$$changesr+ = I(x,y) * (C^{RR} * S^{R}(b,y) + C^{RL} * S^{L}(b,y)) \tag{4.5}$$

2. The parameter λ multiplied by the arbor function then it is multiplied by the result of the calculation from 1.

3. The decay terms are subtracted from the results so far for left and right eyes, i.e. γ, which is the decay term associated with the total strength, minus ϵ, which is the decay term associated with the arbor function, is subtracted from *changesl* and *changesr*.

4. The result obtained from subtracting the decay terms is assigned to a temporary weight array:

$$left\_changes = changesl - (\gamma * S^L(LGN\_cell, cx\_cell) - \epsilon * A(cx\_cell, LGN\_cell) \tag{4.6}$$

$$right\_changes = changesl - (\gamma * S^R(LGN\_cell, cx\_cell) - \epsilon * A(cx\_cell, LGN\_cell) \tag{4.7}$$

   which is the further processed by the function constraints.

4.3.7 *constraints* (limits the total synaptic strength)

This function was implemented to limit the total synaptic strength by applying constraints as follows:

1. Looping over each cortical cell and looping over each LGN cell for each eye

$$sum\_left = sum\_left + changesl$$
$$sum\_right = sum\_right + changesr$$

   Total changes for left and right eyes calculated:

$$sum\_changes = sum\_left + sum\_right$$

2. Looping over each cortical cell for each eye, subtracting:

$$temporary\_array = temporary\_array - (sum\_changes/(SQR\_LGN * 2))$$

3. And looping then over each LGN cell

$$sum\_left = sum\_left + changesl$$
$$sum\_right = sum\_right + changesr$$

   looping over each cortical cell

$$temporary\_array = temporary\_array - (sum\_changes/SQR\_LGN)$$

4.3.8 *updating_weights* (updates values bounded in a range)

In this function the weights are limited to a pre-determined range. Looping for each cortical cell and for each LGN cell, values are bounded between the upper and lower limits declared i.e. weights above or below limits are set equal to those maximum and minimum values. Then the weights are updated for both eyes.

4.3.9 *storing_weights* (stores in a file the updated weights)

In a double loop for each cortical cell and for each LGN cell the updated weights are stored in a binary file.

4.3.10 Simulations Steps

The simulation steps were as follows. For each eye:

1. the generation of the cell position table, the gaussian values tables (to be used in the correlation and Interaction functions), a flat arbor function table, and weight initialisation using small random values were calculated only once.

2. Using the information calculated in the previous iteration, at each time step and for each synapse the derivative described in equation 3.5 was calculated. As specified by [Miller *et al* 89, footnote 23], the DECAY terms were usually set to 0 and λ was set to 0.008. It was necessary to use a big enough value for λ to observe the development of the model without affecting the results of exploring different parameters. Some positive DECAY values were tried as well.

3. Constraints were applied first to each cortical cell and then to each LGN cell as follows [Miller 1989, footnote 23]. The sum of the changes for each cortical cell stored, and the changes then adjusted by subtracting from each a fraction of the total change, giving a total zero-change weight, limiting in this way the total synaptic strength supported. The same process was applied for LGN cells.

4. To update the weights, the changes made were added to the old weights limiting the values to a range between 0 and 8, as was the case in Miller's simulations.

5. The updated weights were stored in a file every 10 iterations. The weight values stored were used to generate graphic representations (described below) and analyses, each 10 time steps, to show how they developed from 0 to 200 iterations.

4.4 Total Dominance Program

The overall dominance values were calculated using a small program C. A method was chosen to calculate these values which was based on the following expression:

$$Overall\_dominance = \frac{\sum_{right} weights - \sum_{left} weights}{\sum_{right} weights + \sum_{left} weights} \tag{4.8}$$

This program has two functions:

1. The first function reads the data values stored in a file every ten time steps during the execution of Miller's model program.

2. The dominance function was implemented by looping over each cortical cell calculating the overall dominance value as the difference between the total (summed) strength of right-eye and of left-eye geniculate inputs to the cell. Normalisation was included in this implementation in order to keep all the values in the range 0-1. Then the absolute values of dominance from all cortical cells were averaged together.

4.5 Graphic Display

Visualisation and analysis of results are difficult when the amount of data is too big, as is the case here. In order to have a graphic display of the results, graphic tools such as PRO - MATLAB ver 3.5 and xgraph for Xwindows were used to display partial results such as dominance values and some results from single cortical cell simulations.

It was necessary to have a global visual result display at different time steps, with data obtained during the simulation in order to understand how the process developed. For this reason a graphic display program was implemented which used input data from the files in which the results were stored in the calculation program. The program was written with an Xwindows tool (Xwidget), which allows a graphic display to be produced in a Xwindows environment.

4.5.1 Graphic Program Implementation

The program generates a interactive graphic window interface. The sizes of the window and control panel are defined in the declaration section as variables. It includes the global declaration file used in the main (Miller) program, since some of the global variables such as the weight array, are shared. The main functions are described below; see figure 4–2.

Selection Function

The widget tool has the facility to generate menus. A menu with the control panel options was generated allowing any of the following functions to be selected:

- Quit
- Get data
- Left LGN

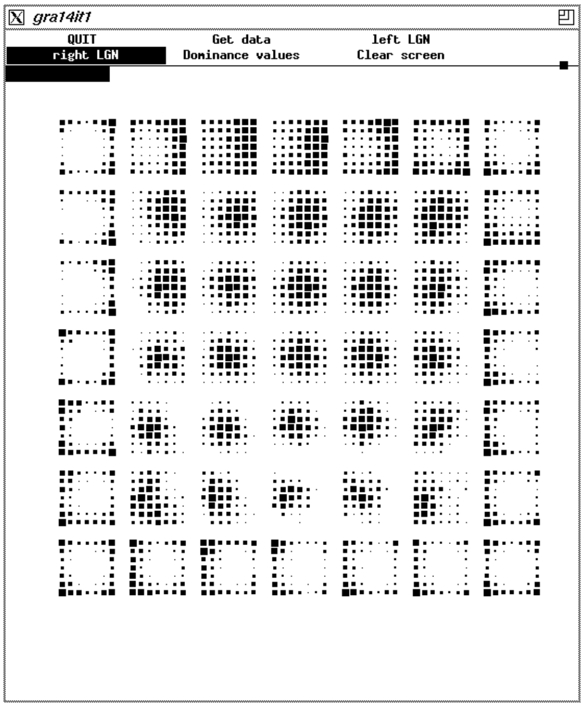

Figure 4–2: Graphic program panel, showing the weights of the right LGN for the control at 20 iterations.

- Right LGN
- Dominance values
- Clear screen

Quit

In this the data file in use is closed and the exit command to finish the program is executed.

Get Data

With this function reads from the data file the data values stored every ten iterations during the main (Miller) program execution. This is done looping for all LGN cells and all cortical cells.

Draw weights

1. In this function the sizes of the boxes used to display each cortical representation are initialised according to the total panel size declared.

2. A selection between *left LGN*, *right LGN* or *dominance values* is allowed, and the appropriate function is called with the parameters involved. This is implemented within a loop. The display shows the weight values for each cortical cell, with inputs from the left LGN, or the right LGN depending which of these options is selected. The values from both inputs are displayed if *dominance values* is selected (see examples in figures 6–5 and 6–6).

3. For the left or right LGN a function is shared but the foreground and the background for the cortical cells are selected in inverse combination for each i.e. black background and white foreground for the left LGN and inverse for the right one. In this way it is possible to distinguish easily between the values from one eye and from the other.

4. In a loop the maximum weight value is found.

5. In each element of the matrices squares are plotted whose size is proportional to each cell weight taking in account the maximum weight value in each matrix (i.e. the largest weight value has a square of maximum size).

Draw dominance

This function displays simultaneously the final distribution of dominance values from left and right inputs.

Chapter 5

Simulation Results: One cortical cell

The exploration of the parameter space was done first for one cortical cell to do a complete parameter exploration in a reasonable amount of time. Then an exploration of the more important parameters using multiple cortical cell. The exploration in the second set of simulation was selecting the more important parameters according with the goals of this project, given the average time of execution (15 hours) per simulation. In this chapter the results from the simulations with one cortical cell exploring the parameter space, are described. A grid of 7x7 cells in each LGN was used. The parameter space was investigated by varying parameters such as the step size of development λ, initial weights values, decay terms (γ and ϵ), number of input cells, the width of the correlation function within eyes, and the width and strength of the correlation function between eyes. The results of the parameter space exploration will be analysed in Chapter 7. The parameters explored can be classified into two main categories, according to the type of properties which they affect.

- *General Type* - These parameters did not affect the properties of the model behaviour, but it was important in seeing how they affected the program performance (e.g. the program speed). Also, it was necessary investigate those parameters which affected in some way properties in the model, but which were not central to the purpose of this study, such as λ and the decay terms.

Parameter	*value*
LGN matrix	7x7
CX matrix	1x1
initial weight values range	0.8-1.2
synaptic strength range	0-8
decay terms γ/ϵ	0/0
λ	0.008
correlation widths:	
within LL	2.8
within RR	2.8
strength between eyes	0.0

Table 5–1: Initial parameter values.

- *Central Type* - This other type involved parameters which were crucial because they affected the model directly in ways of interest by influencing different properties, such as ocular dominance and topographic refinement.

5.1 Varying General Parameters

The initial set of parameters from which the exploration of the parameter space started is shown in table 5–1. Using these parameters a control simulation was plotted at different iterations to observe the model behaviour under these conditions, as shown in figure 5–1. In this figure x and y axis represent the number of cells per side in the sheets, and z represents the weight value en each cell. In the graphs the weight values for each cell are plotted at 0, 10, 20, 40, 70 and 140 iterations. The scale for z in this figure is in a range of 0.8-1.2 for 0 and 10 iterations and of 0.0-8.0 for 20, 40, 70 and 140 the scale, (these graphs were plotted using Mat-Lab tool). Inputs from both eyes are plotted together for each graph, by placing the weight matrices from left and right inputs together.

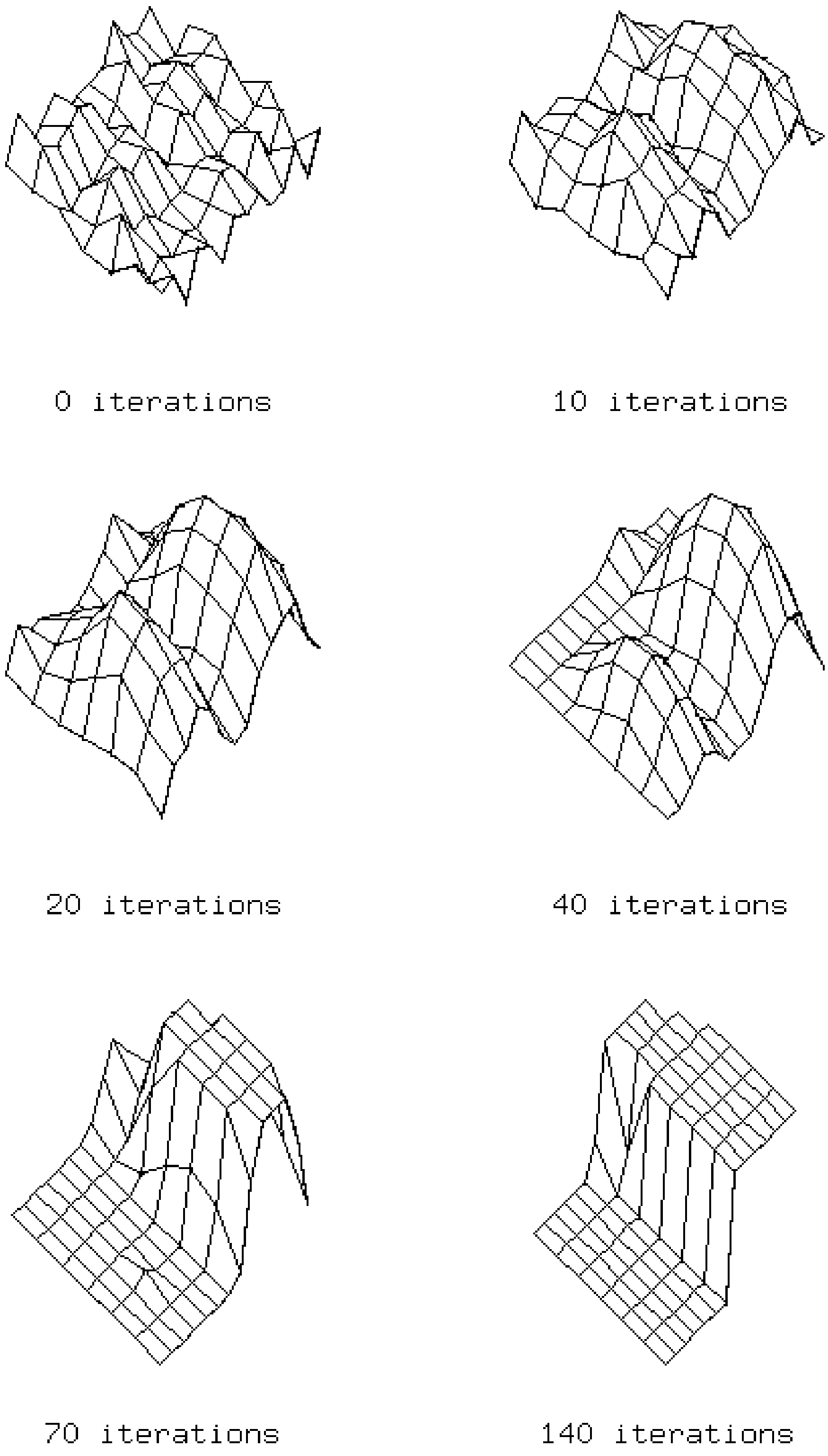

Figure 5–1: Simulation of one cortical cell control case, showing the changes from both left and right inputs after different numbers of iterations.

number	*ranges of values*
1	0.0 - 1.0
2	0.5 - 1.5
3	0.8 - 1.2
4	0.8 - 2.0
5	0.8 - 2.5

Table 5–2: Different range of initial values tested.

Program speed

For the case of one cortical cell the average speed of execution was tested using the initial parameters listed in table 5–1. The process processing time average for this set of cases was 45 seconds for 200 iterations.

Lambda

The step size of the development in the model is given by λ. The λ values tested in the range of 0.003-0.015, ([Miller *et al* 89, footnote 23]) it was observed that the speed which dominance was achieved increased as the λ value increased. Taking this into account, we were interested in a value which would allow us to observe the development of the model behaviour without qualitatively changing this behaviour under different conditions during the exploration of other parameters. The λ value chosen which satisfied this requirement was 0.008, and it was used in all the following simulations.

5.1.1 Initial Range of Weights

In this set of simulations, the aim was to find out how results were affected using different ranges of initial small random weight values. Based on the range used by Miller (0.8-1.2) ranges smaller and bigger than his were used to be able to analyse the effects in both directions. The simulations here were done using the values in table 5–2.

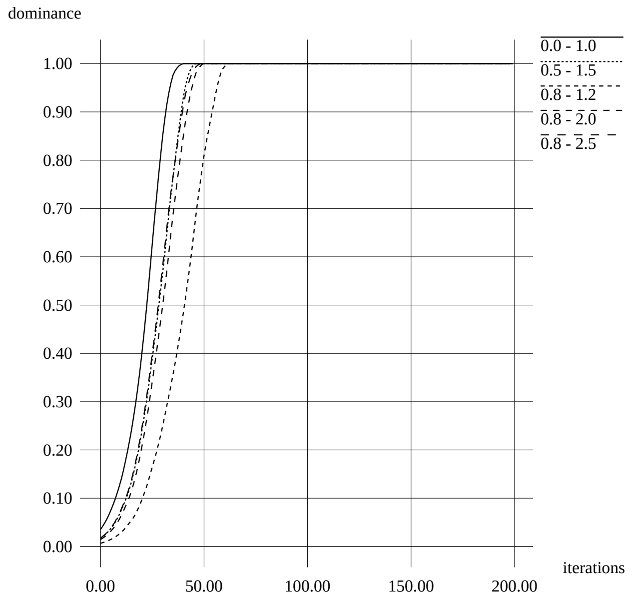

Figure 5–2: Effect of varying the initial weight range using different range sizes.

number	*ranges of values*
1	0.0 - 1.0
2	0.0 - 2.0
3	0.0 - 3.0
4	0.0 - 4.0
5	0.0 - 5.0
6	0.0 - 6.0
7	0.0 - 7.0
8	0.0 - 8.0

Table 5–3: Increasing the initial weight range by one.

In fig. 5–2 we have the following results:

- Overall dominance was reached (1.0) in range 0.5-1.5 (number 2 in table 5–2) at 62 iterations and,

- dominance was reached for the other ranges before 50 iterations with a very similar speed as for the the ranges number 2, 3 and 5 in the same table and slightly slower for the range 1 (0-1).

A second set of simulations was tried in which the initial weight values were varied maintaining a fixed lower limit of 0 and increasing the higher limit by one each time from 1 to 8 as is shown in table 5–3. The results (plotted in figure 5–3) are as follows; the speed to reach dominance decreased as the range of initial values increased.

Another set of simulations was carried out in which the initial weight values were varied. A range of values with a fixed difference of 0.4 between the lower and higher value was tried, and the position of the simulated window was varied from 0 to 8 as is shown in table 5–4.

The results were as follows:

- If the window was in a position higher than 1.8 - 2.2 dominance was not reached.

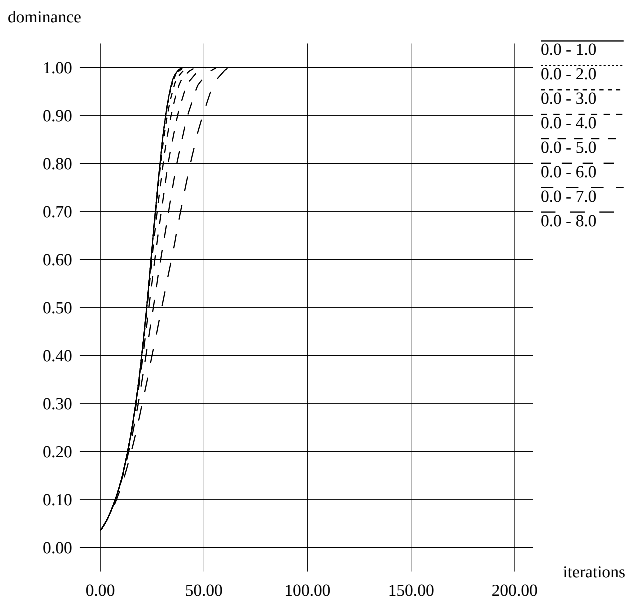

Figure 5–3: Effect of varying the initial weight range, incrementing the range by one with a fixed lower limit.

number	*ranges of values*
1	0.0 - 0.4
2	0.8 - 1.2
3	1.8 - 2.2
4	2.8 - 3.2
5	3.8 - 4.2
6	4.8 - 4.2
7	5.8 - 6.2
8	6.8 - 7.2
9	7.6 - 8.0

Table 5–4: Varying the initial weight values within a small window of 0.4.

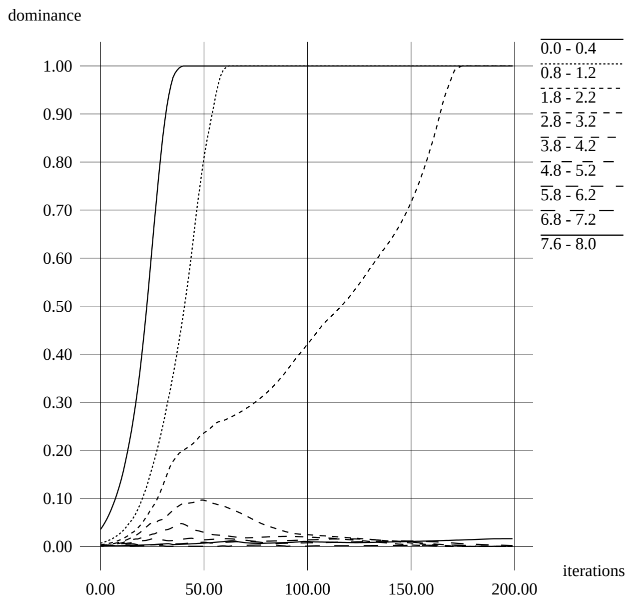

Figure 5–4: Effect of a range window of fixed width = 0.4 moving from 0 to 8.

- As the values in the window increased above this range, the overall dominance values tended to zero as shown in figure 5–4.

- In the first three cases in table 5–4 dominance was reached, and as the window was moved to a higher range, the speed at in which dominance was reached decreased.

5.1.2 Decay Terms

We explored the effect produced by varying the decay terms:

- First the ϵ decay term was set to 0.01, 0.05 and 0.1. No variation in the results was observed.

- For the three cases the speed at which dominance was reached was similar to that observed with $\epsilon = 0$.

- This term is associated with the arbor function which, as mentioned in Chapter 4, was a flat function set to 1.

- The width of the within-eye correlation function was fixed to 2.8 (for both RR LL) and between-eye to 0 (LR RL).

For variations in γ, the dominance curve behaviour was different:

- Using the same range of values as ϵ, the speed to reach overall dominance decreased as the γ value increased.

- A big difference in speed for γ values of 0.05 and 0.1 was observed. The results are plotted (dominance vs iterations) in figure 5–5.

- Using the same values as above, for both terms, ϵ and γ simultaneously, the results obtained were the same as those shown in γ figure 5–5.

5.2 Varying central parameters

The results of varying the crucial parameters affecting the behaviour of Miller's model are described in this section. In order to compare and analyse the results as proposed in the project objectives, we did some simulations that were similar to those of [Miller *et al* 89] and others which followed the suggestions arising from the theoretical predictions of [Dayan & Goodhill 92, Goodhill 91].

5.2.1 Correlations

Within eyes

Miller studied ocular dominance stripe development using different widths in the gaussian correlation functions, he chose narrower and broader correlations width

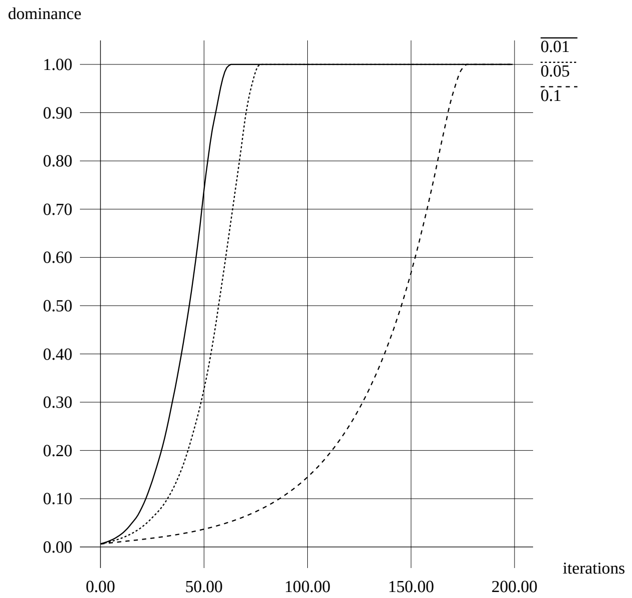

Figure 5–5: Influence of using different γ values, decay term associated with total strength.

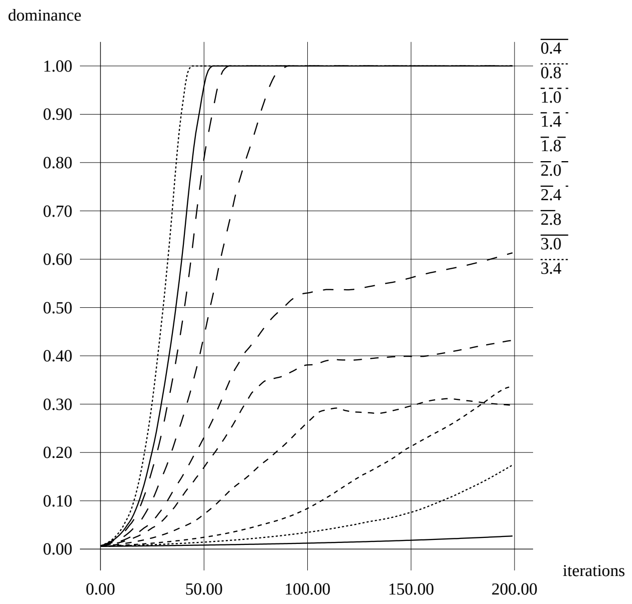

Figure 5–6: Effect of using different width of correlation within eyes (RR LL).

values 1.4 and 2.8. In this project a wider range of correlation width values was used in the simulations so we could observe how they affect the results and compare our results with those obtained by Miller and with the theoretical predictions.

The width for correlation within eyes was varied within to a range of values 0.4-3.4, and using zero correlation between eyes the simulation results are plotted in figure 5–6 (dominance against iterations).

In figure 5–6 it is possible to observe that changing the correlation values changes not only the speed at which dominance is reached but in some cases also leads to binocular solutions. The Dominance was reached for values of 2.4, 2.8, 3.0 and 3.4: it was reached faster as the value was increased. For the values smaller than 2.4 dominance was not reached and the maximum values obtained for overall dominance decreased as the correlation width decreased, leading to binocular solutions.

between-eye correlation width	*strength values*
2.8	0.3
2.8	0.4
2.8	0.5
2*2.8	0.3
2*2.8	0.4
2*2.8	0.5
3*2.8	0.3
3*2.8	0.4
3*2.8	0.5

Table 5–5: Varying width and strength of between-eye correlations.

Between eyes

In Miller's simulations it was assumed that the correlation between eyes was zero or negative; he did no do simulations using positive correlation between eyes. Since, as discussed by [Dayan & Goodhill 92, Goodhill, 91], it is biologically implausible that there are no correlations between eyes, one of the main objectives in this project was to explore Miller's model using a larger number of values of correlations both within and between eyes, and to compare the results with those predicted by [Dayan & Goodhill 92, Goodhill 91].

In this set of simulations a fixed within-eye correlation (w-e-c) of 2.8 was used. The correlation width between eyes was changed using as a base correlation within eyes of 2.8; values of once, twice and three times this width were used (2.8, $2*2.8$, $3*2.8$). For each of these between-eye widths three sets of simulations were done, varying the strength value of the correlation function between eyes to 0.3, 0.4 and 0.5. as in table 5–5.

The results are plotted as follows:

- Using a between-eye correlation width (b-e-c) of 1*w-e-c (2.8), dominance was reached only for a strength value of 0.3. The speed at which dominance was

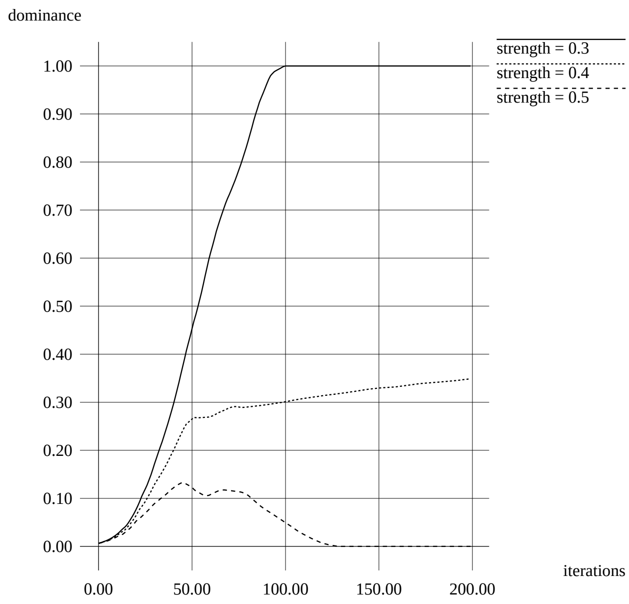

Figure 5–7: Effect of varying strength values of between-eye correlations with a fixed width of 2.8.

reached was slower than with zero correlation between eyes. The results of these simulations are plotted in figure 5–7.

- In the case of b-e-c of 2*w-e-c (5.6), dominance was reached only for a strength value of 0.3 and at a slower speed than for the case of 1*w-e-c these results are plotted in figure 5–8.

- Finally for 3*w-e-c (8.4) for, dominance was not reached for any of the cases as shown in figure 5–9.

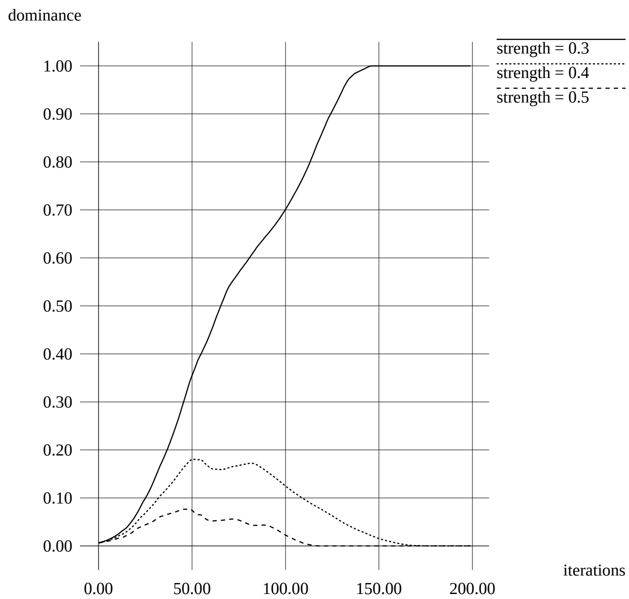

Figure 5–8: Effect of varying strength values of between-eye correlations with a fixed width of 2*2.8.

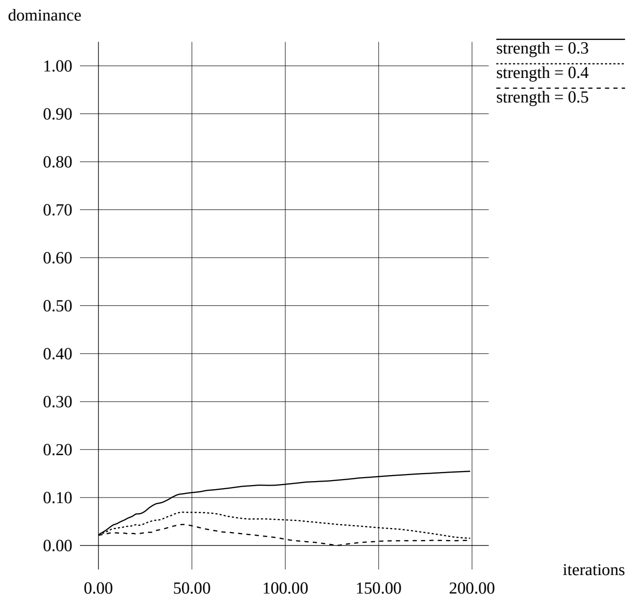

Figure 5–9: Effect of varying strength values of between-eye correlations with a fixed width of 3*2.8.

5.2.2 Number of Input Cells

It was suggested in [Goodhill 91] that binocular solutions start to catch up with monocular solutions as the number of cells increases, i.e. dominance is not reached. For this reason it was part of the aim of in this project to explore empirically the effects of different sizes of grids in the simulation results.

The sizes used for these simulations varied from 5x5 to 17x17 increasing, in steps of by 2 (odd numbers only). This range was chosen because it included a reasonable number of different sizes which took a reasonable program speed during the simulations. Here the width of the correlation function within eyes was constant at 2.8, and between-eye correlations was zero.

The results, plotted in figure 5–10, show that dominance was reached using grid sizes 5x5 and 7x7 only. In the case of a 9x9 grid, dominance was not reached but it can be observed that the maximum dominance value did not decrease to the extent it did for bigger sizes. On the other hand for sizes bigger than 9x9, binocular solutions were obtained, i.e. a dominance value of 1.0 was not obtained.

The following sets of simulations were done in order to see if the effect produced by incrementing the grid size could be changed by increasing the width of the within-eye correlation function in proportion to the grid size.

These sets were chosen on correlations with based on the minimum correlation width value at which 7x7 grid reached dominance (2.4). This width was decreased and increased in proportion to the decrement or increment in the grid size. The values produced by this adjustment are shown in table 5–6. Dominance was reached for all grid sizes. The speed at which it was reached was different for each size: it was reached faster as the grid size increased, these results are shown in figure 5–11.

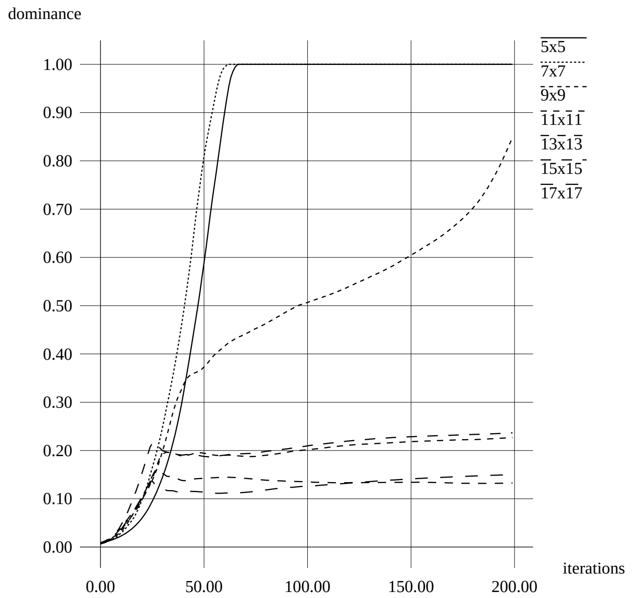

Figure 5–10: Effect of different input grid sizes in the simulations.

size	*correlation width*
5x5	1.7
7x7	2.4
9x9	3.1
11x11	3.8
13x13	4.5
15x15	5.2
17x17	5.9

Table 5–6: Using correlation proportional to grid size.

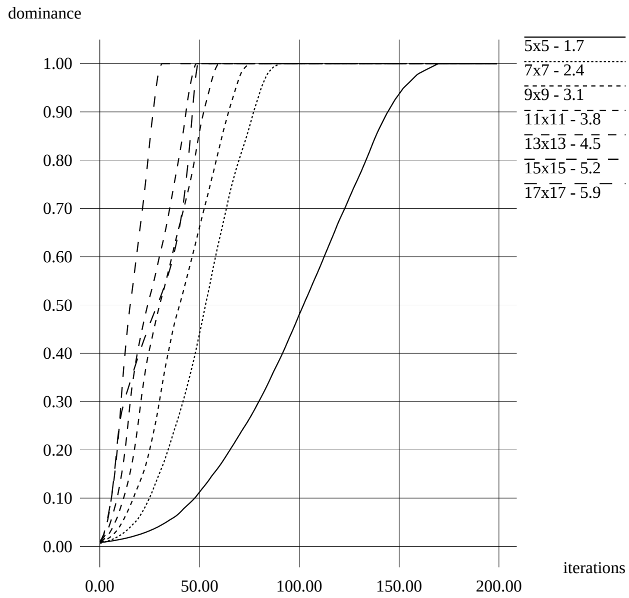

Figure 5–11: Effects of correlation width adjusted proportionally to grid sizes

Chapter 6

Simulation Results: Multiple Cortical Cells

In this chapter the results from the simulation set using multiple cortical cells under parameter variation are described.

6.1 Varying General Parameters

6.1.1 Grid size and speed

According to the parameter classification described in last chapter, the only general parameter tested for the multiple cortical cell case was the grid size, in order to observe how it affected program speed. Simulations were done using the parameters listed in table 6–2, (which are similar to those Miller used), except that the grid sizes were changed for both LGN and cortical representations equally; the sizes tried were 5x5, 6x6 and 7x7 for 200 iterations. The main objective of varying this parameter was to find the largest number of cells that could run in a reasonable period of time, i.e. less than 24 hours. The results are presented in table 6–1; in figure 6–1 the grid size represent the number of elements per side (x, y) for both LGN and cortical grids, z represents the weight value for each element.

grid size	*time in minutes*
5x5	93
6x6	301
7x7	1286

Table 6–1: Effect of varying grid size on the program speed.

Parameter	*value*
LGN matrix	7x7
CX matrix	7x7
initial weight values range	0.8-1.2
synaptic strength range	0-8
decay terms γ/ϵ	0/0
λ	0.008
correlation widths:	
within LL	2.8
within RR	2.8
strength between eyes	0.0

Table 6–2: Control set.

6.1.2 A Control Case

To analyse the effect of varying the parameters it was necessary to have a reference control case; we set this up using parameters similar to those Miller used (see table 6–2).

The simulation results are shown in figure 6–1 for different time steps; these represent the total strength of right and left eye inputs to the central cell in the cortical sheet. The scale on this figure is in a range of 0.8-1.2 for 0 and 10 iterations and of 0.0-8.0 for 20, 40, 70 and 140. As in the control case in chapter 5 these figures show the values from both inputs simultaneously in order to compare between them. In order to plot them left and right eye inputs together, the weight values matrix of left is placed in a file with the weight values matrix of right input, then as only one matrix it is plotted (using Matlab tool). Half part of each graph represent then each eye input. The result plotted are form the central cortical cell at 0, 10, 20, 40, 70 and 140 iterations, the results are similar to those plotted in the one cortical cell control case.

The results were displayed in a matrix of all cortical cells using the graphic program, in this graphs each square in the matrices consist of total number of cells (49) inputs from left and right LGN separately. In each element of the matrices the size of the squares is proportional to the maximum weight value in each matrix (white squares for left eye inputs and black for right eye inputs.

In this way the changes for each cortical cell from left and right eye inputs were compared at each time step. For the control simulation the graphics obtained were as follows:

- In figure 6–2 the values at 0 iterations are small random values for both cases.

- At 20 iterations (figure 6–3) it is possible to observe topographic refinement in some units from both left and right inputs.

- In the results at 40 iterations some monocular solutions are shown: when a unit becomes monocular an empty space is left in the retinotopic 0position in the other input (figure 6–4).

- At 70 iterations almost all the units are monocular as observed in figure 6–5.

- Finally at 140 iterations complete dominance is reached.

The results are plotted together in the same panel with the dominance option, see figure 6–6. An interesting comparison can be made between these figures of the graphic program and those which are shown in figure 6–1 since both are plotted from the same set of data values. In figure 6–6 the total dominance reached is shown.

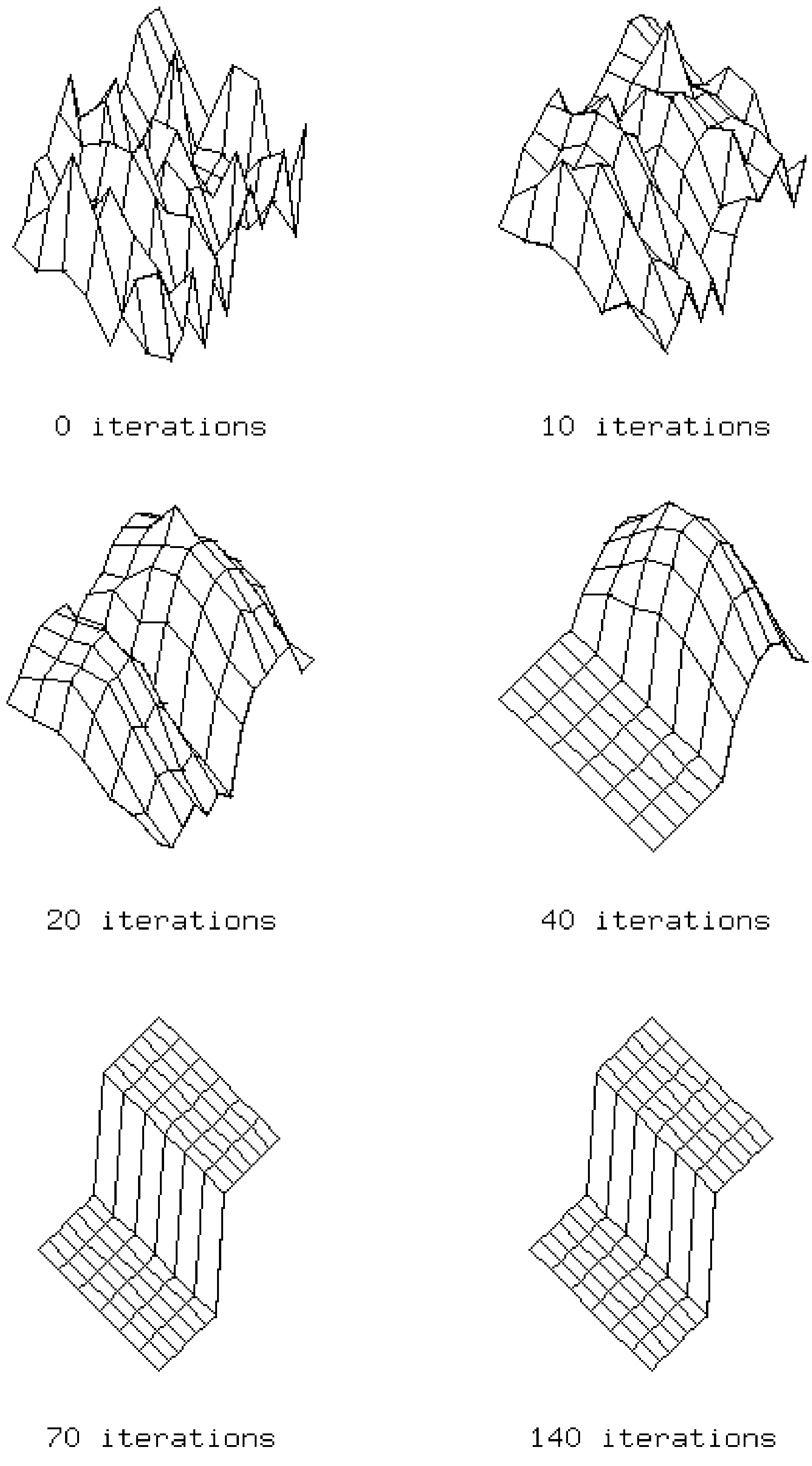

Figure 6–1: Simulation of a control case, showing the changes in the weight values, from both left and right inputs after different number of iterations.

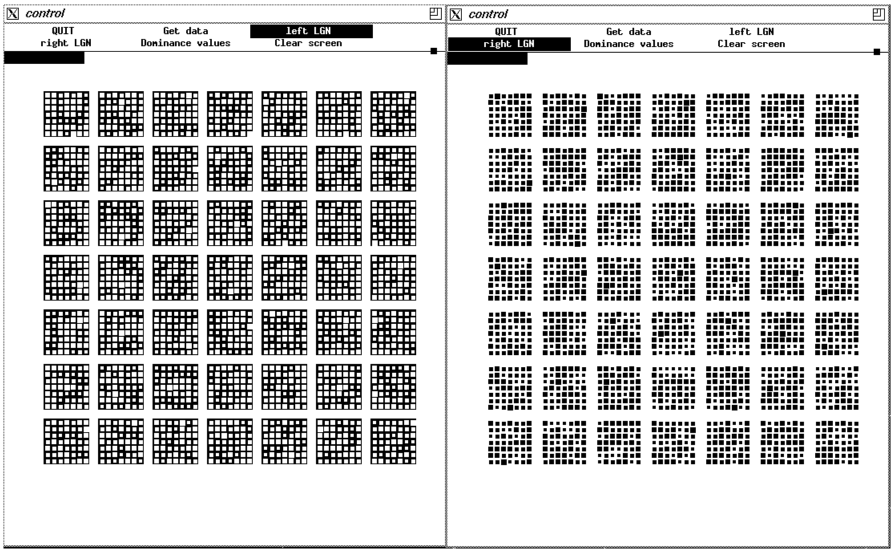

Figure 6–2: Graphic results of a control case in the whole layer at 0 iterations, small random values.

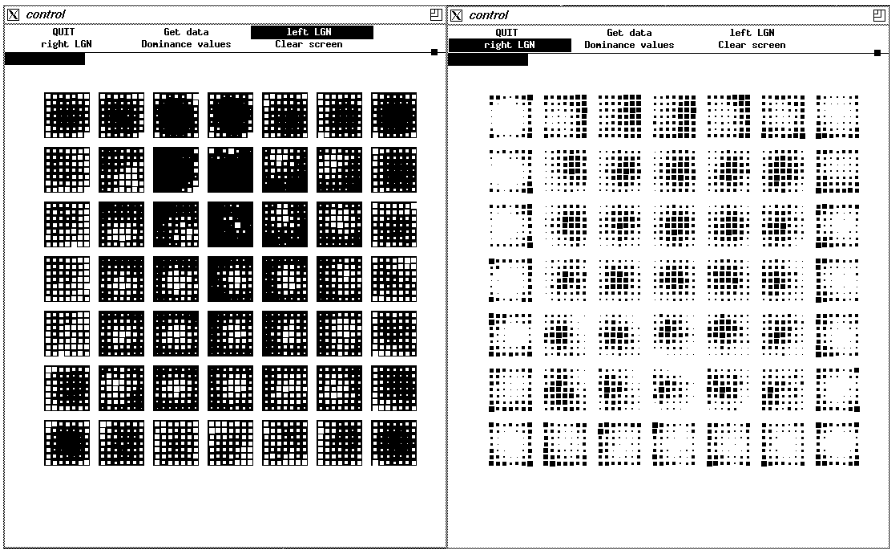

Figure 6–3: Graphic results of a control case in the whole layer at 20 iterations.

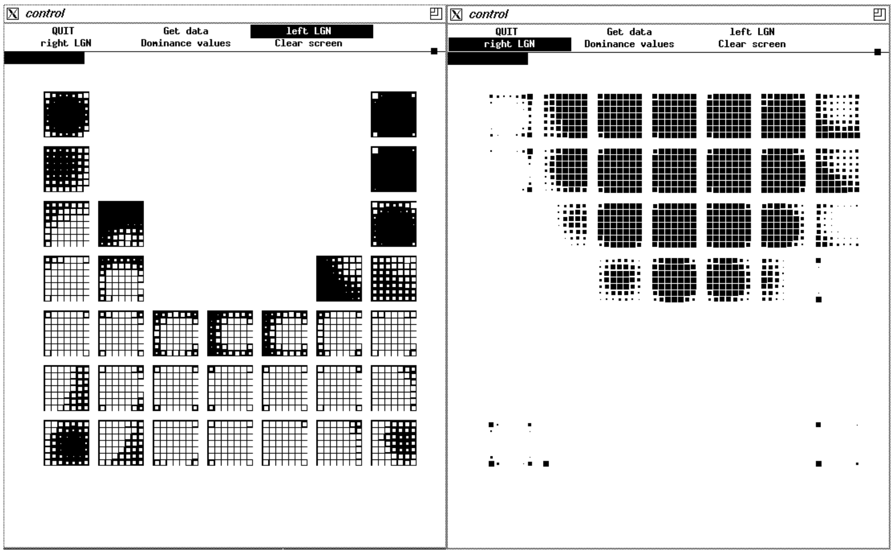

Figure 6–4: Graphic results of a control case in the whole layer at 40 iterations.

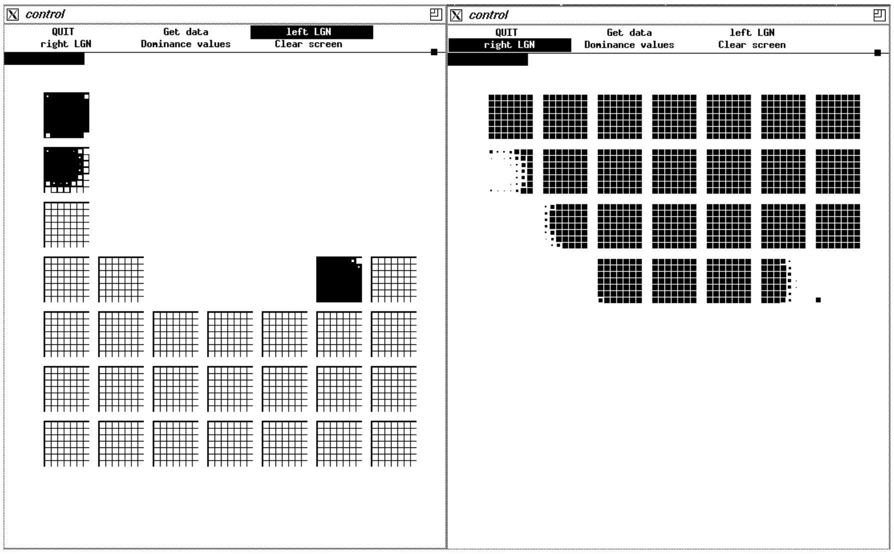

Figure 6–5: Graphic results of a control case in the whole layer at 70 iterations.

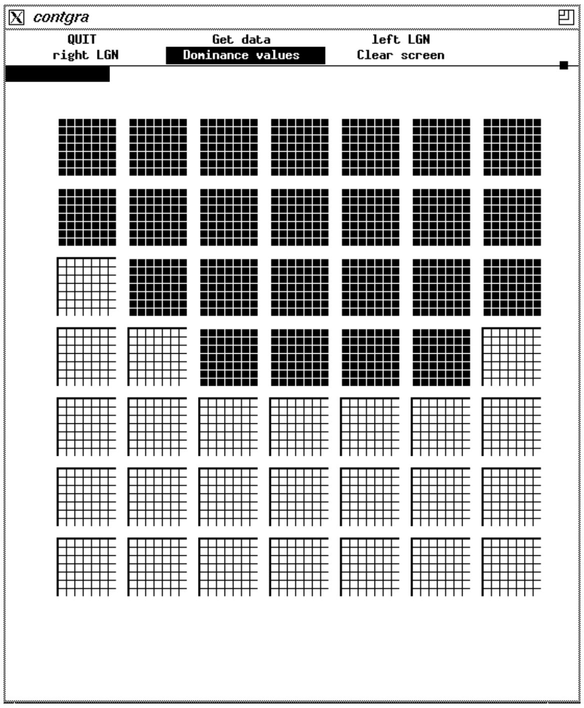

Figure 6–6: Graphic results of a control case in the whole layer at 140 iterations.

6.2 Varying Central Parameters

6.2.1 Positive Between-eye Correlations

This set of simulations is one of the most important in this project since the principal goal of the project was to investigate the effect of more realistic inputs to Miller's model. Positive correlations between eyes are more realistic correlations than the zero between-eye correlation assumed by Miller which, as mentioned, seems to be biologically implausible.

In this section we considered how the development of dominance was affected by varying the strength of the between-eye correlation function. The conditions used in this set of simulations were correlation width within eyes of 2.8 and correlation width between eyes of to 2*2.8. The values of strength used were 0.3, 0.4 and 0.5.

The results obtained were the following:

- In all three cases dominance was not reached. Here it is important to say that in the case using a strength factor of 0.3 a maximum value near to 1 was reached, through this value did not remain stable, but showed an increasing tendency (figure 6–7).

- For the cases of between-eye correlations strength values of 0.4 and 0.5 dominance was not reached. The maximum value reached in the 0.4 case is roughly halfway between 0 and 1 and it is not possible to observe a well defined tendency in the curve. In the case of 0.5 the curve tends to zero. Then these last two simulations lead to binocular solutions. The graphic visualisation of these binocular solutions can be observed in figure 6–8.

6.2.2 Varying Interaction Width

To explore the effect of varying interaction width two set of simulations were made:

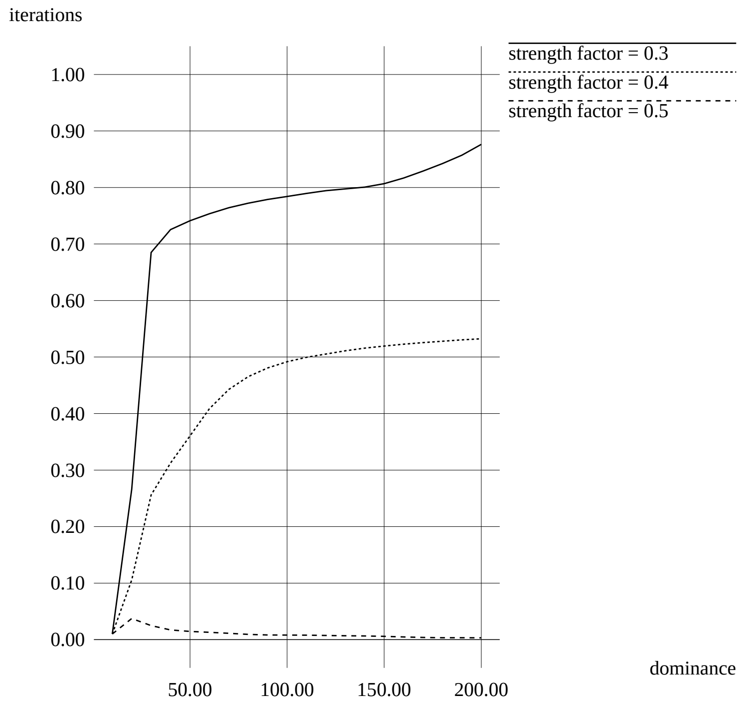

Figure 6–7: Different strength values. Fixed correlation width within eyes of 2.8 and between eyes of 2*2.8

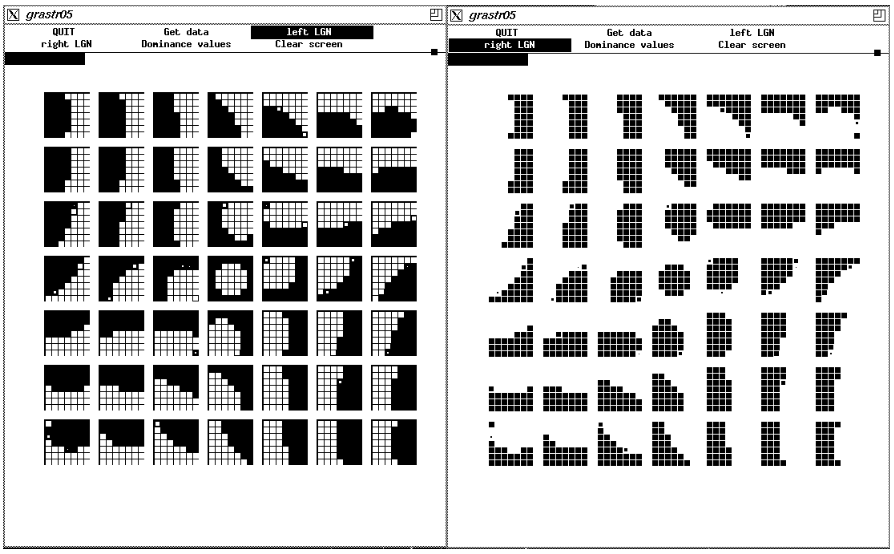

Figure 6–8: The effect of the strength value of 0.5 at 200 iterations.

- The first set of simulations use a fixed width of correlation within and between eyes of 2.8. A fixed strength value of 0.3 was used and the interaction took the values 1.0, 2.0 and 3.0.

  The results of these simulations are plotted in figure 6–9[1]. It is possible to observe in each case dominance was not quite reached but the maximum overall dominance value was near to 1. The results in these cases are very similar to one another except that the speed at which dominance was reached was slower for the case of an interaction width of 1.0. Once reached, the maximum values remain stable.

- In this second set of simulations the same fixed width of correlation within eyes of 2.8 was used, but for this set a narrower correlation width between eyes of 1.4. was used.

  The results were as follows: dominance was reached for the case of an interaction width of 1.0 at a speed slower than that of which the others reach their maximum value. For interaction widths of interaction width 2.0 and 3.0 ocular dominance was not quite reached, and the maximum values of nearly 1 remained stable during the following time steps. In (figure 6–10) it is observed that at 200 iterations some binocular solutions remain the same distribution is obtained using interaction width of 2.8.

6.2.3 Varying Interaction Width Values With Zero Correlation Between Eyes

In order to explore completely the effects of varying the interaction width, the set of simulations was carried out under the following conditions:

[1]the graph plotted for interaction width of 1.4 is equal to this one.

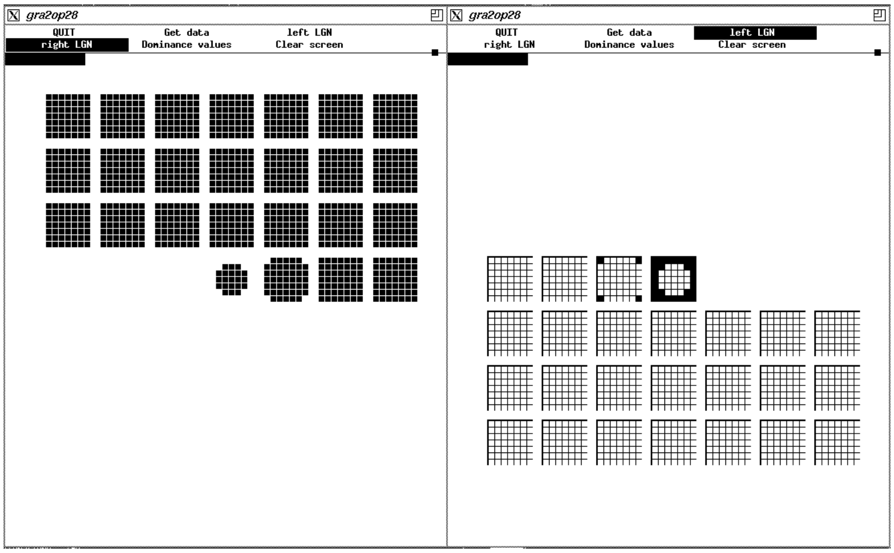

Figure 6–9: Positive correlation width between eyes using a 2.8 within them, and interaction width of 2.0 at 200 iterations.

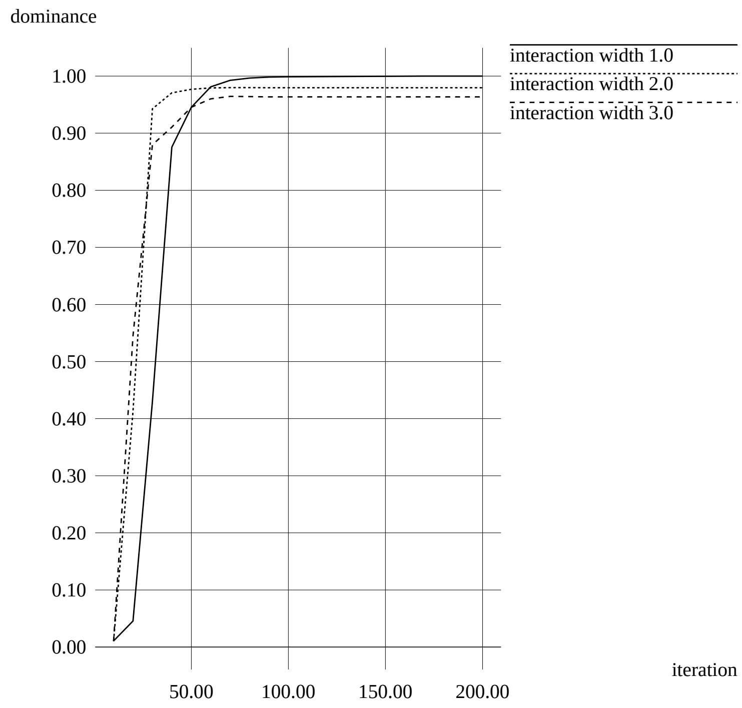

Figure 6–10: Result at 200 iterations with correlation width of 1.4 between eyes and interaction width of 3.0

- Maintaining the correlation width within eyes fixed at 2.8, and the correlation between eyes fixed at zero, the interaction width took the values 1.0, 2.0 and 3.0.

  The results obtained from these simulations are plotted in figure 6–11. Dominance was reached for an interaction width of 1.0 at 130 iterations. But for the other two values, 2.0 and 3.0, dominance was not quite reached. Their maximum values were reached before 100 iterations, and they are near to 1 and stable in the following iterations. These results were similar to those obtained with small positive correlations between eyes.

- The next set used the same conditions as in the previous set of simulations, except for a narrower correlation width of 1.8 within eyes. The results are plotted in figure 6–12 and they are as follows:

  In any of the cases simulated, dominance was not reached. These results were inconclusive because the curves at 200 iterations had an increasing tendency to 1; simulations with more time steps should be done in order to see if complete dominance is reached if more than 200 iterations are allowed.

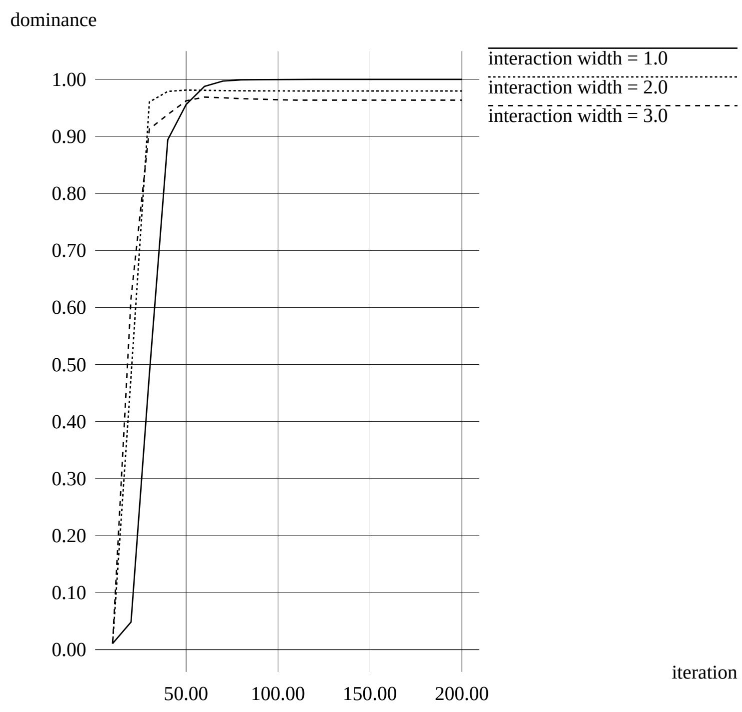

Figure 6–11: Varying the interaction width with fixed correlation width within eyes of 2.8 and zero correlation width between eyes.

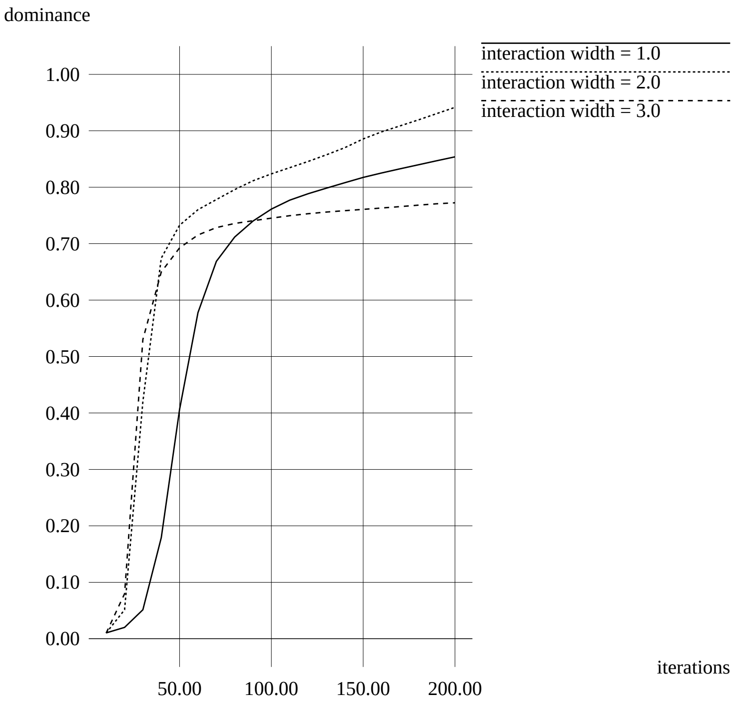

Figure 6–12: Varying the interaction width with a narrower correlation width within eyes of 1.8 and zero correlation width between eyes.

Chapter 7

Analysis of the Results

In this chapter the effect of varying the parameters are considered with particular reference to if dominance occurs, speed to reach dominance, and topographic refinement.

7.1 Comparing One and Multiple Cortical Cells Results

The simulation results for one and multiple cortical cells are in general terms similar between each other.

The exploration of the strength of between-eye correlation is one of the more important investigated in this project. For this reason it is important to compare the results obtained varying the strength value for each of simulations in one and multiple cortical cells. As it is shown in figures 6–8 and 6–7, the common features are that for the cases of strength value of 0.4 and 0.5 the dominance was not reached. For strength value of 0.3 in one cortical cell the dominance was reached but for multiple cortical cell it was not. In the second case because the curve tended to 1 it could be possible that using more number of iterations the dominance could be reached, this suggest that more exploration in this point should be done. In general terms it is possible to say that for both sets (one and multiple cortical cells) the model behaviour is similar. And as mentioned before these results support the theoretical predictions form [Dayan & Goodhill 92, Goodhill, 91] that adding positive between eye correlations prevent dominance.

7.2 Development of a Single Cortical Cell

7.2.1 Effects of Varying the Initial Range of Weight Values

In order to understand how varying the initial range of weight values affected the simulation results three set of simulations were done . The analysis of these results is described below.

- In the case in which the ranges of values chosen were smaller and bigger than those used by Miller (see 5–2 in Chapter 5), no radical modification in the results was observed. In all the cases ocular dominance was reached. The speed in which it was reached was less than 62 iterations.

- For the case in which the lower limit was fixed to 0 and the upper limit in the range was increased by one each time until the maximum value allowed the results obtained shows the following: as with the first set no radical modifications in the results were observed, a) for all the cases the dominance was reached, b) the speed to reach the dominance decreased as the range was moved to higher values, as in the previous simulations (see fig. 5–3).

- For the case of a fixed range window the results are radically different compared with the previous two. There were two main features in these simulations; one is the speed to reach the dominance decreased as the range was moved to higher values, and second while the dominance was reached only for values ho higher than the range 1.8-2.2, the ranges above of that lead to binocular solutions (see fig. 5–4).

One difference between the first and second simulations is that in the second the starting point in the overall dominance curves is the same for all the ranges

decay value	*iterations*
0.01	63
0.05	78
0.1	178

Table 7–1: Speed in which the complete dominance is reached varying decay term GAMMA.

used, that could be explained by the fact that the lower limit was fixed to zero with the same value for all the cases.

The justification of using a specific range of 0.8-1.2 was not given in [Miller *et al* 89]. The reason to use a range of small values arise from the simulation results and it is because using a range with small values allows the complete development of the model simulations. With this it was possible to observe features as the topography refinement development, because it happened before the ocular dominance is reached and the speed of the latter is slower with the range used. In this way it is possible to observe this development at each timestep, on the other hand if the range of values chosen lay near to the upper limit this did not allow the development.

7.2.2 Influence of the Decay Terms

The increment of the two decay terms involved affect the results differently. Varying the value of the decay term ϵ did not modified the results because it is associated with the arbor function which in our simulations was a flat function equal to 1 i.e. all synapses are affected equally. On the other hand the decay term γ produced a delay to reach dominance as it is increased, because it is associated with the total synaptic strength, i.e. the stronger synapses get punished more than the weakers ones.

The speed at which dominance was reached was slower as the decay term γ was increased as it is shown in table 7–1.

within-eye	*iterations*
correlations	
2.4	91
2.8	62
3.0	55
3.4	44

Table 7–2: Speed in which the complete dominance is reached at different within-eye correlation values.

7.2.3 Effects of Varying the Correlation

Within Eyes

1. The effect of varying the correlation width within eyes observed in our results was; above a width of 2.4 monocular solutions are obtained. On the other hand with values below of 2.4 to binocular solutions are obtained.

2. The speed at which monocularity was reached increases in a way roughly proportional to the correlation width as shown in the figure 7–2. Then more correlated elements produce monocular solutions faster.

3. Given the property that the correlations decrease with distance, the central input elements have more correlated neighbours than the elements in the periphery, this leads to a faster growth in the elements in the centre of the receptive field producing in this way topographic refinement [Miller *et al* 89]. Comparing one case in which the dominance is reached as using within-eye correlation width value set to 2.4 with another in which it was not reached as with value of 2.0, the refinement occurs faster in the values from the eye which is becoming dominant but in the values from the other eye the refinement occurs faster for the within-eye correlation of 2.0 but remains longer, as it is observed in the figure 7–1. Then intuitively it would be expected that given a bigger correlation width to the synapses in the periphery it would reduces topographic refinement during development.

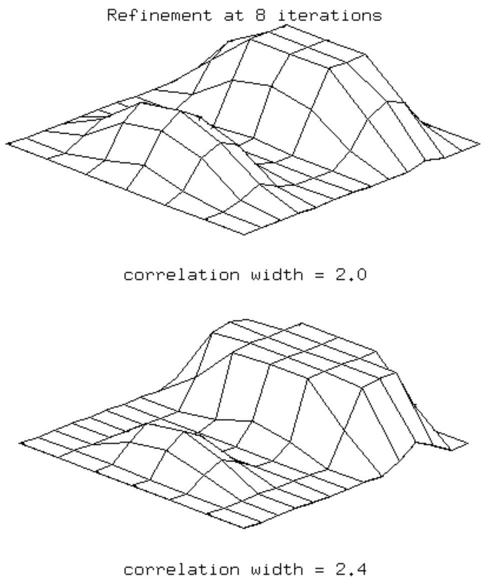

Figure 7–1: Comparing refinement in correlation width 2.0 and 2.4

Between Eyes

1. For this set of simulations the conditions were: The correlation width within eyes was set to 2.8. The changes in the between-eye width and strength of correlation function. For these simulations were as follows; one, two and three times the correlation width within eyes (2.8), that means (2.8, 2∗2.8 and 3∗2.8). Then the values in the strength used for each of these between-eye correlation widths were 0.3, 0.4 and 0.5. *Results* - As predicted by [Dayan & Goodhill 92, Goodhill, 91], adding some positive between-eye correlations can lead to binocular solutions. In the cases of 1 time (2.8) and 2 times (2*2.8) the strength of the between-eye correlation has to be higher

than 0.3 to prevent the ocular dominance. But if the between-eye width is increased to $3*2.8$ all the solutions become binoculars.

2. In the case in which the cortical unit saturates at complete dominance was slower than the speed in which dominance was reached using zero correlation between eyes.

7.2.4 Effect of Different Input Grid Sizes

In the results obtained varying the grid sizes, it was observed that only for small sizes (5x5 and 7x7) ocular dominance was reached and for size 9x9 the maximum value was 0.848 tending to 1, to know if it could reach ocular dominance it should be tested with more than 200 iterations. For sizes greater than 9x9 the overall dominance values tended to zero. The explanation for this is that as described before, the correlations decrease with distance, and for big distances it was observed that this not only leads to faster growth in the elements in the centre but the overall dominance values did not lead to a ocular dominance at all.

7.2.5 Relation Between Grid Sizes and Correlation Width

The explanation given for the results using different input grid sizes with a fixed correlation width, is proved by using proportional correlations widths as the size of the grid was increased. The results were as expected the ocular dominance was reached in all the sizes using this proportional correlation width. The correlation width was adjusted directly proportional to one of the values in which ocular dominance was reached using the 7x7 size (2.4). In this second set the speed in which dominance was reached decreased as the grid size increased.

7.3 Development of Multiple Cortical Cells

7.3.1 Effect of positive correlations between eyes

To see how strong the correlation between eyes needed to be to prevent dominance, different strengths of correlations between eyes were tried using fixed correlation width within eyes set to 2.8 and between eyes set to $2*2.8$. With the strength value set to 0.3 ocular dominance was not reached at 200 iterations but the maximum value reached was close to 1, it could be possible that dominance would be reached but it should be tested with bigger number of iterations. For the values 0.4 and 0.5 the results obtained were binocular, the central cortical cell of these results is shown in figure 7–2.

7.3.2 Effect of Varying the Interaction Width

For this set of simulation the strength of correlations between eyes values was fixed at 0.3. The variation on the interaction width was tested with two different within-eye correlation widths (2.8 and 1.4):

1. Complete dominance was not reached using a fixed positive correlation width between eyes and varying the interaction width to 1.0, 2.0 and 3.0, the maximum value reached in all the cases was bigger than 0.9 near to 1 but not reaching complete dominance, for 1.0 all the elements were dominated by one eye except two, using interaction width set to 2.0 three units became binocular and the rest monocular, and with interaction width set to 3.0 five units became binocular. Then the number of binocular solutions was increased as interaction width was increased, decreasing then the monocular ones.

2. Using a narrower between-eye correlation width fixed to 1.4 and varying in the same range the interaction width (1.0, 2.0 and 3.0). The results shown

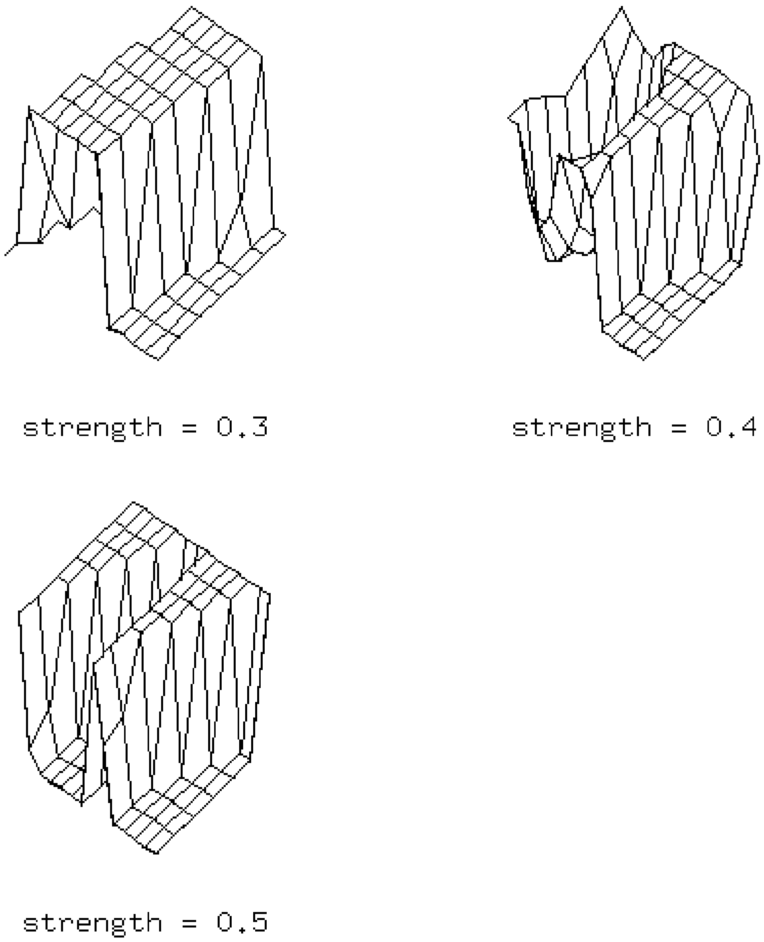

Figure 7–2: Comparing the central cortical cell under effects of varying the strength of correlation function between eyes.

that for the case of 1.0 interaction width value, the ocular dominance is reached at 170 iterations, but with interaction width equal to 2.0 and 3.0 dominance was not reached these simulations lead to binocular solutions as it was shown in figure 6–8 in Chapter 6.

7.3.3 Varying Interaction Width with Zero Correlation Between Eyes

The effects observed varying the interaction width using zero between-eye correlations were analysed for the following two cases of different values in the correlation width within eyes:

Using a Correlation Width of 2.8

In this case only for the case of interaction width set to 1.0 the overall ocular dominance was reached and the monocular solutions decreased as the interaction width value was increased. These results are similar to those obtained using positive between-eye correlation of width 1.4 and within-eye of width 2.8, comparing figure 6–10 and figure 6–11. Given this evidence it is possible to say that using positive between-eye correlation of width 1.4 and strength value of 0.3 did not affect the model behaviour, leading to monocular solutions.

Using a Narrower Correlation Width (1.4)

With a narrower correlation width value within eyes dominance was not reached The results show a tendency to increase to 1. In order to do a complete analysis of these cases it would be necessary to do longer simulations.

From the results it is possible to say that the effects of varying interaction width, did not influence in a significant manner the dominance values. And as it is discussed in [Miller *et al* 89], this parameter can alter the widths of the patches, but this was not explored in this project. The size of the matrices used did not allow to evaluate the changes in the stripe width.

Chapter 8

Conclusions

This chapter summarises the main points of the project. Following this, some conclusions regarding the analysis of the experimental results are presented. Finally, future work as suggested by the results obtained is discussed.

8.1 Summary

- The motivation for this project is provided by the theoretical analysis of [Dayan & Goodhill 92, Goodhill, 91] about Miller's model of ocular dominance development. In this analysis it was suggested that under more realistic conditions the model could give binocular rather than monocular solutions. This suggestion was to be tested empirically by creating and testing a computer implementation of Miller's model.

- The biological bases of Miller's model were described, namely the main brain structures involved and ocular dominance formation in the visual pathway of cat and monkeys.

- Some previous models of ocular dominance segregation and one of map formation were briefly described to give a general background of the work done in this area. Miller's model, and the closely related of Linsker were described in more detail.

- Miller's model was implemented to explore the parameter space of this model. The data structures chosen for the implementation were discussed and the conditions required to run the program were given.

- As discussed in [Dayan & Goodhill 92, Goodhill, 91] the assumption of no correlations between eyes is biologically implausible. For this reason simulations explored the case of using positive between-eye correlations. Some general and central parameters were explored for this purpose.

- An analysis and comparison between results obtained from simulations performed for one and multiple cortical cells was done. The results were compared with those obtained by Miller and the theoretical prediction of [Dayan & Goodhill 92, Goodhill, 91].

8.2 Model, Predictions and Results

Given the motivation of this project mentioned above, an empirical investigation of the model's parameter space was done to analyse mainly the effect of adding positive correlations between eyes and some other parameters. From the results described in chapter 5 and 6 and the analyses in chapter 7, the following conclusions were reached.

- In the case of adding positive correlation between eyes which is a case not considered by Miller, the results obtained lead to binocular solutions which support the theoretical predictions of [Dayan & Goodhill 92, Goodhill, 91].

- Taking into account parameters which are part of the parameter space exploration but not the central part of this project, the following results were obtained: In the initial range of weight values the justification to use small range of values was to allow the complete model development during the simulations. It was observed that ocular dominance was not reached using ranges near to the upper limit of the given range of values.

- The decay term γ affected the speed at which dominance was reached. As γ increased the speed to reach dominance decreased.

- The exploration of the effects of increasing the number of input cells, showed that, given a fixed within-eye correlation width, as the grid size is increased, the monocular solutions decreased. This supports the suggestion of [Dayan & Goodhill 92, Goodhill, 91] that the binocular solutions start to catch up with the monocular solutions as the number of cells increases.

- Given the size of the matrices used in this project it was not possible to evaluate changes in the stripe width. It could be interesting to see the effect on this feature by varying the interaction function.

Miller's model of ocular dominance works under very specific values, following his assumptions. There are several predictions which involve various parameters in the model. Unfortunately given the time and the hardware, it wasn't possible to investigate the complete parameter space. Then in conclusion the most important parameter empirically investigated was the positive between-eye correlations. It was demonstrated that by adding positive between-eye correlations Miller's model of ocular dominance segregation is prevented.

8.3 Further work

We have been testing some of the theoretical predictions of [Dayan & Goodhill 92, Goodhill, 91], but there are some more which will be interesting to test empirically using this implementation. For example negative within-eyes correlations and making the within-eye correlation parameters slightly different for each eye.

Miller's work is modeling several biological mechanisms that can contribute to the development of ocular dominance. It is important to emphasise that this model works under several assumptions. One of these assumptions, namely that there is no correlation between eyes, seems to be biologically implausible. For this reason and for the biological interpretation of this model it will be important to analyse in detail the biological plausibility of Miller's model's mechanisms, parameters and assumptions.

Some suggestions for further work to improve this implementation:

- It could be very useful to link the model's program with the graphic program. In this way an user interface to modify parameters could be included in the control panel and the results of each modification could be seen with the same program.

- Given the important problem of processing speed, an implementation of this model on a parallel processing computer could be an alternative solution. It may even be possible to observe results dynamically as they are generated. Larger grid sizes could also be used in the simulations.

Bibliography

[Bienenstock *et al* 82] E.L. Bienenstock, L.N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, (2):437–455, 1982.

[Dayan & Goodhill 92] P.S. Dayan and G.J. Goodhill. Perturbing hebbian rules. *Advances in Neural Information Processing Systems*, 4:19–26, 1992.

[Goodhill 91] G. Goodhill. *Correlations, competition, and optimality: modelling the development of topography and ocular dominance*. Unpublished PhD thesis, School of Cognitive and Computing Sciences, The University of Sussex, 1991.

[Hubel & Wiesel 77] D.H. Hubel and T.N. Wiesel. Functional architecture of macaque monkey visual cortex. *Proc. R. Soc. Lond.*, 198:1–59, 1977.

[Kuffler *et al* 84] S.W. Kuffler, J.G. Nicholls, and A.R. Martin. *From Neuron to Brain*. Sinauer Associates Inc., 1984.

[Linsker 88] R. Linsker. Self-organization in a perceptual network. *Computer*, pages 105–116, March 1988.

[Miller 89] K.D. Miller. *Correlation-based mechanisms in visual cortex: theoretical and experimental studies.* Unpublished PhD thesis, Medical School, Stanford University, 1989.

[Miller 90] K.D. Miller. Correlation-based models of neural development. In *Neuroscience and Connectionist Theory (Chapter 7)*, pages 267–353. Hillsborough, NJ: Lawrence Erlbaum., 1990.

[Miller *et al* 89] K.D. Miller, J.B. Keller, and M.P Stryker. Ocular dominance column development: analysis and simulation. *Science*, 245:605–615, 1989.

[Stryker & Harris 86] M.P. Stryker and W. Harris. Binocular impulse blockade prevents the fomation of ocular dominance columns in cat visual cortex. *Jou. Neurosci.*, 6:2117–2133, 1986.

[Swindale 80] N.V. Swindale. A model for the formation of ocular dominance stripes. *Proc. R. Soc Lond. B.*, pages 243–264, 1980.

[Willshaw & von derMalsburg 76] D.J. Willshaw and C. von der Malsburg. How patterned neural connection can be set up by self organisation. *Proc. R. Soc Lond. B.*, (194):431–445, 1976.

Appendix A

Miller's Model Program

```
/* --- miller1.h --- This is the header file for the miller1.c program,
       The general functions, variables and arrays are declared in this
       file to be accesed and used by the program.
*/

#include <stdio.h>
#include <math.h>
#include <stddef.h>

#define CELLS_LGN 7                 /* Number of CELLS per side        */
#define CELLS_CX  1                 /* used in cortex and LGN layers   */



#define SQR_LGN 49                  /* total number of cells in matrix */
#define SQR_CX   1



#define DIM 2            /* Numbers associated with Left and Right eyes */
#define l   0                                      /*      left eye    */
#define r   1                                      /*     right eye    */
#define LGN 0                                      /*      LGN type    */
#define CX  1                                      /*   cortex type    */

double pos_LGN[SQR_LGN][DIM];       /* x,y coordinates associated with */
```

```
                                    /* each position in the LGN       */
double pos_CX[SQR_CX][DIM];         /* x,y coordinates associated with */
                                    /* each position in the cortex    */

double CorLL[CELLS_LGN];            /* In these arrays the Gaussian   */
double CorRR[CELLS_LGN];            /* values are stored for both,    */
double CorLR[CELLS_LGN];            /* Correlation and Interaction    */
double CorRL[CELLS_LGN];            /* functions                      */

double Inter[CELLS_CX];


double arbor[SQR_LGN];              /* matrix generated by arbor fun */



double weight[DIM][SQR_LGN][SQR_CX];  /* In this matrix the first block */
                                      /* corresponds to the eyes Left or*/
                                      /* Right, the second to the LGN   */
                                      /* cells and the third to the     */
                                      /* cortex cells                   */

double changes[DIM][SQR_LGN][SQR_CX];    /* Matrix to storage the changes  */
                                      /* calculated sofar               */

#define SMAX    8.0                 /* limit of synaptic strenghts 0-8.*/
#define SMIN    0.0



#define INITM     0.8                 /* Range of intial weight values */
#define INITP     1.2

int iterations = 200;                            /* Number of iterations */
```

```
#define INIT_RAN  0   /* What point in random number sequence to start */

#define GAMMA 0                                                /* Decay terms */
#define EPSILON 0

#define LAMBDA 0.008             /* Step size for development equation */

#define FACTOR_CLL 2.8            /* Width values for Gaussian function */
#define FACTOR_CRR 2.8
#define FACTOR_CLR 0.4
#define FACTOR_CRL 0.4

#define FACTOR_I 1.0

char *filename = "corop04";
FILE *fd, *fopen();                              /* data file pointers */
```

Appendix B

Miller's Model Program

```
/* -----------------------------------------------------------------

        File      : miller.c  / Miller's Model
        Author    : Rocio Aguilar-Chongtay
        Date      : August 1992
        Purpose   : This program is an implementation of Miller's model
                     [Miller etal 1989].

------------------------------------------------------------------- */

#include "miller.h"                          /* variables declaration */
#include "random.c"                      /* a decent random generator */

int c=0;
/* --------------------------------------------------------------- */

main(argc, argv)                                 /* control module */
int argc;
char *argv[];
{     int i, chatty = 1;
     extern double random();
                                         /* running random generator */
     for (i = 0; i < INIT_RAN; ++i)
        random();
```

```
    if (chatty == 1)
      printf("\n Finding X,Y positions ");
         find_xy_position();

    if (chatty == 1)
      printf("\n Generating Gaussian tables");
        gauss();

    if (chatty == 1)
      printf("\n Calculating Arbor function");
        arborF();

    if (chatty == 1)
      printf("\n Initializing weights\n");
         build_weight_matrix();

    if (chatty == 1)
      printf("\n Changes in the total synaptic strength\n");
      fd = fopen(filename, "wb");   /* Erasing old datafile */
      fclose(fd);
      for (i = 0; i< iterations; ++i)
        {
         printf("Iterations = %d\n", i);
         weight_changes();
         printf("\t\n Applying constraints\n");
         constraints();
         printf("\t\n Checking ranges and Updating weights\n");
         updating_weights();
         if (i % 10 == 0)
          {
           printf("\n Storing results in a datafile\n");
           storing_weights();
    }
}
```

```
}

/* ----------------------------------------------------------------- */



/* initialising weight_matrix with the random values generated
 weight_matrix initialization */

double generate_random(x, y)          /* generating random values */
double x,y;
{
 double val;
 extern double random();    /* using the O'Keffe random generator */
 val = (y - x) * random() + x;
 return(val);
}

build_weight_matrix()
{int i,j;
 double generate_random();        /* assigning the random values  */
     for (i=0; i< SQR_LGN; ++i)      /* to the weight_matrix         */
        for (j=0; j< SQR_CX; ++j)
        {
          weight[l][i][j] = generate_random(INITM, INITP);
          weight[r][i][j] = generate_random(INITM, INITP);
}
}
/* ----------------------------------------------------------------- */
/* POSITION - finding the coordinates in which some CELLS is located
              in some bidimensional matrix
*/
find_xy_position()
{
 int i, x=0, y=1;
```

```
   for (i=0; i< SQR_LGN; i++)  /* looping to get all LGN cells position */
    {
     pos_LGN[i][x] = i % CELLS_LGN; /* storing the cell number asociated*/
     pos_LGN[i][y] = i / CELLS_LGN; /* with x,y coordinates obtainded   */
    }

   for (i=0; i< SQR_CX; i++)/* looping to get all cortex cells position */
    {
     pos_CX[i][x] = i % CELLS_CX;   /* storing the cell number asociated*/
     pos_CX[i][y] = i / CELLS_CX;   /* with x,y coordinates obtainded   */
    }

}
/* ----------------------------------------------------------------- */
/* Generating Gaussian function tables */

gauss()
{ int x;
  double x2;

  for (x = 0; x < CELLS_LGN; ++x)
  { x2 = ((double) (x*x));
    CorLL[x] = exp(-x2/(FACTOR_CLL*FACTOR_CLL));
    CorRR[x] = exp(-x2/(FACTOR_CRR*FACTOR_CRR));
    CorLR[x] = exp(-x2/(FACTOR_CLR*FACTOR_CLR));
    CorRL[x] = exp(-x2/(FACTOR_CRL*FACTOR_CRL));

    Inter[x] = exp(-x2/(FACTOR_I*FACTOR_I));

  }

/*
    printf("x = %4d C_LL = %6.3f\n", x, CorLL[x]);
    printf("x = %4d C_RR = %6.3f\n", x, CorRR[x]);
```

```
    printf("x = %4d I    = %6.3f\n", x, Inter[x]);
*/

}

/* ------------------------------------------------------------------ */
/* Finding the distances*/
double CorreInt(a,b,funct)
int a, b, funct;
{
  double result;
  int x=0, y=1, valx, valy, xdiff, ydiff;

   if(funct == 0)
     {
     valx = pos_CX[b][x] - pos_CX[a][x];
     valy = pos_CX[b][y] - pos_CX[a][y];
     }
   else
     {
     valx = pos_LGN[b][x] - pos_LGN[a][x];
     valy = pos_LGN[b][y] - pos_LGN[a][y];
     }

     xdiff = abs(valx);
     ydiff = abs(valy);

     switch(funct)
     {
      case 0:
        result = Inter[xdiff]*Inter[ydiff];
        break;
      case 1:
        result = CorLL[xdiff]*CorLL[ydiff];
```

```
        break;
      case 2:
        result = CorRR[xdiff]*CorRR[ydiff];
        break;
      case 3:
        result = CorLR[xdiff]*CorLR[ydiff];
        break;
      case 4:
        result = CorRL[xdiff]*CorRL[ydiff];
        break;
     }

/*
     printf("CorreInt:\n");
     printf("a = %4d, b = %4d, funct = %4d\n", a, b, funct);
     printf("xdiff = %4d, ydiff = %4d\n", xdiff, ydiff);
     printf("Result = %6.3f\n", result);
*/
     return(result);

}

/* ------------------------------------------------------------------ */

/* Calculating Arbor function in this case a flat arbor function is used
   where 1 is assigned */

arborF()
{
        int i;
          for (i = 0; i < SQR_LGN; ++i)
        {
          arbor[i]=1;
         }
}
```

```
/* ------------------------------------------------------------------- */
/* In this function the weight changes are calculated */

 weight_changes()
{ double CorreInt(),changesl, changesr;

 int a,b,x,y,funcI=0,funcCLL=1, funcCRR=2, funcCLR=3, funcCRL=4;

 changesl = 0;
 changesr = 0;

          for (x = 0; x < SQR_CX; ++x) /* fore each cortical cell */
  {
/*            printf("x = %d\n", x);  */
           for (a = 0; a < SQR_LGN; ++a)   /* for each LGN cell */
              {
              for (y = 0; y < SQR_CX; ++y) /* changes for all y, b */
                 for (b = 0; b < SQR_LGN; ++b)
                    {
                             /* calculating for the left eye */

                         changesl += CorreInt(x,y,funcI)*
                                   (CorreInt(a,b,funcCLL)*weight[l][b][y]+
                                    CorreInt(a,b,funcCLR)*weight[r][b][y]);

                            /* calculating for the right eye */

                         changesr += CorreInt(x,y,funcI)*
                                   (CorreInt(a,b,funcCRR)*weight[r][b][y]+
                                    CorreInt(a,b,funcCRL)*weight[l][b][y]);
              }

               changesl*=LAMBDA*arbor[x]*arbor[a];
               changesr*=LAMBDA*arbor[x]*arbor[a];
```

```
                changes[l][a][x] = changesl - ( GAMMA*weight[l][a][x]-
                                                EPSILON*arbor[x]*arbor[a]);

                changes[r][a][x] = changesr - (GAMMA*weight[r][a][x]-
                                                EPSILON*arbor[x]*arbor[a]);

/*
                printf("Weight Changes: x = %4d, a = %4d\n", x, a);
                printf("changes[l][a][x] = %6.3f, changes[r][a][x] = %6.3f\n",
                         changes[l][a][x], changes[r][a][x]);
*/
      }
       }

}

/* ----------------------------------------------------------------- */
/* In this function the constraints are applied */

constraints()
{
  double suml,sumr,sumtot;
  int  x,a,i;

  for (x = 0; x < SQR_CX; ++x) /* fore each cortical cell */
    {
      suml = 0;
      sumr = 0;
      sumtot = 0;
                                          /* adding changes */
      for (a = 0; a < SQR_LGN; ++a)
        {
         suml +=changes[l][a][x];
         sumr +=changes[r][a][x];
```

```
        }

       sumtot = suml+sumr;
/*       printf("sumtot = %6.3f\n", sumtot); */
                                /* substracting constrain values */
      for (a = 0; a < SQR_LGN; ++a)
        {
         changes[l][a][x] -=  (sumtot/(SQR_LGN*2));
         changes[r][a][x] -=  (sumtot/(SQR_LGN*2));
        }
    }

  for (a = 0; a < SQR_LGN; ++a)
    {
      suml = 0;
      sumr = 0;

      for (x = 0; x < SQR_CX; ++x)
        {
         suml +=changes[l][a][x];
         sumr +=changes[r][a][x];
        }

      for (x = 0; x < SQR_CX; ++x)
        {
         changes[l][a][x] -= (suml/SQR_LGN);
         changes[r][a][x] -= (sumr/SQR_LGN);
        }
    }

    printf("Constraints:\n");

/*    for (x = 0; x < SQR_CX; ++x)
         for (a = 0; a < SQR_LGN; ++a)
         {
```

```
              printf("x = %4d, a = %4d\n", x, a);
              printf("changes[l][a][x] = %6.3f, changes[r][a][x] = %6.3f\n",
                        changes[l][a][x], changes[r][a][x]);
         }
*/

}


/* ------------------------------------------------------------------- */
/* In this function the weights are updated and limited to a range
   between 0 and 8
*/
 updating_weights()
{
int x, a;

    for (x=0; x< SQR_CX; ++x)
      for (a=0; a< SQR_LGN; ++a)
        {                                       /* updating weights */
         weight[l][a][x] += changes[l][a][x];   /* left eye         */

         if(weight[l][a][x] > SMAX)   /* checking with range values */
            weight[l][a][x] = SMAX;   /* between the upper and lower*/
         else                         /* bounds                     */
           if (weight[l][a][x] < SMIN)
            weight[l][a][x] = SMIN;
                                                /* updating weights */
         weight[r][a][x] += changes[r][a][x];   /* right eye        */

         if(weight[r][a][x] > SMAX)   /* checking with range values */
            weight[r][a][x] = SMAX;   /* between the upper and lower*/
         else                         /* bounds                     */
           if (weight[r][a][x] < SMIN)
            weight[r][a][x] = SMIN;
        }
```

```
}
/* ------------------------------------------------------------------ */
/* Storing the weight values every 10 iterations in a datafile to be
   used in the graphic program
*/

void storing_weights()
{
 int n=1, a, b, size, x;

 x =abs(SQR_CX/2);

 fd = fopen(filename, "a+b");
 size = (sizeof(weight[DIM][SQR_LGN][SQR_CX]));

      for(a = 0; a < SQR_LGN; ++a)   /*writing the data in a file  */
        {
         for (b = 0; b < SQR_CX; ++b)
            {
             fwrite(&weight[l][a][b], size, n, fd);
             fwrite(&weight[r][a][b], size, n, fd);
             }
        }



 for(a = 0; a < SQR_LGN; ++a)
 {
      if (a % CELLS_LGN == 0)
           printf("\n");

      printf("%7.3f", weight[l][a][x]);
  }
  printf("\n");
```

```
 for(a = 0; a < SQR_LGN; ++a)
 {
      if (a % CELLS_LGN == 0)
           printf("\n");

      printf("%7.3f", weight[r][a][x]);
  }

  printf("\n");


 fclose(fd);

}

/* ----------------------------------------------------------------- */
```

Appendix C

Overall Dominance Calculation

```
#include <stdio.h>
#include "miller.h"             /* Including header with parameters */

int c=1, i;


main()
{
  fd = fopen(filename, "rb");                  /* opend data file */

  for (i = 0; i< iterations/10; ++i)
   {
    read_data();
    dominance();
   }
}

read_data()                          /* reading from the datafile */
{
   int its, cortex, lgn, size, n;
   size = abs(sizeof(weight[DIM][SQR_LGN][SQR_CX]));

   if ((its*10)-10 == iterations)        /* check iteration limit */
     {
```

```
      its = 0;
      fd = fopen(filename, "rb");
     }

         /* reading the data generated by Miller's model program */



          for (lgn = 0; lgn < SQR_LGN; ++lgn)
            for (cortex = 0; cortex < SQR_CX; ++cortex)
       {
         fread(&weight[l][lgn][cortex], size, 1, fd);
         fread(&weight[r][lgn][cortex], size, 1, fd);
       }
      ++ its;
 }


dominance()
{
 int a,b ;
 double  h, sum, sum_b, avedom, domcx[SQR_CX],cx_cells;;

 cx_cells = ((double) SQR_CX);
 avedom = 0;
 sum_b  = 0;

 for(b = 0; b < SQR_CX; ++b)           /* for each cortical cell */
  {
  domcx[b]=0;
  sum = 0;
  for(a = 0; a < SQR_LGN; ++a)         /* for each LGN cell */
    {
     domcx[b] += (weight[r][a][b]-weight[l][a][b]); /* calculating the*/
     sum      += (weight[r][a][b]+weight[l][a][b]); /* dominance for  */
```

```
    }                                                  /* individually */
     domcx[b] /= sum;
  }

 for(b = 0; b < SQR_CX; ++b)
      sum_b += fabs(domcx[b]);

    avedom = (1/cx_cells) * sum_b;                     /* normalizing */

      printf("%d %7.6f\n", c*10, avedom);

 ++c;

}
```

Appendix D

Miller's Model Program

```
/* graph10.c

   Uses xlist widget to create items for selection.

*/

#include <stdio.h>
#include <math.h>
#include <values.h>


#define CGRAPHICS              /* Xwindows and widgets include files */

#include <X11/X.h>
#include <X11/Xlib.h>
#include <X11/Xatom.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xaw/Command.h>
#include <X11/Xaw/Paned.h>
#include <X11/Xaw/Box.h>
#include <X11/Xaw/Cardinals.h>

#include <X11/Xaw/Form.h>
```

```
#include <X11/Xaw/Viewport.h>
#include <X11/Xaw/AsciiText.h>
#include <X11/Xaw/Dialog.h>
#include <X11/Xaw/Label.h>
#include <X11/Xaw/List.h>
#include <X11/Xaw/MenuButton.h>
#include <X11/Xaw/Scrollbar.h>
#include <X11/Xaw/SimpleMenu.h>
#include <X11/Xaw/SmeBSB.h>
#include <X11/Xaw/SmeLine.h>
#include <X11/Xaw/StripChart.h>
#include <X11/Xaw/Toggle.h>


#include "miller.h"

String fallback_resources[] =
{
    "*Paned.height:           650",          /* Frame sizes */
    "*Paned.width:            550",
    "*Box.width:              550",
    "*Box.height:             550",
    "*go.label:               Doing....",
    NULL,
};

XPoint base = { 40 , 550 };
XPoint step = { 60 , -40 };

Widget box;
Display *disp;
Window win;
GC gcb,gcw;
XGCValues *gcv;
```

```
static void Syntax(), make_list_widget(), Selection();

#define frand() ((double) random() / MAXLONG)

int check = 0;

static int its=0;

int PIC_SIZE     = 550;              /* window pixel size */
int Xi           = 50;               /* initial position  */
int Yi           = 50;               /* window scale factor */
double WSCALE    = CELLS_CX;

#define CORX CELLS_CX         /* Values for graphic program */
#define CORY CELLS_CX
#define LGNX CELLS_LGN
#define LGNY CELLS_LGN

#define MAXR    CELLS_LGN                /* maximum row number */
#define MAXC    CELLS_LGN             /* maximum column number */

void
main(argc, argv)
int argc;
char **argv;
{

     Widget toplevel, outer, getdatcom, disforcom;

     XtAppContext app_con;

     fd = fopen(filename, "rb");

     toplevel = XtAppInitialize(&app_con, "Xwidgets", NULL, ZERO,
            &argc, argv, fallback_resources, NULL, ZERO);
```

```
     outer = XtCreateManagedWidget( "paned", panedWidgetClass, toplevel,
      NULL, ZERO);

     make_list_widget(outer);

     box = XtCreateManagedWidget( "box", boxWidgetClass, outer,
    NULL, ZERO);

     XtRealizeWidget(toplevel);

     win = XtWindow(box);
     disp = XtDisplay(box);

     gcb = XCreateGC(disp,win,0,NULL);        /* colour values for */
     gcw = XCreateGC(disp,win,0,NULL);        /* black and white   */

     XSetForeground(disp,gcb,1);
     XSetForeground(disp,gcw,0);

     cls();

     XtAppMainLoop(app_con);

}

static void
Syntax(app_con, call)
XtAppContext app_con;
char *call;
{
    XtDestroyApplicationContext(app_con);
    fprintf(stderr, "Usage: %s\n", call);
    exit(1);
}
```

```
static void
make_list_widget(w)
Widget w;
{
     Widget list;                        /* Widgen MENU */
     Arg args[1];                        /* options available in panel */
     static String items[] = {
        "        QUIT         ",
        "      Get data       ",
        "      left LGN       ",
        "      right LGN      ",
        "  Dominance values   ",
        "    Clear screen     ",
        NULL
};

     XtSetArg(args[0], XtNlist, items);
     list= XtCreateManagedWidget( "list", listWidgetClass, w, args, ONE);
     XtAddCallback(list, XtNcallback, Selection, (XtPointer)NULL);
}


static void
Selection(w, closure, call_data)
Widget w;
XtPointer closure, call_data;
{
     int selected,data;
     Widget popup;

     XawListReturnStruct *item = (XawListReturnStruct*)call_data;

     selected = item->list_index;
     XDrawString(disp, win, gcw, 10, 10, "Doing...", 8);
```

```
    XSync(disp,FALSE);
                          /* selecting and executing an option from the MENU */
    switch(selected)
    {
      case 0:
        fclose(fd);               /* closing datafile before quit */
        exit(0);                                          /* quiting */
      case 1:
        get_data();                   /* readng from the datafile */
        break;
      case 2:
        cls();
        draw_weights(0);   /* drawing weight values from left LGN */
        break;
      case 3:
        cls();
        draw_weights(1);  /* drawing weight values from right LGN */
        break;
      case 4:
        cls();
        draw_weights(2);  /* drawing weight values from dominant eye */
        break;
      case 5:             /* clearing the drawing block */
        cls();
        break;
      default:
        printf("LIST ITEM NOT RECOGNIZED\n");
        break;
    }

    XFillRectangle(disp,win,gcb,0,0,100,15);
    XSync(disp,FALSE);
}

get_data()                /* reading from the datafile */
```

```
{
  int cortex, lgn, size;
  size = abs(sizeof(weight[DIM][SQR_LGN][SQR_CX]));

         if ((its*10)-10 == iterations)
           {
            its = 0;
            fd = fopen(filename, "rb");
            }
         printf("iterations = %d\n", its*10);

          /* reading the data generated by Miller's model program */

         for (cortex = 0; cortex < SQR_CX; ++cortex)
           for (lgn = 0; lgn < SQR_LGN; ++lgn)
        {
          fread(&weight[l][lgn][cortex], size, 1, fd);
          fread(&weight[r][lgn][cortex], size, 1, fd);
        }
        ++its;
 }

draw_weights(w)  /* placing in the cortical-LGN block in the window */
int w;
{    int rx, ry, size, boxsize, xstart, ystart;

     size = PIC_SIZE / (LGNX+1);

       boxsize = size / (CORX+1);

       if (boxsize*CORX == size)
         --boxsize;

     XSetLineAttributes(disp,gcw,1,LineSolid,CapProjecting,JoinMiter);
 for (ry = 0; ry < LGNY; ++ry)
```

```
  for (rx = 0; rx < LGNX; ++rx)
          {
               xstart = Xi + size*rx;
               ystart = Yi + size*ry;

               switch(w)
                  {
                    case 0:
                      draw_unit_weight_eye(l, rx, ry, xstart, ystart, boxsize);
                      break;
                    case 1:
                      draw_unit_weight_eye(r, rx, ry, xstart, ystart, boxsize);
                      break;
                    case 2:
                      draw_unit_weight_dominant(rx, ry, xstart, ystart, boxsize);
                      break;
                    case 3:
                      draw_unit_weight_threshold(rx, ry, xstart, ystart, boxsize);
                      break;
  }

   }
     XSync(disp,FALSE);
}

                              /* drawing weight values from left or right eye */

draw_unit_weight_eye(eye, rx, ry, xstart, ystart, boxsize)
int eye, rx, ry, xstart, ystart, boxsize;
{
 int cx, cy, wsize, back, front, size;
 double maxweight = 0, temp_l, temp_r;

 size = PIC_SIZE / LGNX;
```

```
if (eye == 0)          /* left eye */
{
    back=0;
    front=1;
 for (cx = 0; cx <CORX; ++cx)
  for (cy = 0; cy <CORY; ++cy)
      {
        temp_l = weight[eye][ry*LGNX+rx][cy*CORX+cx];

        if (temp_l > maxweight) /* looking for the maximum weight value */
           maxweight = temp_l;
      }
}
else                  /* right eye */
{
     back=1;
     front=0;
     for (cx = 0; cx <CORX; ++cx)
      for (cy = 0; cy <CORY; ++cy)
      {
        temp_r = weight[eye][ry*LGNX+rx][cy*CORX+cx];
        if (temp_r > maxweight) /* looking for the maximum weight value */
           maxweight = temp_r;
      }
}
                       /* putting the background in the opposite colour */
      if ( maxweight > 0)
        {
         shade_region(xstart-1, ystart-1, boxsize*LGNX+1, boxsize*LGNY+1, back);
         WSCALE = ((double) (boxsize - 1));

            /* calculating and drawing the  weight squares */

        for (cy = 0; cy < CORY; ++cy)
          for (cx = 0; cx < CORX; ++cx)
```

```
             {
              wsize = ((int) (WSCALE * weight[eye][ry*LGNX+rx][cy*CORX+cx] /
                                                  maxweight));

              shade_region(boxsize/2+xstart+boxsize*cx-wsize/2,
                           boxsize/2+ystart+boxsize*cy-wsize/2,
                                               wsize, wsize, front);
             }

 }
}



draw_unit_weight_dominant(rx, ry, xstart, ystart, boxsize)
int rx, ry, xstart, ystart, boxsize;
{
 draw_unit_weight_eye(r, rx, ry, xstart, ystart, boxsize);
 draw_unit_weight_eye(l, rx, ry, xstart, ystart, boxsize);
}




draw_unit_weight_threshold(cx1, cy1, xstart, ystart, boxsize)
int cx1, cy1, xstart, ystart, boxsize;
{
}

double mymod(x)
double x;
{
     if (x > 0)
          return(x);
     else
          return(-x);
}
```

```
shade_region(x, y, xsize, ysize, bw)  /* shading regions from points */
int x, y, xsize, ysize, bw;    /* x, y till sizes xsize, ysize with colour bw */
{
     if (bw == 0)
          XFillRectangle(disp,win,gcb,x,y,xsize,ysize);
     if (bw == 1)
          XFillRectangle(disp,win,gcw,x,y,xsize,ysize);
}

draw_square(x1, y1, x2, y2, c)
int x1, y1, x2, y2, c;
{
     XDrawLine(disp, win, gcw, x1, y1, x2, y1);
     XDrawLine(disp, win, gcw, x2, y1, x2, y2);
     XDrawLine(disp, win, gcw, x2, y2, x1, y2);
     XDrawLine(disp, win, gcw, x1, y2, x1, y1);


}


cls()
{
    XFillRectangle(disp,win,gcw,0,0,PIC_SIZE,PIC_SIZE);

    XSync(disp,FALSE);
}
```